Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis - IMESA

FERNANDO BARBOSA MENDES

SOFWARE TRIDIMENSIONAL PARA CONTROLE DE VÉRTICES DO CORPO HUMANO

ASSIS
2009

Fundação Educacional do Município de Assis
Instituto Municípal de Ensino Superior de Assis - IMESA

SOFWARE TRIDIMENSIONAL PARA CONTROLE DE VÉRTICES DO CORPO HUMANO

FERNANDO BARBOSA MENDES

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito de Curso de Graduação, analisado pela seguinte comissão examinadora:

Orientador: Prof. Felipe Alexandre Cardoso Pazinatto

Analisador (1): _____

Analisador (2): _____

ASSIS
2009

Fundação Educacional do Município de Assis
Instituto Municípal de Ensino Superior de Assis - IMESA

FERNANDO BARBOSA MENDES

SOFWARE TRIDIMENSIONAL PARA CONTROLE DE VÉRTICES DO CORPO HUMANO

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, analisado pela seguinte comissão examinadora:

Orientador: Prof. Felipe Alexandre Cardoso Pazinatto

Área de Concentração: Informática

ASSIS
2009

DEDICATÓRIA

Dedico este trabalho primeiramente a DEUS por estar sempre ao meu lado, e por me dar uma família e amigos maravilhosos, a minha família que sempre está ao meu lado e me apóia em todas as minhas decisões e aos meus amigos que independente da situação, se é boa ou ruim estão sempre presentes.

AGRADECIMENTOS

Aos professores da FEMA, que contribuíram para o meu crescimento profissional, além de construir laços de amizade com todos eles. Gostaria de agradecer em especial dois professores que foram fundamentais para a conclusão deste curso, o professor Felipe Alexandre Cardoso Pazinatto que acreditou em meu potencial e me orientou em meu último trabalho de faculdade, e gostaria de agradecer muito a professora e doutora, Marisa Atsuko Nitto, que se dispôs a me ajudar em meu trabalho e apesar de não ser minha orientadora, foi fundamental para que eu não desistisse de meu trabalho (TCC), me proporcionando conselhos, idéias e tudo mais que necessitei para o término deste.

Aos familiares, minha mãe Edna Maria Barbosa por ser carinhosa, atenciosa, compreensiva, enfim por me fornecer tudo que uma pessoa precisa para continuar sempre a luta nunca desistir, meu pai João José Gonçalves Mendes que mesmo não podendo mais estar ao meu lado, foi além de um grande pai, um grande amigo e me preparou para os obstáculos que a vida oferece e minhas irmãs pela paciência.

Aos amigos, que independente de estar perto ou longe, torcem por mim, aos amigos da faculdade em especial a minha turma que apesar de algumas desavenças foram companheiros e solidários.

A uma pessoa muito especial que me acompanhou desde quase o começo da minha faculdade e me proporcionou muita felicidade, companheirismo, carinho, por tudo, obrigado Fabiane Barros.

RESUMO

Este trabalho visa abrir um arquivo texto, que contenha vértices (x, y, z) de um corpo tridimensional, e a partir dele construir controle de rotação do corpo (x, y), controle de zoom (aproximar ou afastar o corpo), e por último construir uma barra de rolagem sobre este corpo, e à medida que se movimenta esta barra (Z) sobre o corpo, mostrará uma pequena parte da imagem onde a respectiva barra estará passando.

Todos estes controles serão acionados através de eventos do teclado do computador, pois a cada clique no teclado de certa letra irá fazer com que o corpo receba alguma ação, podendo ser tanto de rotação, zoom ou com a barra de rolagem, assim modificando o corpo.

Contudo a finalidade do trabalho é mostrar uma aplicação utilizando a linguagem Java em conjunto com a OpenGL.

Palavras-chaves: OpenGL, Processamento de Imagens, Java

ABSTRACT

This work aims to open a text file, which contains the vertices (x, y, z) of a three-dimensional body, and from it build a control body rotation (x, y), zoom control (zoom in or out of the body), and finally build a scroll bar over this body, where as this moves forward slash (Z) on the body, it will show a small part of the image where the respective bar will be passing (region of interest).

All these controls will work through events from keyboard of the computer, and each key press will do with that body receives some action, wich could be rotation, zoom or using the scroll bar, browse across the body.

However the main purpose of work is to show an application using the Java language in conjunction with OpenGL.

Keywords: OpenGL, Image Processing, Java

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

API	Aplication Programming Interface
OBJ	Wavefront File
TXT	Arquivo Texto
3D	Tridimensional
2D	Bidimensional
PHIGS	Programmer’s Hierarchical Graphics System
SGI	Silicon Graphics Inc
ARB	Architecture Review Board
IBM	International Business Machines
SUN	Sun Microsystems
GUI	Graphical User Interface
GLUT	OpenGL Utility Toolkit
GLU	OpenGL Utility Library
FSG	Fahrenheit Scene Graph
RGBA	Red Green Blue Alpha
RGB	Red Green Blue
AMD	Advanced Micro Devices
JVM	Java Virtual Machine

JSE	Java Standard Edition
JEE	Java Enterprise Edition
JDBC	Java Database Connectivity
JSP	Java Server Pages
JME	Java Micro Edition
HTML	Hypertext Markup Language
OOP	Object-Oriented Programming
IDE	Integrated Development Environment

SUMÁRIO

_____**Capítulo 1**

Neste capítulo será feita uma introdução deste trabalho contendo além da introdução, os objetivos, as justificativas e a motivação.

1. INTRODUÇÃO

Com o desenvolvimento tecnológico constante, observa-se que cada vez mais, diversas áreas de negócios tentam se manter atualizadas e com equipamentos cada vez mais sofisticados para melhorar a qualidade e rapidez de seus serviços. A área médica sem dúvida é umas destas, em vista disto, este projeto visa desenvolver um sistema de visualização de um corpo tridimensional para controle de vértices, explorando dentro da área de ciência da computação, a área de processamento de imagem, e uma linguagem de programação em conjunto com uma API (Interface de Programação de Aplicativos) gráfica.

Este trabalho pode-se futuramente integrar com uma área médica (clínica ginecológica), tendo apenas que acrescentar alguns controles para se adequar à respectiva área.

O Processamento de imagens é uma área que tem como objetivo manipular ou modificar objetos gráficos através de tratamentos digitais, aplicando técnicas avançadas para a construção, análise, reconstrução, e posteriormente o reconhecimento da mesma. Dentre os temas científicos comuns em processamento de imagens, destaca-se: a compreensão de imagens, a análise em multi-resolução e em multi-frequência, a codificação e a transmissão de imagens. Entretanto, ao utilizar no computador, denomina-se processamento de imagem digital.

Existem vários tipos de formatos de imagens, as quais podem ser subdivididas em duas categorias maiores: bitmap e vetorial.

Este trabalho usará uma imagem vetorial criada por Alicia Trush, onde o arquivo é do tipo .**OBJ** (wavefront file), e usará a API OpenGL em conjunto com a linguagem Java, também conhecida como JOGL (OpenGL e Java), para a programação, visando abrir um corpo tridimensional, onde o usuário terá a possibilidade de rotacionar e aproximar a imagem.

1.1 Objetivo

O objetivo deste trabalho é colocar em prática alguns conceitos aprendidos ao longo do curso de ciência da computação, principalmente os conceitos adquiridos neste trabalho de pesquisa cientifica, nesta determinada área (Computação Gráfica).

Este trabalho visa mostrar a utilização da linguagem Java em conjunto com a API OpenGL. Aplicada em uma imagem tridimensional de um corpo humano, onde este (corpo humano) será aberto através de um arquivo texto que possui os vértices (eixo x, eixo y e eixo z) do corpo humano, e assim construir controles de rotação (eixo x, eixo y) e um controle de aproximar e distanciar o corpo desta imagem, além de capturar uma parte da imagem através de uma barra de translação que se encontra no corpo tridimensional, exibi-la ao lado do corpo esta respectiva imagem capturada. Entretanto estes controles funcionarão através de eventos do teclado do computador (rotação do corpo, zoom e translação da barra).

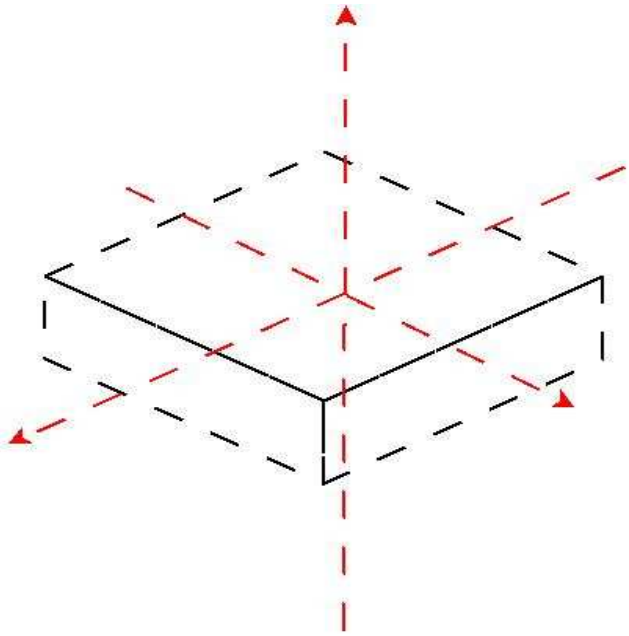

Figura 1 – Objeto Tridimensional

Figura 1. Representação de um objeto tridimensional com coordenadas x, y e z.

Como trabalho futuro ficará a parte de integrar este trabalho a um exame clínico ginecológico, tendo assim que construir um controle, onde este poderá aplicar pontos no corpo tridimensional e salvá-los em um banco de dados, além de um controle de exclusão de pontos do corpo tridimensional.

1.2 Justificativa

As justificativas que norteiam o desenvolvimento deste trabalho são:

- Aprendizagem e utilização de computação gráfica em um meio profissional;
- Aprendizagem e uso da linguagem Java;
- Utilização da OpenGL para Java;
- Utilização de rotinas de processamento de imagens em Java;

1.3 Motivação

A idéia deste trabalho se desenvolveu a partir de um trabalho realizado em uma das matérias da faculdade (Engenharia de Software). Verificou-se junto ao especialista na área médica (área ginecológica) que o mercado não possuía um sistema com uma visualização de um corpo. Entretanto este trabalho não tem como foco desenvolver um sistema específico para a área médica, e sim mostrar na prática a fundamentação teórica que norteiam este trabalho, deixando a integração deste trabalho com a área médica como trabalhos futuros.

1.4 Estrutura do Trabalho

No capítulo 1 serão apresentadas a introdução, o objetivo, a justificativa, a motivação e a estrutura do trabalho.

No capítulo 2 serão apresentados os conceitos básicos de processamento de imagem, arquivo OBJ, API OpenGL, linguagem de programação Java, ambiente de desenvolvimento Netbeans, banco de dados MySql e o sistema operacional.

No capítulo 3 será abordada a descrição do problema e a modelagem do problema.

No capítulo 4 será apresentada à implementação em módulos

No capitulo 5 será feita a conclusão

No capítulo 6 serão apresentados os objetivos para trabalhos futuros

No capítulo 7 serão apresentadas as referências bibliográficas

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será feita uma descrição dos conceitos básicos para o desenvolvimento do projeto. Serão abordados conceitos sobre processamento de imagem, arquivo OBJ, API OpenGL, linguagem de programação Java, ambiente de desenvolvimento Netbeans, banco de dados MySql e o sistema operacional.

2.1 Introdução de Processamento de Imagem

A imagem digital é a materialização de grande parte dos processos da computação gráfica. Neste sentido, ela serve como elo entre o usuário e esses procedimentos, revelando os seus resultados. Pode-se afirmar que a imagem está presente em todas as áreas da computação gráfica, seja como produto final, no caso da visualização, ou como parte essencial do processo de interação, no caso da modelagem [Gomes e Velho, 1994].

A imagem é o resultado de estímulos luminosos produzidos por um suporte bidimensional. Essa é a percepção da imagem no universo físico, seja ela como resultado de um processo intermediado, que pode ser através de projeção do mundo tridimensional [Gomes e Velho, 1994].

A computação gráfica parte de uma informação precisa para obter uma imagem. O processamento de imagens parte da imagem ou de uma seqüência de imagens para obter a informação.

A interpretação e análise quantitativa de imagens representam um ponto de apoio importante em várias disciplinas científicas. A medicina é uma destas áreas, entre outras.

Ao observar do ponto de vista ótico, uma imagem é um conjunto de pontos que converge para formar um todo, ou seja, é um suporte para efetuar troca de informação [Gomes e Velho, 1994].

Atualmente existem duas técnicas complexas de desenvolvimento, onde está associada à análise da informação, e a segunda em melhoria (termo em inglês “ENHANCEMENT”).

2.1.1 História

O processamento de imagens teve o início nos anos 60 no JET PROPULSION LABORATORY – JPL da NASA, sediado em Pasadema, Los Angeles, com o objetivo de obter o máximo de informações das imagens enviadas de volta a terra pelas espaçonaves não tripuladas de exploração lunar e planetária.

2.1.2 Análise

A análise está relacionada ao tratamento onde existe uma descrição da informação presente na imagem. Pode-se chamar de parametrização, onde vários parâmetros são utilizados para descrever diferentes informações dentro de uma imagem [Gomes e Velho, 1994].

A técnica de análise de imagens pode variar significativamente segundo a sua complexidade e a necessidade de tempo de processamento, onde se encontra um nível elevado de complexidade de informação.

2.1.3 Melhoria “ENHANCEMENT”

Este termo associa-se a melhoria da qualidade de uma imagem, com o foco de ser julgado por um observador humano.

Os sistemas dedicados a melhorar a qualidade da imagem trabalham geralmente muito rápidos, assim permitem ao usuário um julgamento sobre várias imagens processadas, pois estes softwares são construídos em “hardware” (máquina) na maioria das vezes [Gomes e Velho, 1994].

2.1.4 Imagem Digital

Ao trabalhar no computador com representações dos modelos discretos de imagem, é importante considerar que podemos idealizar uma imagem em qualquer das possíveis combinações: contínua – contínua, contínua – quantizada, discreta – contínua e discreta – quantizada.

Na prática, a imagem contínua – contínua serve no desenvolvimento dos métodos matemáticos para o processamento de imagens; a imagem discreta – quantizada constitui a representação utilizada por vários dispositivos gráficos. A imagem discreta – contínua, é um formato conveniente para a maior parte das operações com imagens, pois a função imagem assume valores de ponto flutuante que, embora representados por um número finito de bits, aproximam valores reais. Uma imagem do tipo discreta – quantizada é chamada de imagem digital [Gomes e Velho, 1994].

A imagem consiste, essencialmente, das coordenadas dos pixels (elemento de imagem), e da informação de cor de cada pixel. Estes elementos estão diretamente relacionados com a resolução espacial e resolução de cor da imagem. O número de componentes do pixel é a dimensão do espaço de cor utilizado. Sendo assim, cada pixel em uma imagem monocromática (é uma função de intensidade de luz) tem um único componente [Gomes e Velho, 1994].

O gamute de uma imagem digital é o conjunto de cores do espaço de cor quantizado (número de cor reduzido) da imagem. Uma imagem cujo gamute possui apenas duas cores é chamada de imagem de dois níveis, imagem bitmap ou imagem binária, e quando uma imagem cujo gamute possui mais dois níveis é chamada de imagem com meio-tom ou imagem com escala de cinza.

Para exibir uma imagem em algum dispositivo gráfico, o gamute de cor da imagem não pode ser maior do que as cores disponíveis no espaço físico de cor do equipamento.

Na codificação a representação discreta da imagem é quantizada, e a imagem digital resultante é transformada em um conjunto de símbolos organizados de acordo com uma estrutura de dados. Os diversos métodos de codificação têm por objetivo obter um código compacto de imagem. Desta forma, estão diretamente associados com as diversas técnicas de compressão de imagens.

Na compressão de imagens, a quantização de uma imagem, permite uma redução do número de bits (digito binário) utilizado para armazenar o seu gamute de cores. Deste modo reduz o espaço necessário para o armazenamento da imagem, e diminuí-se o volume de dados no caso de transmissão da imagem através de algum canal de comunicação.

Quando uma imagem é exibida em um dispositivo matricial, como exemplo um monitor, a sua representação matricial é mapeada na matriz de pontos do monitor. O formato físico do pixel do monitor é dado pela função de espalhamento do pixel do monitor, ou seja, pela curva de resposta do pixel ao impulso que gera a cor do pixel. A qualidade da imagem exibida leva em consideração diversos fatores desse mapeamento [Gomes e Velho, 1994]. Quatro desses fatores estão relacionados com a geometria e com o espaço de cor do monitor: tamanho físico do pixel, densidade dos pixels, resolução geométrica e resolução de cor do monitor [Gomes e Velho, 1994].

- Tamanho físico do pixel: a dimensão física do pixel é determinada pela distância entre os diversos pontos na matriz do monitor.

- Densidade de pixels: A densidade de pixels é medida pelo número de pixels por área na matriz do monitor.

- Resolução geométrica do monitor: é definida pela ordem da matriz que define a memória da imagem do monitor.

- Resolução de cor do monitor: é o gamute de cor do dispositivo, que determina o número de cores disponíveis para o valor de cada pixel.

2.2 Arquivo OBJ

Os arquivos OBJ (wavefront file), são arquivos de objetos geométricos que podem ser armazenados no formato ASCII (utilizando a extensão do arquivo .Obj), ou em formato binário (utilizando a extensão .MOD), onde este último é de propriedade.

O formato do arquivo OBJ suporta linhas, polígonos e forma livre de curvas e superfícies, onde linha e polígonos são descritos em termos de pontos, enquanto as curvas e superfícies são definidas com os pontos de controle e outras informações, dependendo do tipo de curva.

OBJ não exige qualquer tipo de cabeçalho, porém é comum ao iniciar um arquivo ter uma linha de comentário de algum tipo, ficando assim de livre escolha. As linhas de comentário começam com uma marca (#). Espaço em branco e linhas em branco pode ser livremente adicionado ao arquivo para ajudar a formatação e legibilidade. As linhas são lidas e processadas ate o final do arquivo.

Os seguintes tipos de dados pode ser incluídos em um .OBJ:

Dados de vértices:

Vértices (v)
Textura de vértices (vt)
Normal de vértices (vn)
Parâmetro de espaço de vértices (vp)

Curva livre/atributos de superfície:

Grau (deg)
Base de matriz(bmat)
Tamanho do passo (step)
Tipo de curva ou superfície (cstype)

Elementos:

Ponto (p)
Linha (l)
Face (f)
Curva (curv)
Curva 2D (curv2)
Superfície (surf)

Curva livre/declarações de superfície corporal:

Os valores do parâmetro (parm)
Repetição aparando exterior (trim)
Repetição aparando interior (hole)
Curva especial (scrv)
Ponto especial (sp)
Declaração final (end)

Conectividade entre as superfícies de forma livre:

Conectar (con)

Agrupamento:

Nome do grupo (g)
Suavização do grupo (s)
Mesclando grupo (mg)
Nome do Objeto (o)

Display/Render dos atributos:

Interpolação de Transferência (bevel)

Interpolação de cor (c_interp)
Interpolação de anulação (d_interp)
Nível de detalhe (lod)
Nome do material (usemtl)
Biblioteca de material (mtllib)
Lançamento da sombra (shadow_obj)
Lista de rastreamento (trace_obj)
Curva técnica de aproximação (ctech)
Superfície técnica de aproximação (stech)

O arquivo OBJ que foi usado (de Alicia Trush) contém vértices (v), textura de vértices (vt) e faces (f). Porém este arquivo foi editado para a necessidade deste trabalho, onde foi feito um arquivo de texto, contendo somente os vértices do arquivo OBJ (Alicia), como mostra a figura 2.

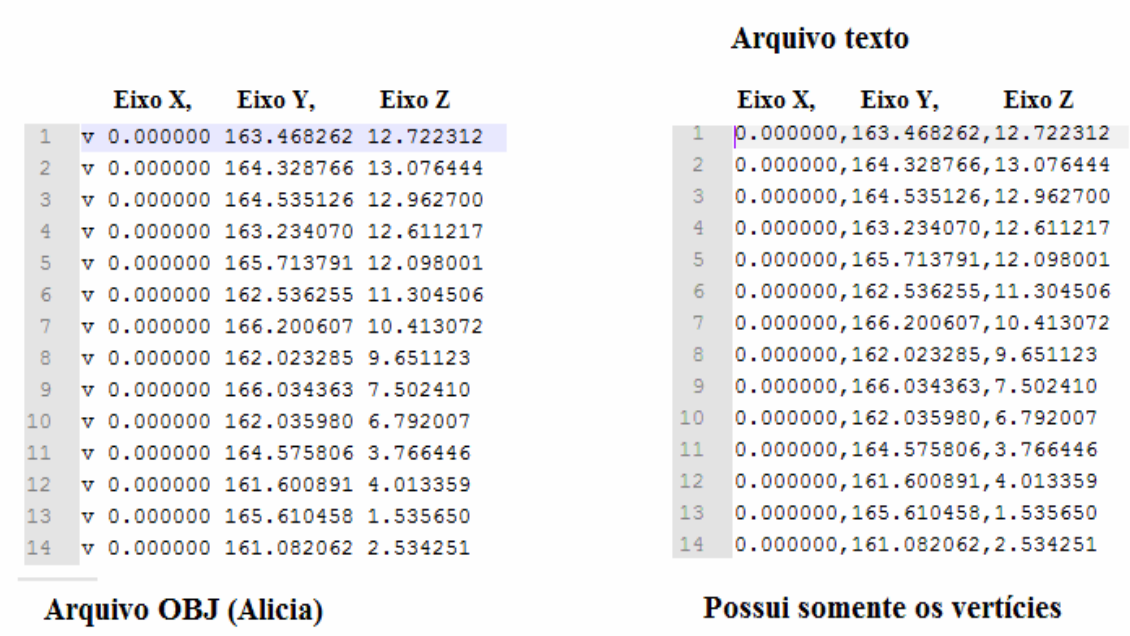

Figura 2 – Arquivo OBJ e TXT.

2.3 Introdução de OpenGL

O OpenGL é uma biblioteca para modelagem (2D e 3D) de rotinas gráficas. É amplamente utilizado em projetos para visualização em geral e também utilizado em muitos jogos, por ser extremamente portátil e rápido. Contudo, sua maior vantagem é a rapidez, pois o OpenGL usa algoritmos complexos que são cuidadosamente desenvolvidos e otimizados [Marcelo e Isabel, 2006].

OpenGL é uma poderosa e sofisticada API(Application Programming Interface – Interface de Programação de Aplicativos), seu funcionamento é como uma máquina de estados complexa, onde a manipulação é feita através de rotinas simples que são chamadas por funções da API OpenGL. Normalmente as rotinas são executadas inteiramente dentro do Hardware da placa gráfica, caso esta tenha suporte. De fato, qualquer placa comprada hoje em dia, possuirá suporte direto das rotinas do OpenGL.

As aplicações OpenGL variam de ferramentas a programas de modelagem usados pela medicina para a criação de imagens de orgãos internos ao corpo humano possibilitando o diagnóstico de males que em outros tempos somente seria possível com intervenções cirúrgicas complicadas e comprometedoras. Além do desenho de primitivas gráficas, OpenGL dá suporte a iluminação, colorização, mapeamento de textura, entre muitos outros efeitos especiais. OpenGL já é aceita como um padrão API para desenvolvimento de aplicações gráficas 3D em tempo real.

A figura 3 mostra uma versão simplificada do pipeline OpenGL. Como uma aplicação faz chamadas às funções API OpenGL, os comandos são colocados em um buffer de comandos. Este buffer é preenchido com comandos, vértices, dados de textura, etc. Quando este buffer é "esvaziado", os comandos e dados são passados para o próximo estágio [Richard e Michael, 2000].

Após a etapa de aplicação das transformações e da iluminação, é feita a rasterização, isto é, é gerada a imagem a partir dos dados geométricos, de cor

e textura. A imagem final, então, é colocada no frame buffer, que é a memória do dispositivo gráfico. Isto significa que a imagem é exibida no monitor [Richard e Michael, 2000].

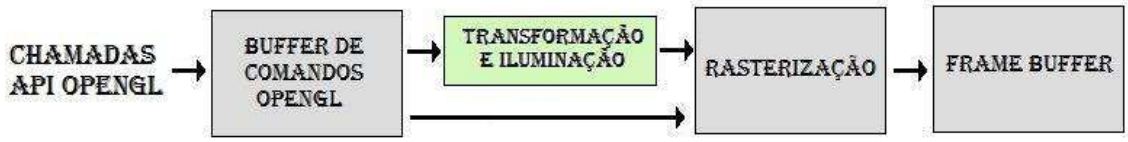

Figura 3 - Versão simplificada do pipeline OpenGL [Richard e Michael, 2000]

2.3.1 História do OpenGL

Na década de 80, construir aplicações com tecnologias 2D e 3D era um grande desafio, pois cada fabricante de hardware, tinha seu próprio conjunto de instruções.

Próximo dos anos 80 surgiu na indústria o PHIGS (Programmer's Hierarchical Graphics System), onde grandes fabricantes começaram a usar este padrão, porém este foi considerado complicado e desatualizado.

No final dos anos 80, surgiu o IRIS GL com a Silicon Graphics Inc. (SGI), que chamou muita a atenção da indústria na época, e foi considerado como uma API gráfica. Consideravelmente mais fácil de usar, a API começou a tornar-se um padrão dentro das indústrias [Mason, 2005].

Entretanto, duas grandes empresas de hardware (Sun e IBM) ainda adotavam o padrão da PHIGS. Isso levou a Silicon a uma grande decisão, pois ela tinha um grande interesse em tornar sua API um padrão público, para que todos os fabricantes de placas de vídeos pudessem adotá-lo.

A Silicon percebeu que a API Iris tinha muito código proprietário e não poderia ser aberta. Ela lidava também não só com desenho 2D e 3D mas com

gerenciamento de janelas, teclado e mouse. Por outro lado a Silicon não queria perder clientes então foi criado a OpenGL.

Desde 1992 á OpenGL é mantido como padrão pelo ARB (Architecture Review Board), um conselho formado por grandes empresas como a IBM, Intel, HP, NVIDIA, SUN, a Silicon Graphics entre outras. Este conselho tem por objetivo manter a especificação e indicar quais novos recursos serão adicionados a cada versão. A OpenGL 3.1 é a versão mais atual [Mason, 2005].

Quando os projetistas criaram a OpenGL, sabiam que os fabricantes de hardware gostariam de colocar seus próprios recursos sem que tivesse que esperar pela ARB para ser aceito oficialmente como um padrão, então para resolver este problema eles deixaram uma maneira de estender o OpenGL, podendo também posteriormente com estas extensões ser aceita pela a ARB e assim fazer parte do padrão oficial [Mason, 2005].

Um dos grandes problemas que os projetistas também pensaram na época seria que muitos hardwares não seriam poderosos para o padrão apresentado. Por esta questão, incluirão extensões de software, que permitiam emular essas funcionalidades.

2.3.2 Arquitetura da OpenGL

A OpenGL é um conjunto de centenas de funções que inclui aproximadamente 250 comandos e funções, que acessa o hardware de vídeo, e a quase todos os recursos dele. Internamente ele funciona como uma máquina de estados, que de modo específico diz à placa de vídeo o que se deve fazer. Onde usando as funções da API, podem-se controlar vários aspectos como cor, transparência, iluminação e outros [Richard e Michael, 2000].

A OpenGL foi projetada para funcionar em qualquer computador mesmo que este não contenha o programa gráfico. Um exemplo disto seriam dois

computadores ligados em rede, ou seja, o computador que gera os comandos é chamado de cliente, enquanto o que recebe e executa os comandos de pintura, transparência ou iluminação é chamada de servidor. O protocolo é padronizado, então é possível que duas máquinas com sistemas operacionais e hardwares diferentes se comuniquem. Se o computador não estiver em rede e rodando OpenGL então pode-se dizer que este é ao mesmo tempo cliente e servidor [Richard e Michael, 2000].

Os comandos da OpenGL não são imediatamente jogados para o hardware. Eles são agrupados para serem enviados mais tarde – o que não só otimiza o uso da rede, mas também abre margem para outras otimizações.

2.3.2.1 Tipos de Dados

Para tornar o código portável, foram definidos tipos de dados próprios para OpenGL. Estes tipos de dados são mapeados dos tipos de dados da linguagem C comuns, que também podem ser utilizados. Como os vários compiladores e ambientes possuem regras diferentes para determinar o tamanho das variáveis da linguagem C, usando os tipos OpenGL é possível “isolar” o código das aplicações destas alterações.

Na figura 4, definida pela Silicon Graphics OpenGL, são apresentados os tipos de dados OpenGL e os valores que podem assumir, os dados primitivos da OpenGL iniciam com o GL e o tipo [Mason, 2005].

Sufixo	Est. Dados	Em C/C++	Nome OpenGL
b	Inteiro 8-bits	signed char	GLbyte
s	Inteiro 16-bits	short	GLshort
i	Inteiro 32-bit	long	GLint, GLsizei
f	ponto flutuante 32-bits	float	GLfloat, GLclampf
d	ponto flutuante 64-bits	double	GLdouble, GLclampd
ub	Inteiro sem sinal 8-bits	unsigned char	GLubyte, GLboolean
us	Inteiro sem sinal 16-bits	unsigned short	GLushort
ui	Inteiro sem sinal 32-bits	unsigned long	GLuint, GLenum, GLbitfield

Figura 4 – Tipos de Dados [Mason, 2005].

2.3.2.2 Nomes das Funções

Os nomes das funções da OpenGL seguem um padrão que indica a qual biblioteca as mesmas fazem parte. Também indica quantos e que tipos de argumentos a função possui. Estas funções possuem uma raiz que representa o comando OpenGL que correspondem às funções. Por exemplo, a função **glColor3f** possui **Color** como raiz. O prefixo **gl** representa a biblioteca gl, e o sufixo **3f** significa que a função possui três valores de ponto flutuante. Enfim os nomes das funções da OpenGL seguem este padrão abaixo, lembrando que em java possui uma diferença, pois antes do prefixo da biblioteca, á um prefixo da biblioteca seguido por ponto final e o restante do função como segue abaixo.

A figura 5 mostra outro exemplo das convenções de nome da OpenGL.

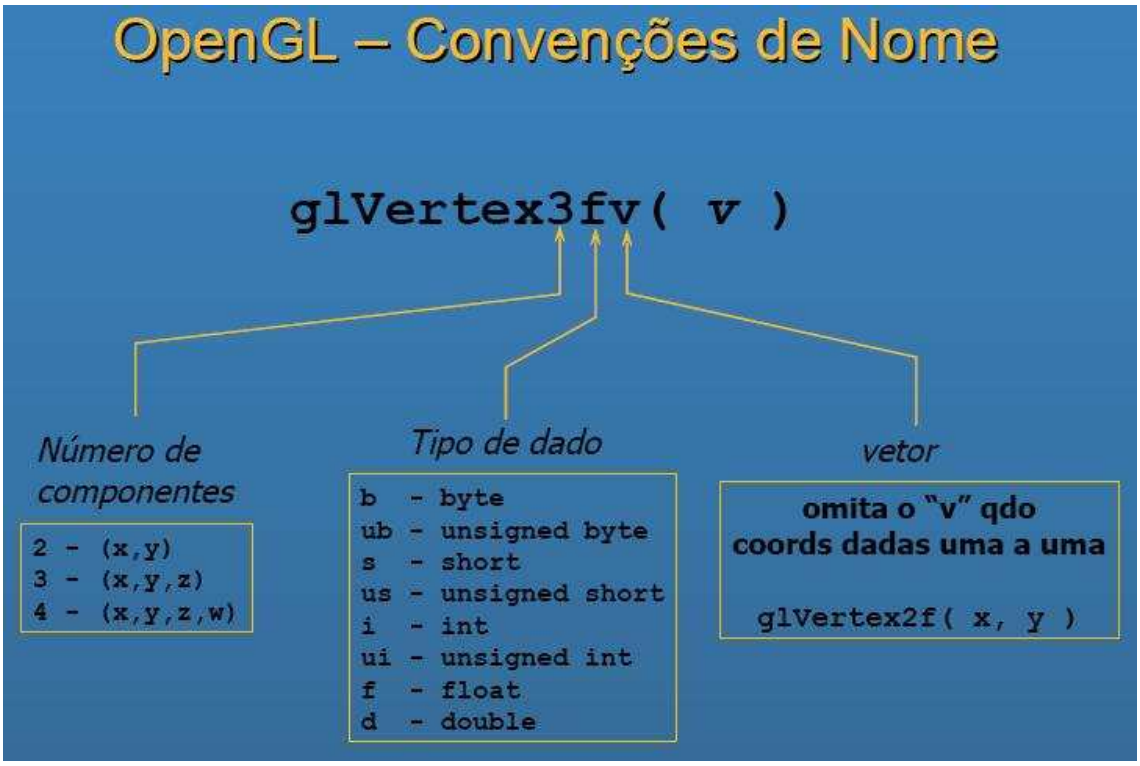

Figura 5 – Exemplo de Nomes de Funções [Richard e Michael, 2000]

Observação: a figura 5 mostra a convenção de nome na linguagem C, pois na linguagem Java, antes do começo do nome vem o prefixo da biblioteca (gl, glu, glut) seguido por ponto final e a convenção normal na linguagem C. No item 2.3.3 serão apresentados às bibliotecas glu e glut.

2.3.3 Bibliotecas

Entre as várias bibliotecas e toolkits (bibliotecas de rotinas) existentes hoje, deve-se destacar as mais utilizadas:

- **GLU - *OpenGL Utility Library***: contém uma serie de funções que encapsulam comandos OpenGL de baixo nível para executar tarefas como, por exemplo, definir as matrizes para projeção e orientação da visualização, e fazer o rendering (renderizar) de uma superfície. Esta

biblioteca é fornecida como parte de cada implementação de OpenGL, e suas funções usam o prefixo glu. Esta é instalada junto ao OpenGl.

- **GLUT -** ***OpenGL Utility Toolkit***: é um toolkit independente de plataforma, para facilitar o desenvolvimento de interfaces, que inclui alguns elementos GUI (*Graphical User Interface*), entre algumas funcionalidades disponíveis pela GLUT estão a criação de janelas e menus pop-up, além de gerenciamento de eventos de mouse e teclado. Esta biblioteca não é domínio público, mas é livre (free). O seu principal objetivo é esconder a complexidade das APIs dos diferentes sistemas de janelas. As funções desta biblioteca usam o prefixo **glut**. É interessante comentar que a GLUT substitiu a GLAUX.

- **GLX -** ***OpenGL Extension to the X Window System***: fornecido como um anexo de OpenGL para máquinas que usam o X Windows System. Funções GLX usam o prefixo **glX**. Para Windows 95/98/NT, as funções WGL fornecem as janelas para a interface OpenGL. Todas as funções WGL usam o prefixo **wgl**.

- **FSG -** ***Fahrenheit Scene Graph***: é um toolkit orientado a objetos e baseado na OpenGL, que fornece objetos e métodos para a criação e aplicações gráficas 3D interativas.

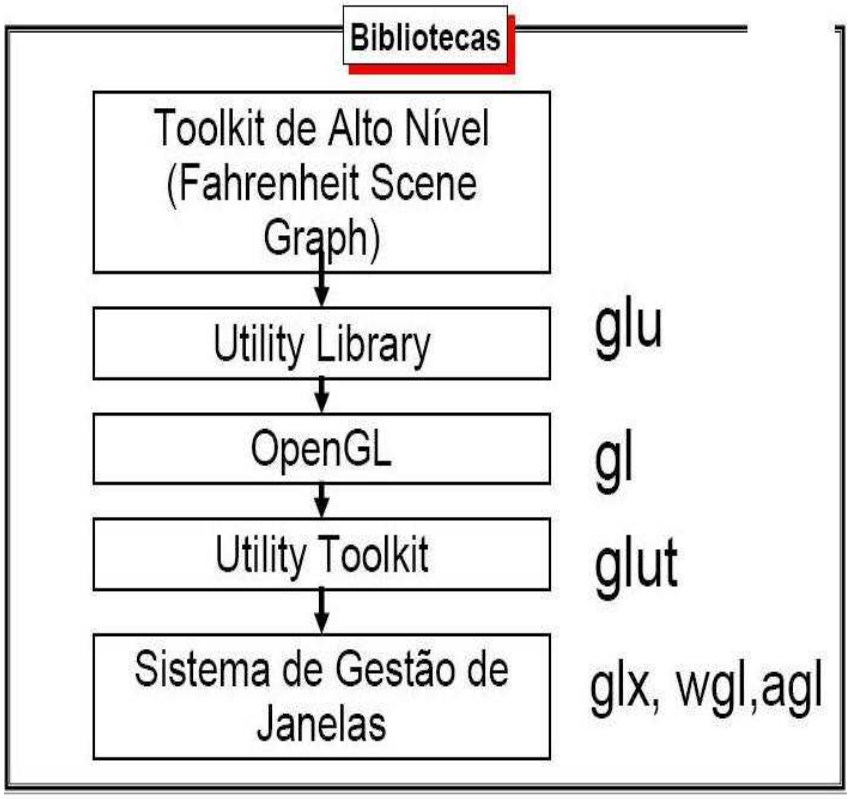

Figura 6 – Bibliotecas da OpenGL [Richard e Michael, 2000].

2.3.4 OpenGL

OpenGL é uma interface para programação de aplicações gráficas. Ela realiza a interface entre software e as chamadas de acesso ao hardware pelo sistema operacional. Esta interface é constituída por cerca de 250 comandos (200 OpenGL e 50 GLUT) distintos que pode-se usar para especificar os objetos e as operações necessárias para produzir aplicações interativas tridimensionais [Mason, 2005].

A OpenGL tem um hardware independe de interface, que pode ser implementado em diversas plataformas de hardware. A OpenGL não fornece comandos de alto nível para a descrição de modelos de objetos 3D. Sendo assim, deve-se criar seu próprio modelo a partir de algumas primitivas geométricas, pontos, linhas ou polígonos.

A OpenGL Utility Library (GLU) fornece muitos recursos de modelagem na OpenGL. GLU é uma parte padrão da OpenGL. Também existe uma biblioteca de alto nível, orientado para objeto toolkit, chamada de Open Inventor (API 3D da Silicon) que é construída nos padrões OpenGL.

2.3.5 Máquina de Estados

OpenGL é uma máquina de estados. Coloca-o em vários estados (ou modos), que, depois, permanecem sem alteração até alguém alterá-lo, ou a menos que uma função seja chamada para isto. Um exemplo, a cor atual é um estado variável. Pode-se definir a cor de branco, vermelha, ou qualquer outra cor, e, posteriormente, cada objeto é desenhado com a cor branca (default), isto, até você definir outra cor. Entretanto pode-se ativar e desativar as variáveis de estado com os comandos gl.glEnable () ou gl.glDisable () [Mason, 2005].

O estado ou modo variável tem um valor padrão, onde por qualquer motivo pode-se alterá-lo. Destaca-se estes seis comandos que são freqüentemente mais utilizados para ler:

- **Variáveis de estado** = gl.glGetBooleanv (), gl.glGetDoublev (), gl.glGetFloatv (), gl.glGetIntegerv (), gl.glGetPointerv () ou gl.glIsEnabled ().
- **Específicas** = gl.glGetLight(), gl.glGetError(), gl.glGetPolygonStipple().
- **Pilha** = gl.glPushAtrib(), gl.glPushClientAttrib(), gl.glPopAttrib(), gl.glPopClientAttrib().

2.3.6 Rendenizar

O núcleo do OpenGL é conhecido como “rendering pipeline”, ou seja, na maioria da implementações do OpenGL, possuem uma mesma ordem de operações, uma série de etapas de processamento.

O figura 7, mostra como a OpenGL faz o processamento de dados. Dados geométricos (vértices, linhas e polígonos), seguem o caminho através da linha de caixas que inclui avaliadores (Evaluators), e por operações de vértices, enquanto os dados de pixel (pixels, imagens e bitmaps) são tratados de forma diferente e a parte do processo. No final, ambos os tipos de dados são submetidos às mesmas etapas finais (Rasterização e por operações de pré-fragmentação), e escritos para o framebuffer [Mason, 2005].

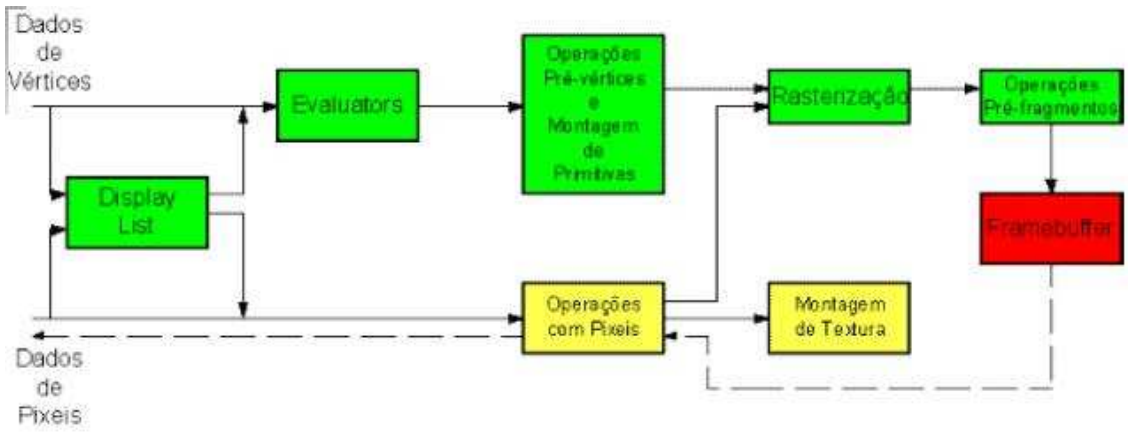

Figura 7 – Rendenização [Mason, 2005].

É importante notar que a OpenGL lida tanto com o desenho na forma vetorial, definido por vértices, como com mapas de bits, definidos pixel-a-pixel. Os principais elementos desse pipeline estão descritos abaixo:

- **Display Lists**: Todos os dados, sejam os de vetorial ou pixels, podem ser salvos num display lists (lista de displays) para uso atual ou posterior;
- **Avaliadores (Evaluators)**: Curvas paramétricas e superfícies podem ser inicialmente descritas por pontos de controle e funções polinomiais;

- **Operações pré-vértices***:* As operações pré-vértice convertem os vértices em primitivas. Então cálculos são realizados para que sejam convertidos do espaço 3D para as coordenadas de tela. Essa etapa ainda realiza operações de cálculos de textura, iluminação, materiais, etc.

- **Montagem de primitivas**: Eliminações de porções que caiam fora do espaço definido por um plano. Passa ou rejeita vértice. Essa fase também aplica os cálculos de perspectiva, o que faz com que objetos distantes pareçam menores em relação a objetos próximos. É aqui também que se decide, se um polígono deve ser desenhado através de pontos ou linhas.

- **Operações com Pixels***:* Os pixeis são primeiramente desempacotados de uma variedade de formatos em um número apropriado de componentes, os dados são escalados e processados por um mapa de pixeis.

- **Montagem de texturas**: Texture objects, com este artifício é possível mudar rapidamente entre um objeto e outro. Permitem a aplicação de imagens como texturas das geometrias, para que elas pareçam mais realistas.

- **Rasterização***:* Conversão de todos os dados em fragmentos. Cada fragmento corresponde a um pixel no framebuffer. Levando em consideração informações sobre sombra, tamanho, largura e outros;

- **Operações sobre fragmentos***:* operações finais são aplicadas aos fragmentos, tais como recortes, operações de máscaras de bits são aplicadas.

2.3.7 Visualizar

OpenGL realiza as seguintes tarefas, a posição e orientação dos modelos em 3D espaciais e como estabelecer o local em 3D também do ponto de vista, isto ajuda a determinar exatamente a imagem na tela.

Uma série de três operações converte um objeto 3D do computador em coordenadas de pixels de posições na tela.

Dentro de uma matriz de multiplicação, incluem modelagem, visualização, projeção e operações. Onde estas operações incluem rotação, translação, projeção ortogonal e perspectiva. Geralmente usamos várias transformações para desenhar uma cena [Marcelo e Isabel, 2006].

A partir da cena já começa a se formar uma janela retangular, onde objetos que estão fora da janela devem ser cortados.

Finalmente deve ser estabelecida uma correspondência entre a tela, e a transformação das coordenadas de pixels.

2.3.8 Cor

Existe dois modos diferentes que a OpenGL trata as cores. Um deles é o modo **RGBA** e o outro **INDEXADO.** Isto depende da biblioteca que se está utilizando para interfacear com o sistema de janelas [Mason, 2005].

RGBA é um padrão utilizado em diversos sistemas e contém os componentes de cores: vermelho, verde, azul e alfa, onde os três primeiros representam as cores primárias e são lineares (0.0 a 1.0). Por sua vez o componente alfa é utilizado em operações de mistura e transparência.

INDEXADO composto por um mapa de cores. Onde este mapa guarda em cada índice os valores para cada componente primário (RGB). Estas cores são trabalhadas pelo próprio índice e não por seus componentes.

Um detalhe importante é que a OpenGL não tem rotinas específicas para alocação de cores, sendo o sistema de janelas responsável por este trabalho [Mason, 2005].

2.3.8.1 Cor do Computador

Existe uma grande variação entre as diferentes plataformas de hardware gráfico, tanto no tamanho da matriz de pixels quanto no número de cores que podem ser exibidos em cada pixel. Em qualquer sistema gráfico, cada pixel tem a mesma quantidade de memória para armazenar a sua cor. O tamanho de um espaço de memória é geralmente medido em bits. Um buffer de 8-bits poderia armazenar oito bits de dados (256 possíveis cores diferentes) para cada pixel. O tamanho do buffer possível varia de máquina para máquina [Mason, 2005].

2.3.8.2 RGBA

Em RGBA, o hardware anula certo número de bitplanes para cada um dos R, G, B e A componentes (não necessariamente o mesmo número para cada componente). Os R, G, B e os valores são normalmente armazenados como números inteiros, em vez de números de ponto flutuante, e eles são dimensionados para o número de bits disponível para armazenamento e recuperação. Por exemplo, se um sistema tem oito bits disponíveis para o componente R, inteiros entre 0 e 255 podem ser armazenados, assim, 0, 1, 2, ..., 255, no bitplanes iria corresponder a valores de R 0 / 255 = 0.0, 1 / 255, 2 / 255, ..., 255/255 = 1,0. Independentemente do número de bitplanes, 0.0 especifica a intensidade mínima e de 1,0 determina a intensidade máxima [Mason, 2005].

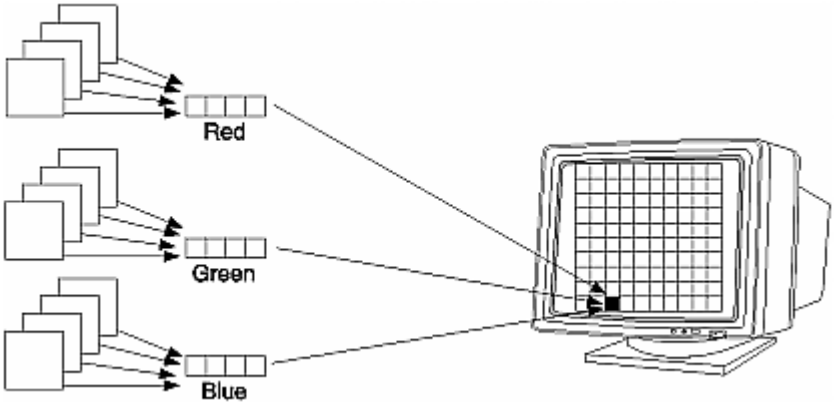

Figura 8 - RGB- Valores de um Bitplanes [Mason, 2005].

2.3.8.3 Mapa de Cor

Um computador armazena a cor índice no bitplanes para cada pixel. Em seguida, os valores de referência da cor no mapa bitplane, são pintados com o correspondente vermelho, verde e azul dos valores do mapa de cor, como mostrado na figura 9 [Mason, 2005].

No modo índice, o número de cores é limitado pelo tamanho da cor do mapa e o número de bitplanes disponíveis. Onde a dimensão da cor do mapa é limitada pela quantidade de hardware dedicado a ele.

Já no modo RGBA, cada cor de pixel é independente de outros pixels, ou seja, se no modo índice o conteúdo de uma entrada mudar a cor, então todos os pixels da mesma cor do índice irão mudar, pois seus bitplanes partilham da mesma cor de mapa.

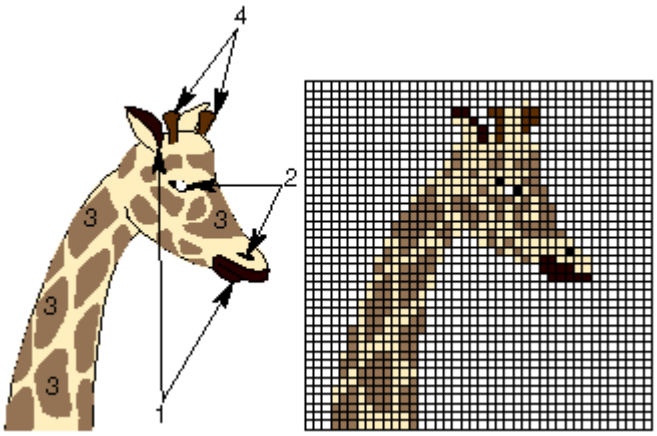

Figura 9 – Mapa de Cor [Mason, 2005].

2.3.9 Iluminação

OpenGL fornece dois modelos para colorização: uma variação de tonalidades com o (GL.GL_SMOOTH) e também pode-se desenhar e usar somente uma cor de preenchimento com o (GL.GL_FLAT).

OpenGL considera que a luz é dividida em quatro cores independentes:

- Ambiente: é a luz que vem de todas as direções, ou seja, a luz refletida no ambiente.
- Difusa: luz que vem de uma determinada direção, atingindo a superfície onde é refletida em todas as direções.
- Especular: luz que vem de uma direção e reflete somente em uma direção.
- Emissiva: simula a luz que se origina de um objeto

Como já descrito a OpenGL calcula a cor de cada pixel em um final, onde é exibido a cena no framebuffer. Parte do cálculo depende da iluminação que está sendo usada, e a forma que os objetos absorvem ou refletem a luz [Mason, 2005].

Figura 10 mostra duas versões da mesma cena (uma única esfera), com iluminação e sem [Mason, 2005].

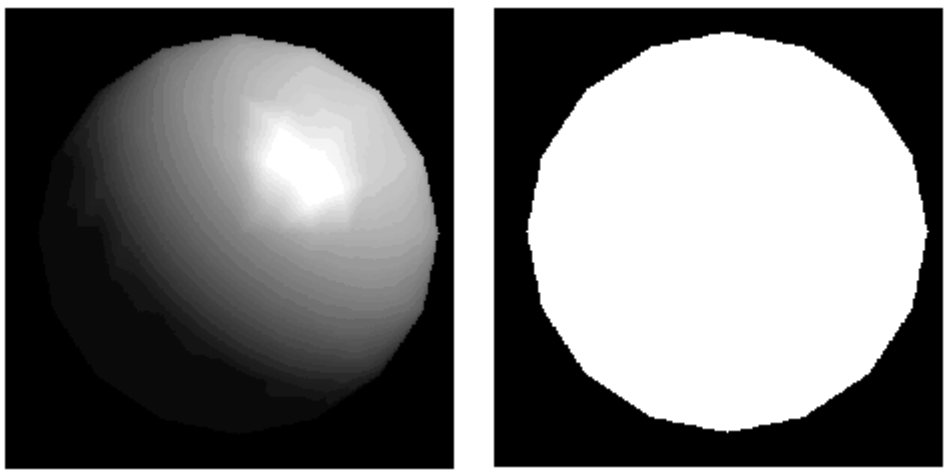

Figura 10 – Iluminação [Mason, 2005].

Na OpenGL, pode-se manipular a iluminação e os objetos em uma cena para criar muitos tipos diferentes de efeitos.

2.3.9.1 Focos

Pode-se ter uma fonte de luz posicional, para atuar como foco, onde limita a forma de luz que ela emite num cone. Para poder criar um foco através de um cone, deve-se determinar a propagação do cone de luz.

Para especificar o ângulo entre o eixo do cone e um raio ao longo da borda do cone, utiliza-se o parâmetro GL.GL_SPOT_CUTOFF. O ângulo do cone no ápice é, então, por duas vezes este valor, como mostrado na figura 11 [Mason, 2005].

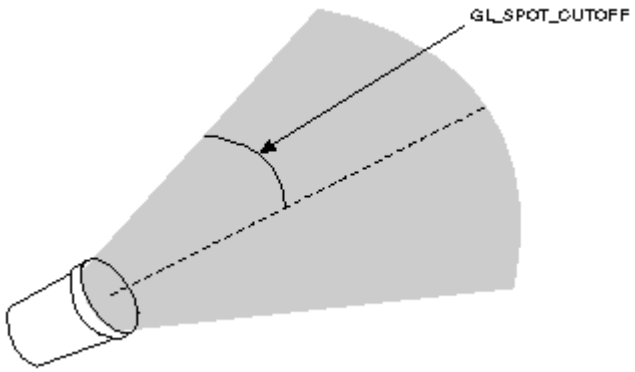

Figura 11 – Focos [Mason, 2005]

2.3.9.2 Misturar

O capitulo de cores discorreu sobre a mistura de cores, mas não foi levado em consideração o componente Alfa. Como descrito naquele capítulo onde o monitor (tela) emite as cores vermelho, verde e azul(RGB), a tonalidade de cores é controlada pela mistura dessas cores em diferentes intensidades. Entretanto quando a mistura é permitida, o valor alfa é muitas vezes usado para combinar o valor da cor do fragmento que está sendo processado com o do pixel já armazenado no framebuffer [Mason, 2005].

2.3.10 Listas de Displays

Com as listas de displays, pode-se melhorar o desempenho desde que possa usá-los para armazenar comandos da OpenGL. Isto funciona principalmente quando entra em questão comandos armazenados em cachê (memória). Ou seja, para redesenhar as mesmas primitivas geométricas varias vezes, ou para aplicar alterações que tenham ocorrido várias vezes [Marcelo e Isabel, 2006].

Executado localmente, muitas vezes pode-se melhorar o desempenho, armazenando freqüentemente os comandos utilizados em uma lista. Alguns

hardwares gráficos podem armazenar listas na memória dedicada do visor ou em um formulário contido em hardware gráfico.

2.3.11 Mapeamento de textura

Mapeamento de texturas é uma ferramenta poderosa para obter um maior realismo nas cenas geradas na OpenGL. Textura refere-se a uma imagem bidimensional, onde ela é aplicada à superfície de um objeto durante o processo de desenho. Essa textura pode ter uma, duas ou três dimensões [Marcelo e Isabel, 2006].

Este texto somente irá tecer uma breve idéia de mapeamento de textura, pois, apesar de ser um assunto importante dentro da OpenGL não será utilizado dentro deste projeto. A figura 12 mostra um mapeamento de textura.

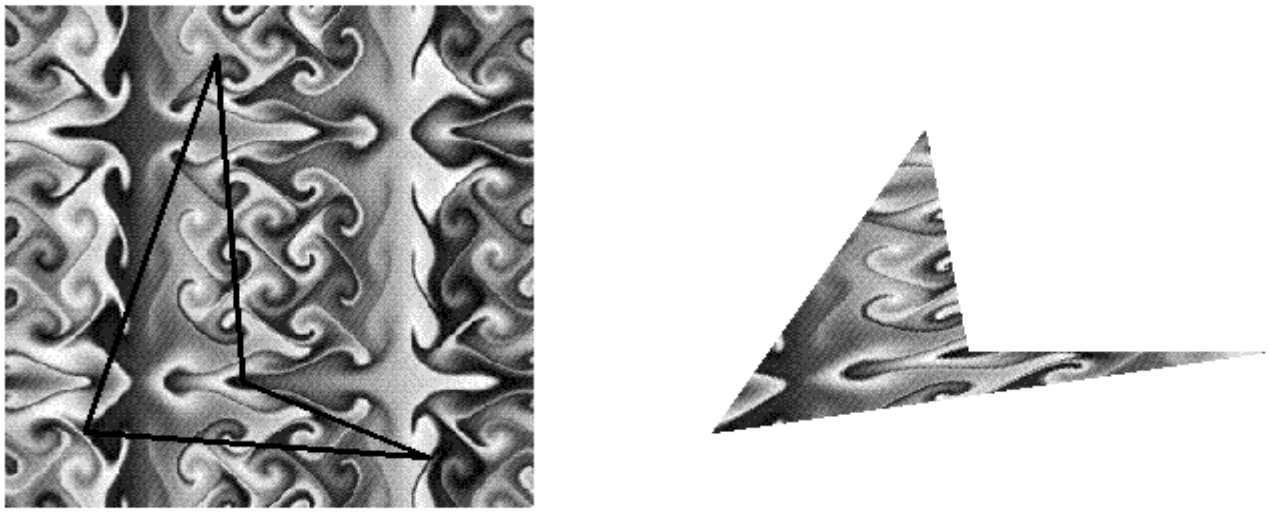

Figura 12 – Mapeamento de textura [Mason, 2005].

2.3.12 Seleção

Antes de utilizar o mecanismo da seleção, deve-se primeiro chamar a cena do framebuffer, e então pode entrar em modo de seleção, ou seja,

redesenhar a cena. Contudo quando estiver em modo de seleção o conteúdo não muda até que o mesmo seja finalizado através da chamada de projeção [Mason, 2005].

2.3.13 Feedback

O Feedback é parecido com a seleção, como descrito acima. Uma vez que você estiver em qualquer dos modos, não são produzidas atualizações, ou seja, os pixels da tela são congelados. O desenho não ocorre, e informações sobre primitivas que teriam sido prestados são enviadas de volta para a aplicação [Richard e Michael, 2000].

A diferença fundamental entre os modos de seleção e de feedback é que a informação é enviada de volta. No modo de seleção, são atribuídos nomes e devolvidos para uma matriz de valores inteiros. No modo feedback, as informações são transformadas em primitivas e assim enviadas de volta na forma de uma série de valores de ponto flutuante.

2.3.14 Transformações Geométricas

São usadas para manipular um modelo. Através delas é possível mover, rotacionar ou alterar a escala de um objeto. A aparência final da cena ou do objeto depende muito da ordem na qual estas transformações são aplicadas.

A biblioteca OpenGL é capaz de executar transformações de escala, rotação e translação através de multiplicação de matrizes e podendo também ser transformada em apenas uma matriz, e assim através de uma única operação fazer várias transformações. A transformação geométrica na OpenGL é armazenada internamente dentro de uma matriz [Marcelo e Isabel, 2006].

gl.glPushMatrix () e gl.glPopMatrix () = são funções que evitam a composição de matrizes a cada nova alteração.

gl.glTranslatef (Tx, Ty, Tz) = é uma função de translação, onde a matriz atual é multiplicada por uma matriz de translação baseada nos valores dados. Seus valores podem ser passados como float ou double.

gl.glRotatef (x, y, z) = é uma função de rotação, que pode receber números float e double como parâmetro. E sua matriz atual é multiplicada por uma matriz de rotação de “Ângulo”.

gl.glScalef (Ex, Ey, Ez) = é uma função de escala, onde sua matriz atual é multiplicada por uma matriz de escala baseada nos valores dados, e seus números podem ser float ou double como parâmetro.

2.4 Linguagem Java

2.4.1 História do Java

Java foi desenvolvida por um grupo de pesquisadores da SUN por volta de 1990 e 1991, pouco antes da explosão da Internet [9]. Fazendo parte de um projeto de desenvolvimento de suporte de software para eletrônica de consumo (geladeiras, lavadoras...), esta equipe coordenada por James Gosling, se chamava de Green Team [Deitel e Deitel, 2001].

Essa linguagem possui estrutura muito semelhante à da linguagem C. Java tem em comum com a linguagem C++ o fato de ser orientada a objetos e mantém com a linguagem C um alto grau de semelhança. Esse paradigma de programação consiste de um grau a mais na abstração da programação, em comparação com a programação estruturada, e tem se mostrado extremamente útil na produção de programas cada vez mais sofisticados, em menor tempo e com maior qualidade.

A programação orientada a objetos é hoje universalmente adotada como padrão de mercado, e muitas linguagens tradicionais foram aperfeiçoadas para implementar esse paradigma, como C++, Object Pascal, etc.

O nome dado a essa linguagem de programação, Java, é o nome de uma ilha do Pacífico, onde se produz certa variedade de café. A inspiração bateu à equipe de desenvolvimento ao saborear esse café em uma lanchonete local, onde se encontravam vários profissionais da área de software para reuniões. Porém este não foi seu primeiro nome, pois já o chamaram de Oak, devido a um carvalho que se podia ver da janela de onde o desenvolvedor da Sun que o "inventou" trabalhava. Entretanto, nesta época havia um produto de nome muito parecido, então renomearam para Java.

2.4.2 Introdução de Java

Java é uma linguagem orientada a objeto. É robusta, com o código aberto, bem documentada e fácil de aprender. Criada inicialmente para o desenvolvimento de pequenos aplicativos e programas de controle de eletrodomésticos e eletrônicos. A Java é portável (roda em qualquer ambiente), multitarefa, segura e dinâmica [Deitel e Deitel, 2001].

Java se tornou tão atraente pelo fato de programas escritos em Java pudessem ser executados virtualmente em qualquer plataforma.

A diferença de Java para as outras linguagens é sua independência de plataforma, embora seja compilada. Utilizando uma máquina virtual diferente em cada plataforma onde funcionam, os programas em si depois de compilados podem ser transportados de um ambiente para outro sem modificações. E para ambientes diferentes como:

- Processadores diferentes: Intel, AMD, etc
- Sistemas Operacionais diferentes: Windows, Linux, MacOs, Solaris, etc

Com isso pode-se desenvolver um programa para Windows que funciona em Linux, MacOs, etc.

Algumas características importantes:

- Orientação a Objeto
- Alto nível
- Tipos de dados Estáticos
- Multi-thread
- Extensível
- Bem estruturada
- Acesso a banco de dados

Em Java podemos escrever o código em um editor de texto, como o bloco de notas do Windows e salvamos o arquivo com a extensão “.java”. Usamos o compilador **javac** para obter o programa executável “.class”. Os arquivos “.class” não possuem código executável (bytecode) para o processador do ambiente de desenvolvimento, como acontece com os arquivos executáveis de outras linguagens. Os códigos são bytecodes para uma Máquina Virtual Java (JVM – Java Virtual Machine).

Para executá-lo usa-se o lançador de programas Java que aciona a Máquina Virtual para executar o programa.

O máquina virtual (JVM) por sua vez traduz os bytecodes java para o processador específico onde está sendo executado o programa [Deitel e Deitel, 2001].

A plataforma Java inclui além da Linguagem de Programação e da Java Virtual Machine (JVM), a API – Biblioteca de Classes.

A biblioteca possui um grande conjunto de classes que implementam várias ferramentas necessárias para os programadores. A maior parte das implementações do Java estão dentro da biblioteca.

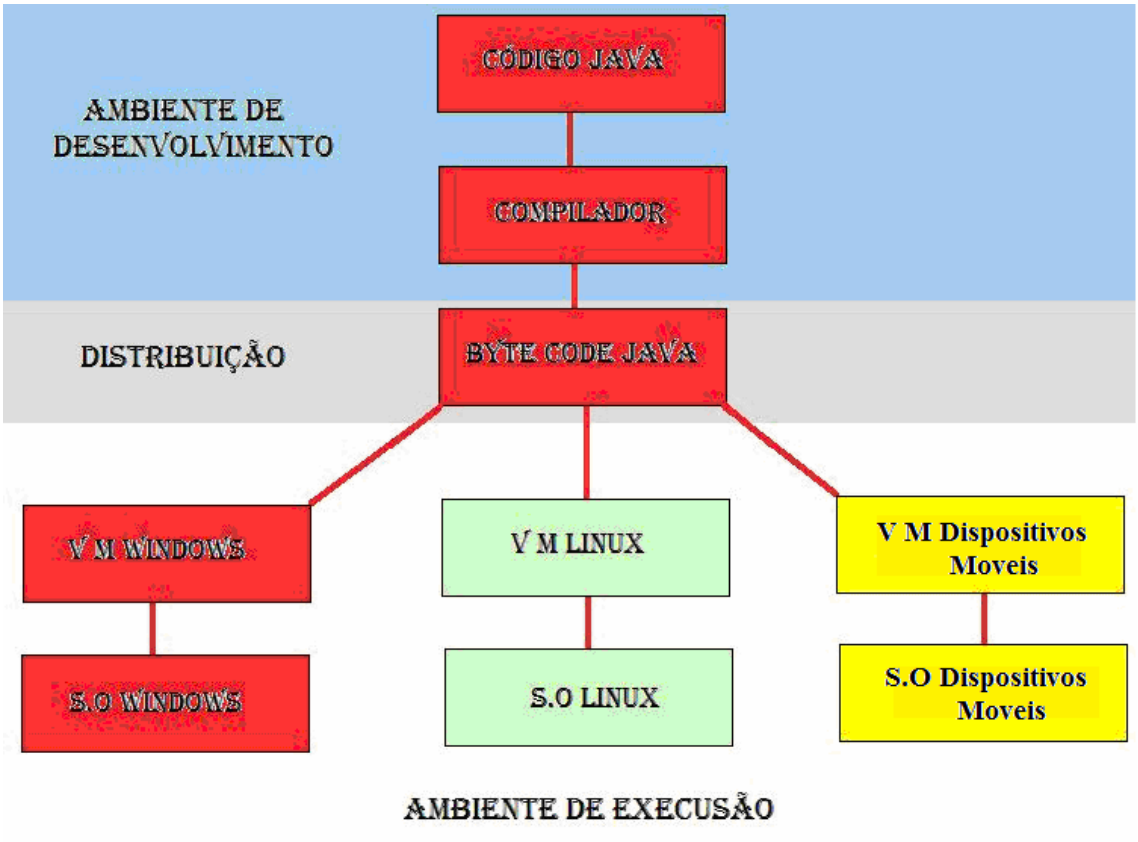

Figura 13 – Compilação e execução de um programa Java [Deitel e Deitel, 2001].

A figura 13 mostra como acontece a compilação e a execução de um programa Java. De um código Java, que está em um arquivo ".java", o compilador (javac) gera o bytecode, um arquivo ".class". Após isso uma máquina virtual Java (JVM) executa o bytecode e roda o programa. Assim, a vantagem é que não é necessário rodar diretamente do sistema operacional, tudo é feito através da JVM, possibilitando a flexibilidade no desenvolvimento, podendo desenvolver projetos em diversas plataformas [Deitel e Deitel, 2001].

Principais tecnologias sobre Java:

JSE

JSE (**J**ava **S**tandard **E**dition): ambiente de desenvolvimento mais utilizado. Principalmente porque está voltado a PCs e servidores, onde há bem mais necessidade de aplicações. Com isto, pode-se dizer que essa é a plataforma principal, já que, de uma forma ou de outra, o JEE e o JME tem sua base aqui.

Esta plataforma é mais indicada para aprender a linguagem, pois é a mais abrangente do Java.

JEE

JEE (**J**ava **E**nterprise **E**dition): é uma plataforma Java voltada para redes, internet, intranets. Além disso, ela contém bibliotecas especialmente desenvolvidas para o acesso a servidores, a sistemas de e-mail, a banco de dados, etc. O JEE foi desenvolvido para suportar uma grande quantidade de usuários simultâneos.

A plataforma JEE contém uma série de especificações, cada uma com funcionalidades distintas. Entre elas, tem-se:

- **JDBC** (**J**ava **D**ata**b**ase **C**onnectivity): utilizado para o acesso a banco de dados.
- **JSP** (**J**ava **S**erver **P**ages): um tipo de servidor Web.
- **Servlets**: para o desenvolvimento de aplicações Web, isto é, esse recurso estende o funcionamento dos servidores Web, permitindo a geração de conteúdo dinâmico nos sites.

JME

JME (**J**ava **M**icro **E**dition): ambiente de desenvolvimento para dispositivos móveis ou portáteis, como telefones celulares e palmtops. Como a linguagem Java já era conhecida e a adaptação ao JME não é complicada, logo surgiram diversos tipos de aplicativos para tais dispositivos, como jogos e agendas eletrônicas, desde que estes aparelhos têm uma JVM.

O JME contém bibliotecas especiais para a atuação em dispositivos portáteis, onde estes dispositivos têm recursos de hardware limitados.

2.4.3 Applets em Java

Diferente de um aplicativo Java, que é executado a partir de uma janela de comando, um applet é um programa Java que executa no APPLETVIEWER (um aplicativo de teste para applets que está incluído com o J2SDK) ou em um navegador web, como o Netscape ou Internet Explorer, então o APPLETVIEWER (ou o navegador) executa um applet quando um documento HTML (Hypertext Markup Language) contendo o applet é aberto no APPLETVIEWER (ou navegador) [Deitel e Deitel, 2001].

Normalmente executam funções bem especificas, onde são usados para adicionar interatividade a aplicações web que não podem ser geradas pelo HTML. O termo applet foi introduzido pela Applescript em 1993, porém para o Java, foram criados em 1995 pela Sun [Deitel e Deitel, 2001].

2.4.4 Orientação a Objetos

A programação orientada a objetos (object-oriented programming - OOP) modela os objetos do mundo real. Ela fornece uma maneira mais natural e intuitiva de ver o processo de programação [Deitel e Deitel, 2001].

A OOP modela objetos do mundo real, seus atributos, seus comportamentos e a comunicação entre objetos.

Alguns conceitos essenciais de Orientação a Objetos:

- **Abstração**: não se repara muito em detalhes, tenta pegar a idéia principal.
- **Encapsulamento**: seu funcionamento interno está protegido e não precisa saber dos detalhes de seu funcionamento interno para poder operá-lo. Ex: um celular, onde não sabemos e nem precisamos saber como é dentro, mas sabemos o que ele faz ou funciona.

- **Classes**: é um conjunto de objetos com características diferentes.
- **Objetos**: é uma instância de uma classe.
- **Associação**: seria uma classe se comunicando uma com a outra, ou seja, um objeto se relacionando com outro objeto.
- **Herança**: é onde uma classe (subclasse) pode estender outra classe (superclasse).
- **Troca de Mensagens**: é uma troca de mensagens de um objeto a outro.
- **Polimorfismo**: é a capacidade de um método ser implementado de varias formas.
- **Métodos**: define as habilidades de um objeto, ou seja, as ações que fazemos. Ex: Ações de um homem = ouvir, falar.
- **Atributos**: são as características de um objeto, ou seja, os valores. Ex: o atributo NOME = Fernando.

2.5 Netbeans

Netbeans é um ambiente de desenvolvimento, também chamado de IDE (Integrated Development Environment = ambiente integrado para desenvolvimento), para criar aplicativos profissionais de desktop, web e movéis de multi-plataformas. Além de auxiliar os programadores a escrever, debugar e compilar códigos.

Esta IDE é gratuita e de código aberto para desenvolvedores de software da linguagem Java e outras linguagens.

Foi criada em 1996 por dois estudantes da Universidade de Charles, em Praga, mas em 1999 a Sun adquiriu o projeto, e assim em 2000 tornou o netbeans uma plataforma OpenSource (código aberto), disponibilizando o código da IDE.

Funciona em qualquer sistema operacional, desde que este suporte a JVM.

2.6 Banco de Dados (MySql)

O MySql é um sistema de gerenciamento de banco de dados relacional, onde ele armazena dados em tabelas separadas em vez de colocar todos os dados em um só local.

É um software gratuito, e foi construído pelos desenvolvedores do MySql, escrito em C e C++ e funciona em diversas plataformas.

2.7 Sistema Operacional (Windows Vista)

O Windows vista é um sistema operacional, onde sua finalidade é servir de interface entre um computador e o usuário, através de um programa ou um conjunto de programas.

Foi desenvolvido pela Microsoft e lançado em 2007, é um sistema operacional pago.

3. Modelagem do problema

Nesta seção será apresentada a descrição e modelagem do problema deste trabalho

3.1 Descrição do problema

O problema que será visto neste projeto de pesquisa, consiste em desenvolver um sistema, onde abrirá uma imagem tridimensional e nesta poderá rotacionar, aproximar ou afastar a imagem, e ainda capturar uma pequena parte do corpo, por intermédio de uma barra de rolagem feita sobre o corpo, em que à medida que se movimenta esta barra sobre o corpo por meio de eventos de teclado, foi desenvolvido um algoritmo que irá capturar a imagem em que a barra estiver e irá mostrar esta pequena imagem capturada ao lado do corpo tridimensional.

O desenvolvimento deste projeto será feito com o Netbeans, a linguagem de programação Java em conjunto com a API OpenGL, onde o sistema abrirá um arquivo texto com uma imagem pronta (imagem de Alicia Trush) com coordenadas de vértices x, y e z dentro da mesma e assim construir controles sobre esta imagem através de eventos do teclado do computador.

3.1 Modelagem do problema

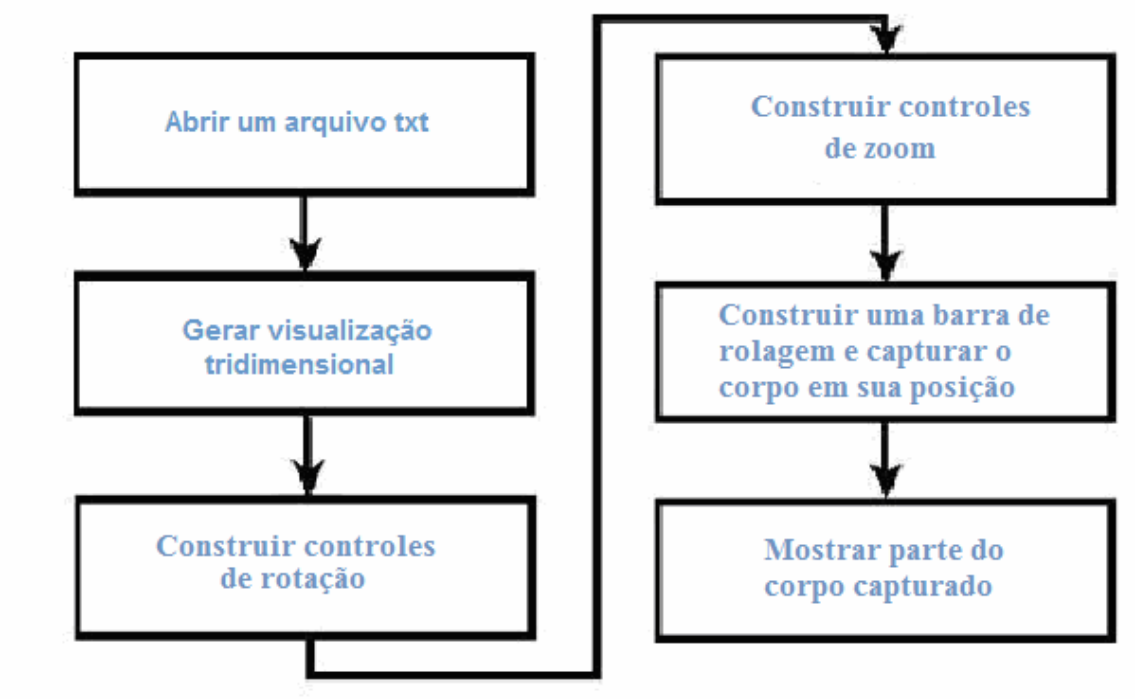

Figura 14 – Estrutura do Trabalho.

A modelagem deste trabalho tem como principal objetivo, abrir uma imagem tridimensional por meio de um arquivo texto (.txt) que contém as coordenadas dos vértices (x, y, z) da imagem, gerando assim por intermédio da projeção das coordenadas a visualização do corpo tridimensional. Tem ainda por objetivo construir controles de: rotações, aproximação (zoom) e uma barra de rolagem neste corpo, em que ao movimentar a barra na posição Z do corpo, capturar uma pequena parte da imagem e mostrar esta imagem ao lado do corpo tridimensional.

4. Implementação

Neste capítulo será apresentado a implementação feita neste trabalho, em que será dividido em módulos.

4.1 Módulo 1: Abrir arquivo texto

Este módulo consiste em abrir um arquivo texto com um modelo tridimensional. Este arquivo deve estar dentro do projeto, e conterá os vértices da imagem tridimensional, como mostra a figura 15.

Observa-se também que estes vértices estão em coordenadas x, y e z separados somente por vírgulas.

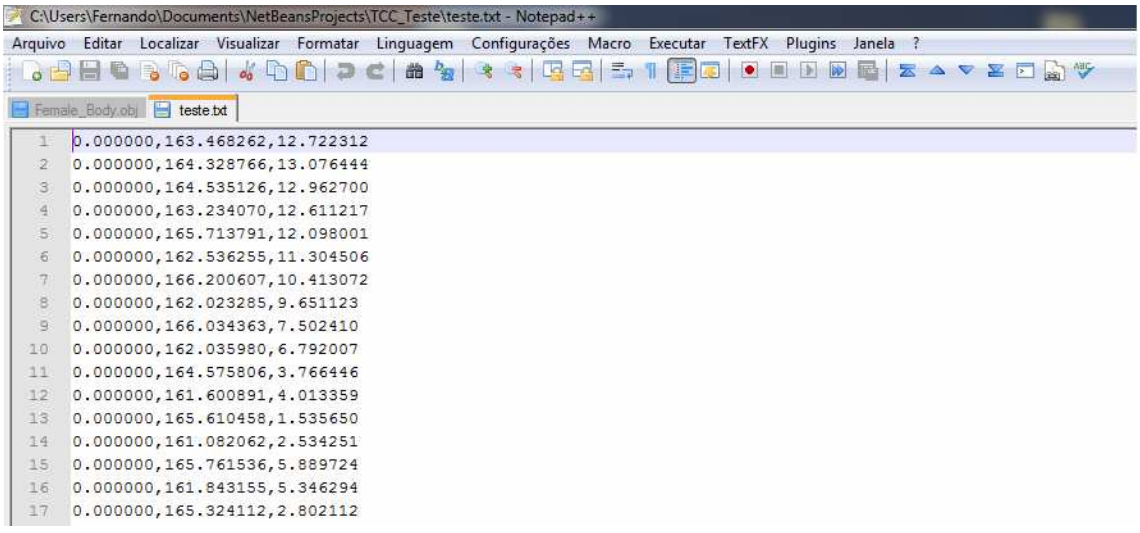

Figura 15 – Arquivo Texto.

A figura 16 mostra uma pequena parte do código para a leitura deste arquivo texto (Main.txt).

```
//A cada iteração, lê uma linha do arquivo e atribui-a a linha:
while ((linha = leitor.readLine()) != null) {

   // utiliza delimitador (,) para dividir os campos
  st = new StringTokenizer(linha, ",");
   String dados = null;

   while (st.hasMoreTokens()) {
       // Pega os campos e joga no "vetor[a]"
      dados = st.nextToken();
      vetor[a] = Float.parseFloat(dados);
      a++;
   }
}
```

Figura 16 – Código para ler o arquivo texto.

Enquanto a linha do arquivo não for nula, o algoritmo continuará lendo o arquivo. Pode-se observar que a (,) tem o objetivo dentro do arquivo de dividir as coordenadas x, y e z, entrando em uma repetição, em que a cada campo separado por vírgula, colocará este campo em um vetor [n posições], já passando o valor dos dados para um vetor de float, iterando posteriormente a variável a.

4.2 Módulo 2: Gerar a Visualização

Este módulo mostra que por meio de algoritmos poderá gerar uma visualização da respectiva imagem tridimensional, neste primeiro momento está imagem será projetado de pontos como mostra a figura abaixo 17.

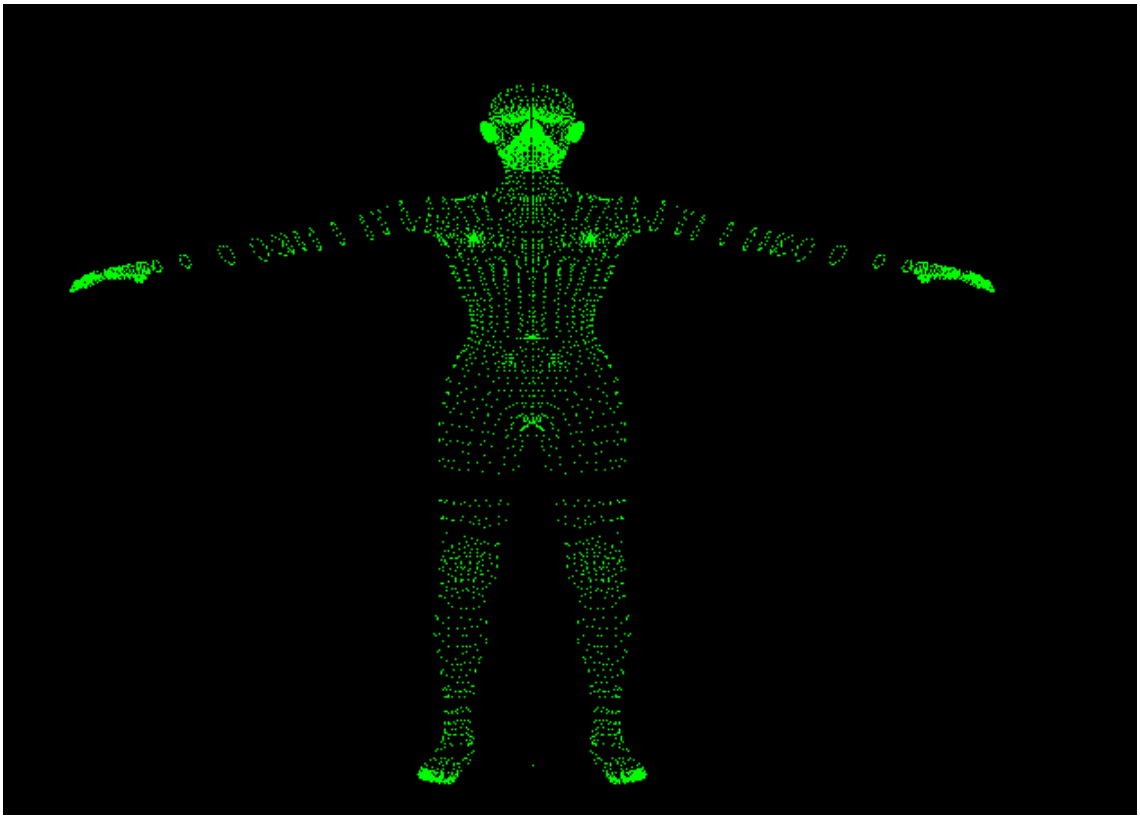

Figura 17 – Corpo Tridimensional.

A figura 18 mostra uma pequena parte do texto onde gera a imagem tridimensional.

```
// Metodo para construir a imagem
public static void imagem3d (GL gl) {

    gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);

    gl.glBegin(GL.GL_POINTS);
    for (int b=0; b<numVet; b+=3) {
        gl.glColor3f (0.0f, 1.0f, 0.0f);
        gl.glVertex3f(  vetor[b], vetor[b+1], vetor[b+2] );
        }

        gl.glEnd();

    //glut.glutSolidTorus(6.0f, 30.0f, 30, 30);

}
```

Figura 18 – Código para construir a imagem.

A figura acima mostra um método em que constrói uma imagem tridimensional de pontos (GL.GL_POINTS), este método recupera o vetor que foi lido no módulo 1 e gera os vértices da imagem através de funções da OpenGL.

4.3 Módulo 3: Controle de Rotação

Este módulo mostra que através de uma função da OpenGL pode-se mover a imagem em três dimensões (x, y e z) . Entretanto para este trabalho será necessário rotacionar o corpo tridimensional, somente na coordenada Y, podendo também rotacionar na posição X. A figura 19 mostra o corpo rotacionado na posição Y, a parte lateral esquerda (esquerda), a parte traseira (centro) e a parte lateral direita (direita) do corpo.

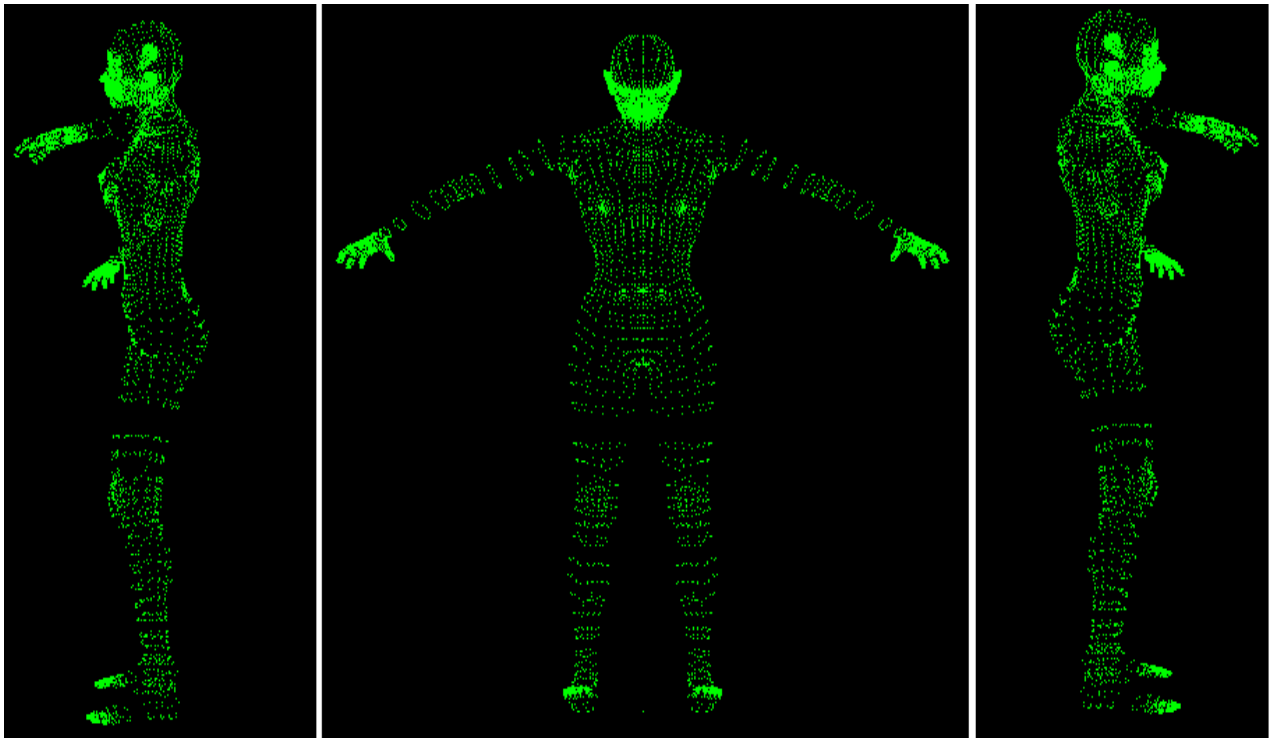

Figura 19 – Corpo rotacionado.

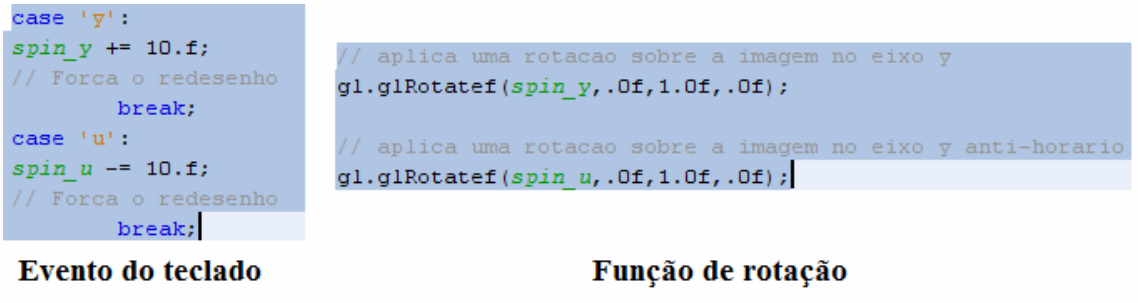

Figura 20 – Código para rotacionar o corpo.

A figura 20 mostra um pequeno trecho de código, que faz a rotação do corpo humano. Foi feito uma função de teclado, assim ao pressionar a tecla **y** ou tecla **u** do teclado do computador, irá atribuir o valor (+ 10 ou - 10) para a variável (spin_y ou spin_u) como mostra na figura 20 (evento do teclado), chamando assim a função display do projeto, em que ela executará novamente o desenho com as respectivas mudanças, pois a função de rotação que se encontra dentro desta função (display) executará novamente, porém com o novo valor da variável (spin_y ou spin_u).

4.4 Módulo 4: Controle de Zoom (Aproximar ou Afastar)

Este módulo mostra que a partir de uma função da biblioteca OpenGL em conjunto com a linguagem Java, pode-se por intermédio de um evento de teclado do computador aproximar ou afastar a imagem. A figura 20 mostra a imagem aproximada e afastada.

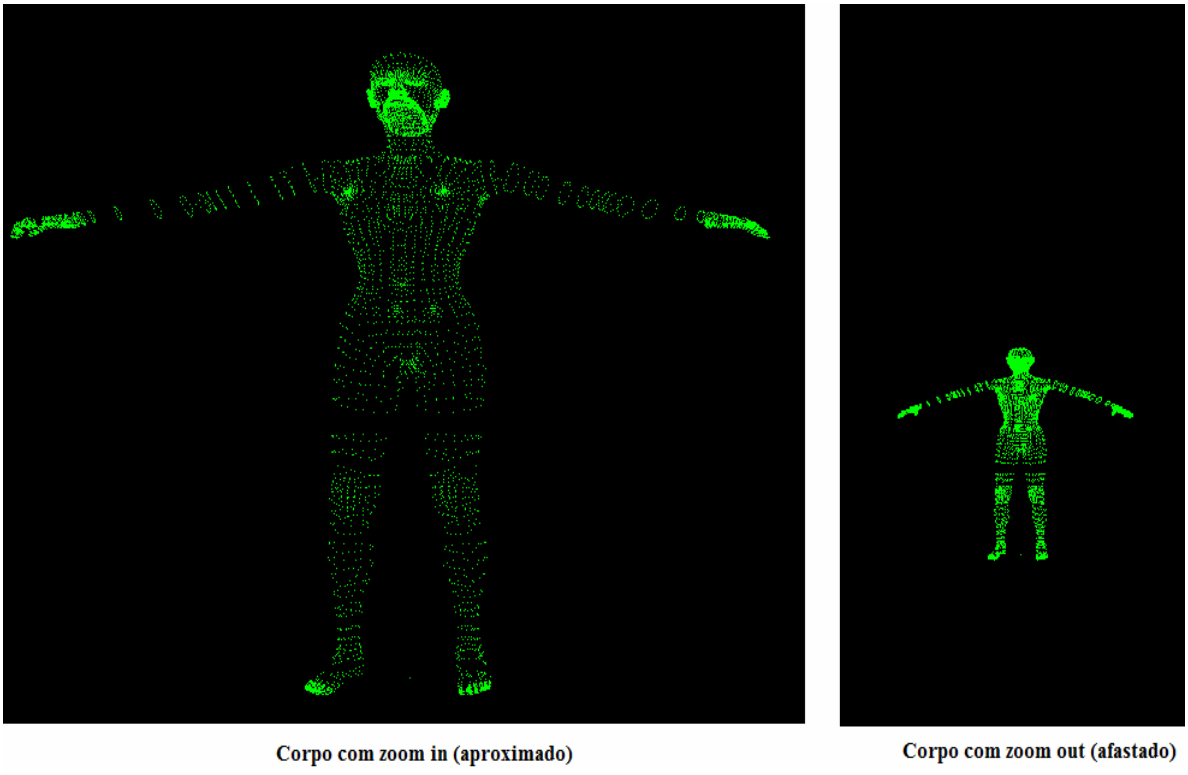

Figura 21 – Corpo com zoom (zoom in e zoom out).

```
//operacaoes de zoom
case 'z':
zoom += 0.2f;
// Forca o redesenho
        break;
case 'a':
zoom -= 0.2f;
// Forca o redesenho
        break;
```

Evento de Teclado

```
// faz um zoom na imagem
gl.glScalef(zoom, zoom, zoom);
```

Função OpenGL

Figura 22 – Código para fazer zoom no corpo.

A figura 22 mostra uma pequena parte do código, que faz o zoom no corpo. Foi feito um evento de teclado, onde a cada vez que ele é chamado, ou seja, clicando no teclado do computador (**z** ou **a** minúsculo), então é atribuído um valor na variável **zoom**, assim após o **break** é chamado uma método de

redesenho, e dentro deste método possui esta função OpenGL que faz o zoom no corpo (eixo x, y e z), com o novo valor atribuído na variável **zoom.**

4.5 Módulo 5: Capturar um parte da Imagem.

Este módulo mostra como foi capturada uma pequena parte da imagem, por meio de uma barra de rolagem, em que à medida que esta barra é movimentada no corpo tridimensional, por intermédio de um evento de teclado, foi feito um algoritmo que recupera estes pontos onde a barra esta e assim estes pontos são desenhados ao lado do corpo tridimensional como mostra a figura 23.

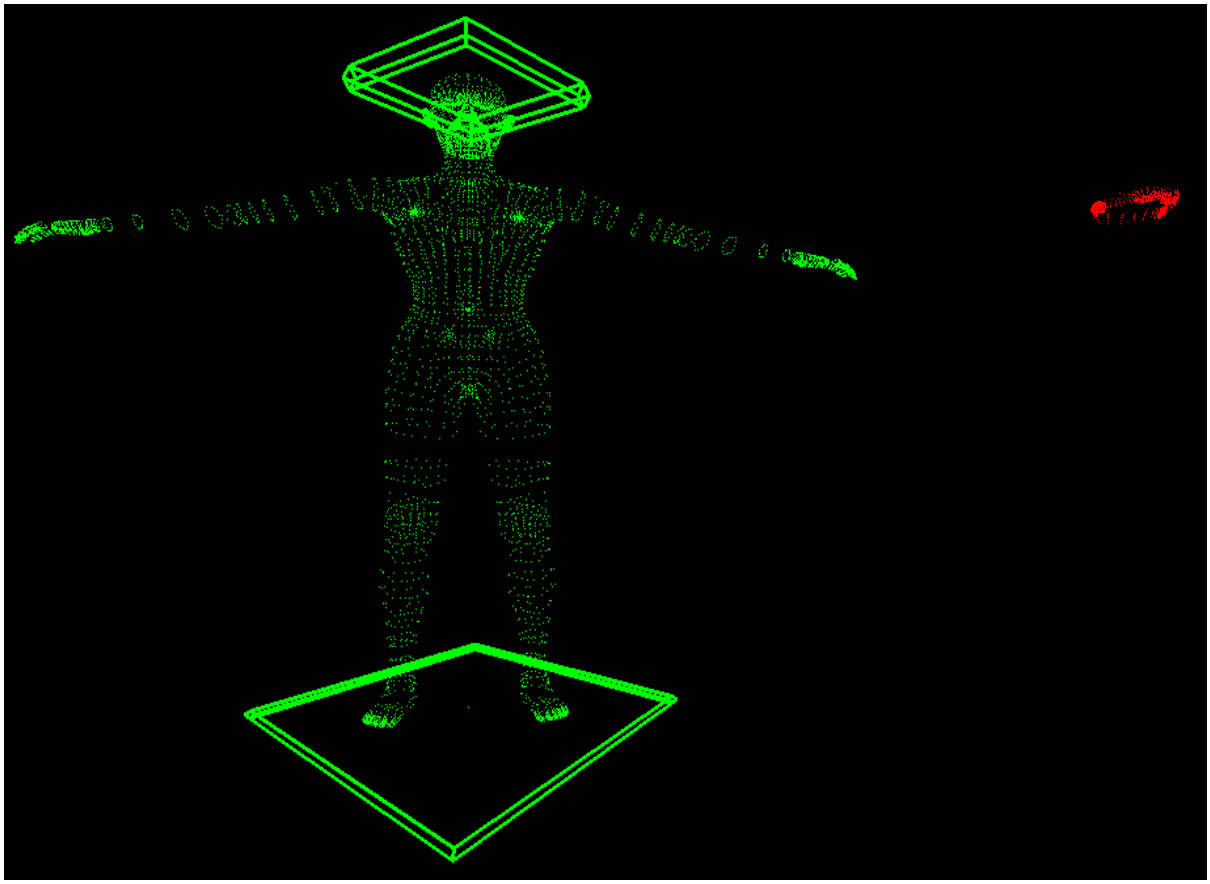

Figura 23 – Imagem capturada.

Pode-se observar que a figura 23 mostra que a barra de rolagem está posicionada na cabeça do corpo tridimensional e ao lado tem uma pequena imagem de pontos (vermelho), ou seja, esta pequena imagem é a imagem captura.

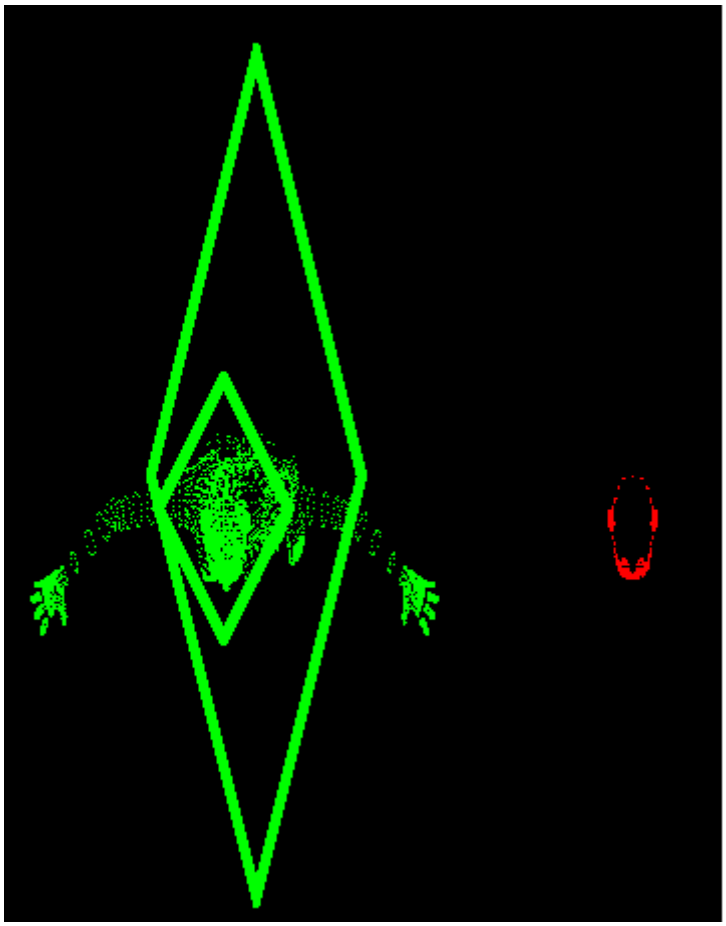

Figura 24 – Imagem capturada movimentada.

A figura 24 mostra a mesma imagem capturada na figura 23, entretanto esta imagem foi rotacionada na posição X e foi um pouco afastada (zoom out).

```
//desenha a barra interior da imagem a de translacao
// aplica uma escala sobre a imagem que é criada
gl.glScalef(1f, -1f, 1f);
// aplica uma rotacao na imagem
gl.glRotatef(90, 1, 0, 0);
// aplica uma translacao sobre a imagem
gl.glTranslatef(0, 0, cont);
//desenha um conjunto de elementos de face solida
//(raio interno, raio externo, numero de planos e aneis)
glut.glutSolidTorus(1.0f, 20.0f, 4, 4);
```

Figura 25 – Código para criar a barra.

A figura 25 mostra um pequeno trecho de código, em que é construída a barra de translação. Nesta é aplicada uma escala (x, -y e z), depois é rotacionada em 90 graus na posição X. Sua translação é controlada através de evento de teclado em que a função de translação da OpenGL recebe uma variável (**cont**) no eixo Z. Esta variável é adicionada, logo após um evento de teclado para este respectivo fim (translação da barra), e a criação da barra é feita por meio da biblioteca **glut**, em que é passado alguns parâmetros para sua criação como mostra a figura 25.

```
//inicio do desenho (imagem 2)
gl.glBegin(GL.GL_POINTS);
// atribui cor a cada vertice
gl.glColor3f(1, 0, 0);
// recebe quais os vertices que irao delimitar as primitivas
gl.glVertex3f(vetor[i-1],vetor[i],vetor[i+1]);
//atribui cor a cada vertice
gl.glColor3f(1, 0, 0);
// recebe quais os vertices que irao delimitar as primitivas
gl.glVertex3f(vetor[posdistancia-1],vetor[posdistancia],vetor[posdistancia+1]);
//final do desenho (imagem 2)
gl.glEnd();
```

Figura 26 – Código para criar uma parte da imagem.

A figura 26 é uma pequena parte do código em que é recuperada uma parte da imagem do corpo dependendo onde estiver à barra de translação e desenhado ao lado do corpo tridimensional, pois para se fazer isto, primeiramente foi construído um método de organização do vetor (método

Selectionsort), logo após este vetor ser carregado com os vértices que estavam no arquivo texto, este método é chamado (Selectionsort). Assim foi feito um algoritmo aonde este irá fazer comparações entre um valor inicial e um valor final na posição Y, após isso irá construir os pontos, onde através da posição Y irá pegar a posição X e Z como demonstra a função da OpenGL (gl.glVertex3f) e desenhar em pontos esta nova imagem.

_____Capítulo 5

5. Conclusão

O sistema implementado apresenta resultados satisfatórios, mostrando a abertura do arquivo texto, a conseqüente aplicação dos dados ao vetor de vértices 3D, a projeção destes vértices pontualmente, no espaço característico da OpenGL e a criação de uma barra de rolagem com a respectiva primitiva gráfica se movimentando pelo corpo tridimensional e capturando uma imagem a medida da translação da barra, além de controle de rotação (eixo X e Y) e um controle de zoom (aproximar ou afastar a imagem).

Como objetivos em pesquisa exigiu-se o estudo aprofundado da API OpenGL, bem como sua interação com a linguagem Java para uma modelagem online do sistema. Exigiram-se ainda conhecimentos em programação e orientação a objetos.

Como objetivos pessoais, obtive uma grande experiência em fazer um projeto com uma grande responsabilidade, além de adquirir grandes conhecimentos nestas respectivas áreas que pesquisei para o desenvolvimento deste projeto, e assim melhorar meus métodos de pesquisa.

6. Trabalhos Futuros

Como trabalho futuro ficará a parte de integrar este projeto em um fim especifico para a área médica (ginecológica), em que se deve construir controles para marcação de pontos neste corpo tridimensional podendo ou não ser salvo em um banco de dados, criar um controle de exclusão de pontos e criar a parte de cadastramento, pagamento, contas a pagar e agendamento de consultas.

Referências Bibliográficas

Livro sobre Processamento de Imagens

Gomes J, Velho L, **Computação Gráfica: Imagem**, Rio de Janeiro, 1994. ISBN: 85-244-0088-9.

Livros de OpenGL

COHEN Marcelo, MANSSOUR H Isabel, **OpenGL: Uma Abordagem Prática e Objetiva**, 1 edição, São Paulo, Editora Novatec, 2006, ISBN 85-7522-084-5.

WOO Mason et al, **OpenGL Programming Guide**, 3 edição, Addison-Wesley Publishing Company, 2005, ISBN 0201604582.

WRIGHT Richard S. Jr. SWEET Michael, **OpenGL SuperBible**, 2 edição, Indianapolis, Indiana: Waite Group Press, 2000, ISBN 1571691642.

Livro de Java

Deitel H.M, Deitel P.J, **Java Como Programar**, 3 edição, Porto Alegre, 2001, ISBN 85-7307-727-1.

Site sobre OBJ

<URL: http://nccastaff.bournemouth.ac.uk/jmacey/RobTheBloke/www/source/obj.html/ >, acesso em Outubro de 2009.

Sites sobre OpenGL

<URL: http://www.sgi.com/products/software/opengl/>, acesso em Maio de 2009.

<URL: http://vinigodoy.wordpress.com/2007/12/26/entendendo-o-opengl/>, acesso em Maio de 2009.

<URL: http://www.inf.pucrs.br/~manssour/OpenGL/>, acesso em Junho de 2009.

<URL: http://www.opengl.org/>, acesso em Junho de 2009

<URL: http://fly.cc.fer.hr/~unreal/theredbook/ >, acesso em Maio e Junho de 2009.

Sites sobre Java

<URL: http://www.javafree.org/index.jf >, acesso em Abril de 2009.

<URL: http://www.devmedia.com.br/canais/default.asp?site=34 >, acesso em Abril de 2009.

<URL: http://www.mundojava.com.br/NovoSite/default.jsp >, acesso em Abril de 2009.

<URL: http://www.dm.ufscar.br/~waldeck/curso/java/introd.html >, acesso em Maio de 2009.

<URL: http://www.dca.fee.unicamp.br/cursos/POOCPP/node3.html >, acesso em Outubro de 2009.

Site sobre Netbeans

<URL: http://www.netbeans.org/ >, acesso em Junho de 2009.

Site sobre MySql

<URL: http://www.mysql.com/ >, acesso em Outubro de 2009.

Site sobre Windows Vista

<URL: http://www.microsoft.com/windows/windows-vista/ >, acesso em Setembro de 2009.