UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

IMPLEMENTAÇÃO DE UM AMBIENTE PARA MODELAGEM DE OBJETOS 3D COM USO DE SWEEPING

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

EDNILSON JOSÉ

BLUMENAU, JUNHO/2001.

2001/1-26

IMPLEMENTAÇÃO DE UM AMBIENTE PARA MODELAGEM DE OBJETOS 3D COM USO DE SWEEPING

EDNILSON JOSÉ

ESTE TRABALHO DE CONCLUSÃO DE CURSO FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Dalton Solano dos Reis — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Dalton Solano dos Reis

_____

Paulo César Rodacki Gomes

_____

Prof. Antônio Carlos Tavares

AGRADECIMENTOS

Em primeiro lugar, a Deus, por tudo.

Aos meus pais, Franciso e Cecília, pela compreensão e apoio durante estes anos de estudo.

Ao meu orientador, prof. Dalton Solano dos Reis, pela atenção dispensada a este trabalho.

Ao meu irmão e colega Odair, pelo apoio e suporte de software nas horas em que precisei.

Aos meus colegas da FURB, em especial a Suzete T. Colling, pela amizade e colaboração.

A todos aqueles que, direta ou indiretamente, contribuíram para a realização deste trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

RESUMO

Este trabalho descreve a implementação de um ambiente para modelagem e visualização de objetos tridimensionais com uso da técnica de *sweeping*. O protótipo permite ler a partir de arquivos, objetos previamente definidos, criar e alterar estes bem como armazená-los para posterior recuperação. Podem ser obtidos objetos de *sweep* através das técnicas de *sweeping* translacional, rotacional e helicoidal. No translacional, o mais simples, um perfil é deslocado ao longo de uma trajetória, no rotacional este perfil gira em torno de um eixo e o helicoidal é obtido através da junção dos outros dois modelos. Implementou-se o protótipo no ambiente de programação Delphi com utilização da biblioteca gráfica OpenGl, o que garantiu bom grau de recursos.

ABSTRACT

This work describes the implementation of an ambient for modelling and visualization of three-dimensional objects with use of the sweeping technique. The prototype allows to read starting from files, objects previously defined, to create and to alter these as well as you store them for subsequent recovery. They can be obtained sweep objects through the techniques of sweeping translational, rotational and helical. In the translational, the simplest, a profile is moved along a path, in the rotational this profile rotates around an axis and the helical is obtained through the junction of the other two models. The prototype was implemented in the programming ambient Delphi with use of the graphic library OpenGl, what guaranteed good degree of resources.

1 INTRODUÇÃO

Um dos mais importantes conceitos em computação gráfica é a modelagem de objetos e imagens. Por modelação entende-se a descrição de objetos e de imagens, que permitem a sua visualização, usando um computador para simular a realidade (PLASTOCK, 1986, p.27).

A modelagem de objetos 3D é necessária em diversas áreas, entre elas a modelagem de objetos que não existam no mundo real, ou seja, a projeção e visualização de um objeto criado (CORRIGAN, 1994, p. 73). As superfícies 3D também são usadas no estudo de objetos que já existem, para que estes possam ser aperfeiçoados.

As superfícies podem ser modeladas utilizando um conjunto de objetos que, interligados formam uma malha. Tornar possível a visualização de uma cena tridimensional em um monitor de vídeo requer a construção de procedimentos que simulem o processo de obtenção de fotografias por meio de câmera. Existem algumas técnicas para este tipo de modelagem, entre elas, Splines, Modelagem Sólido-construtiva e *Sweeping* (CORRIGAN, 1994, p. 54).

Os três principais tipos de modelos geométricos são o aramado (*wireframe*), faces limitantes e sólido. A modelagem de sólidos é um importante aspecto da modelagem geométrica que é usada para criar e comunicar informações sobre sua forma. Ela envolve a criação e manutenção do modelo sólido para futuro acesso e análise.

Segundo VIEIRA (1994, p.14), os métodos de representação de sólidos denominam também as técnicas de modelagem de sólidos, que são as seguintes:

a) instanciamento de primitivas;

b) enumeração de ocupação espacial;

c) decomposição em células;

d) geometria sólida construtiva (*Construtive Solid Geometry* - CSG);

e) representação por *sweeping;*

f) representação por fronteira.

A noção básica embutida nos esquemas de varredura (*sweep*) é muito simples: um conjunto movendo-se através do espaço pode varrer um volume (um sólido) que pode ser representado por um objeto em movimento mais uma trajetória. O local dos pontos gerados por esse processo define um objeto de uma, duas ou três dimensões. Para modelagem de sólidos, a representação por *sweep* é a mais simples de ser compreendida (MORTENSON, 1985, p.455).

As propriedades dos esquemas de *sweeping* translacional e rotacional são em grande parte determinadas por esquemas específicos usados para representar conjuntos bidimensionais.

Este trabalho adota a técnica de modelagem *sweeping*, mostrando trajetória e rotação dos objetos gerados. O protótipo dispõe de uma janela para visualização do modelo, com controles que permitem gerar os três tipos de *sweeping*, bem como controles para a câmara sintética, o que permite a visualização de diferentes pontos de vista.

1.1 MOTIVAÇÃO

São inúmeras as áreas de utilização da computação gráfica, desde engenharia, simuladores de vôo, visualização científica até a área artística. “... a área da computação gráfica e suas aplicações abrem novos horizontes e desafios a todos que reconhecem no computador uma ferramenta para aprendizagem, descoberta e criatividade.” (PLASTOCK, 1986, p.VII).

Na computação gráfica tem-se a modelagem de objetos, onde existem muitas técnicas para construção de modelos tridimensionais e todas apresentam vantagens e desvantagens dependendo da aplicação utilizada. Uma determinada cena pode ser facilmente representada através da técnica de *sweeping*, enquanto que em outras técnicas, ser de difícil ou tediosa construção. Por estes motivos, é que optou-se desenvolver um estudo sobre técnicas de *sweeping* em forma de TCC.

1.2 OBJETIVOS

Os principais objetivos deste trabalho são, o estudo e implementação de um protótipo de software para modelagem de objetos 3D com uso de s*weeping* com visualização de trajetória e rotação.

Os objetivos específicos do trabalho são:

a) Estudo sobre modelagem de objetos 3D, utilizando a técnica s*weeping*;
b) Estudo de ambiente de Câmera Sintética;
c) Implementação do protótipo de software para geração de objetos 3D.

1.3 RELEVÂNCIA

Este trabalho de conclusão de curso apresenta os seguintes aspectos tecnológicos relevantes:

A visualização de cenas tridimensionais por computador tem diversas aplicações. Por exemplo: na engenharia, simuladores de vôo, visualização científica, animação, entre outros.

Ambientes para visualização também podem ser utilizados como ferramentas didáticas para melhorar a instrução sobre o espaço tridimensional e os objetos que o habitam.

1.4 ORGANIZAÇÃO DO TEXTO

O capítulo 1 apresenta a introdução, motivação, os objetivos, a relevância e a organização do trabalho, ou seja, a estrutura geral do trabalho.

Os capítulos 2, 3 e 4 são dedicados aos conceitos Câmera Sintética, OpenGl e Modelagem Tridimensional.

No capítulo 5 é mostrado como foi desenvolvido o protótipo do software, mencionando-se especificação, implementação e o funcionamento.

E por fim, o capítulo 6, onde são expostas as conclusões, analisando as dificuldades, os resultados do trabalho e extensões.

2 CONCEITOS DE CÂMERA SINTÉTICA

Ao longo dos séculos, artistas, engenheiros, projetistas, desenhistas, cartógrafos e arquitetos têm tentado resolver as dificuldades e restrições impostas pelo problema de representar um objeto ou uma cena tridimensional num meio bidimensional – o problema da projeção. O implementador de um sistema gráfico computadorizado depara-se com o mesmo desafio. A capacidade de representar um ponto calculado permite usar o monitor como um instrumento de desenho. A descrição matemática do processo de projeção permite a visualização de imagens de objetos tridimensionais e cenas desejadas (PLASTOCK, 1986, p.157).

2.1 VISÃO GERAL

A câmera é a representação atribuída a um observador de um cenário qualquer, ou seja, é a visão do observador de uma determinada imagem e a representação desta em um plano. É com base na sua localização, orientação, entre outros, que se obtém uma imagem como um todo ou uma parte do cenário disponível (ZINDARS, 1993, p. 5).

A utilização de uma câmera sintética possibilita ao observador visualizar o objeto através de pontos de vista diferentes (posição, alvo, ângulos e foco) (REIS, 1997, p. 20).

Câmera sintética é um conjunto de transformações realizadas em um objeto tridimensional, visto através do sistema de referência do universo, para ser representado em um plano, simulando o efeito de uma câmera fotográfica, isto é, as coordenadas do objeto 3D são ajustados a um plano (SILVA, 2000, p. 30).

2.1.1 UNIVERSO

O universo é um modelo de espaço no qual os objetos a serem exibidos, encontram-se descritos, ou seja, é o espaço disponível que os objetos podem ser representados (ZINDARS, 1993, p. 5).

2.1.2 PLANO DE PROJEÇÃO

Projeção é a forma de representação de figuras em três dimensões numa superfície plana, como por exemplo, a tela do computador (BORGES, 1989, p. 38). Ela representa a

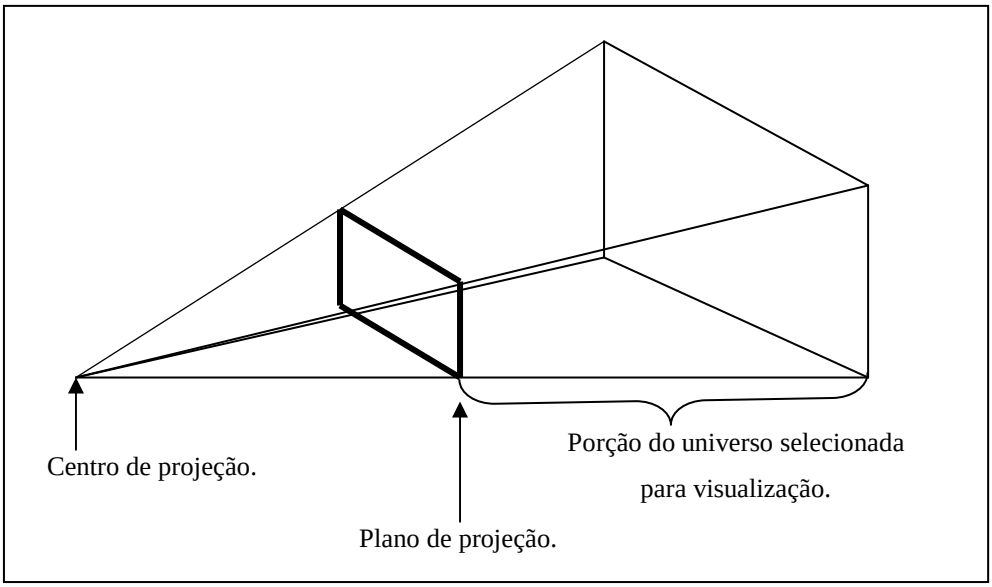
área/plano em que será projetada uma porção do universo selecionada para visualização (Figura 1) (ZINDARS, 1993, p. 6). As projeções são usadas para atingir os objetivos abaixo, ainda que isoladamente (PERCIANO, 1989, p. 162):

a) garantir as medidas, da figura tridimensional, na representação 2D;

b) dar uma visão da figura espacial como um todo, possibilitando a idealização do objeto real;

c) permitir um efeito visual parecido com o de uma fotografia.

Figura 1 - Plano de projeção.



Fonte: (ZINDARS, 1993, p. 6).

2.1.3 VISUALIZAÇÃO DE OBJETOS TRIDIMENSIONAIS

A visualização de objetos tridimensionais num plano consiste em exibir os pontos dos objetos após o cálculo da projeção perspectiva dos mesmos.

Para se calcular os pontos na janela de exibição que irão representar os pontos do objeto, deve-se inicialmente transformar os pontos do sistema de referência do universo para o sistema de referência da câmera, e então aplicar a projeção perspectiva. Por último, mapear as coordenadas do plano de projeções para as coordenadas da janela de exibição[1] (ZINDARS, 1993, p. 19).

[1] Área que se tem disponível para mostrar a porção do universo selecionada.

2.2 SISTEMAS DE REFERÊNCIA

Os sistemas de referência são usados para definir as coordenadas dos vértices de objetos contidos em uma cena tridimensional em relação a determinadas posições ou para transferência para outros dispositivos.

2.2.1 SISTEMA DE REFERÊNCIA DO OBJETO (SRO)

É utilizado para definir as coordenadas dos vértices de cada objeto contido em uma cena tridimensional. A origem deste sistema, normalmente coincide com o centro geométrico do objeto (com a finalidade de facilitar operações de transformação de coordenadas, tais como, rotação, escalamento etc.). Para a definição geométrica dos objetos, pode-se construir tabelas de objetos, faces, arestas e vértices (ver Quadro 1):

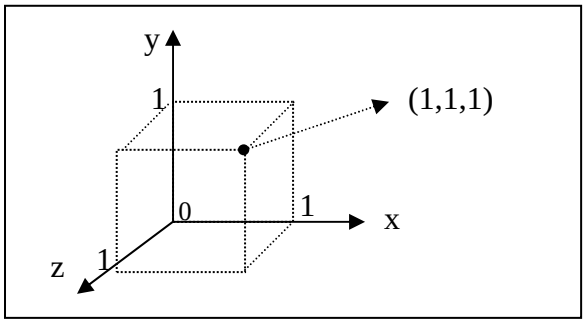
Quadro 1 - Definição geométrica dos objetos.

```
    OBJETOS (no, xsru, ysru, zsru, dimx, dimy, dimz, cor,
estilo de linha, ...)

    FACES (nf, a1, a2, a3, ...);

    ARESTAS (na, v1, v2, ...);

    VÉRTICES (nv, v1, v2, ...).
```

2.2.2 SISTEMA DE REFERÊNCIA DO UNIVERSO (SRU)

É utilizado para definir as coordenadas dos vértices dos objetos de uma cena tridimensional, em relação a um ponto qualquer do universo da cena (Figura 2).

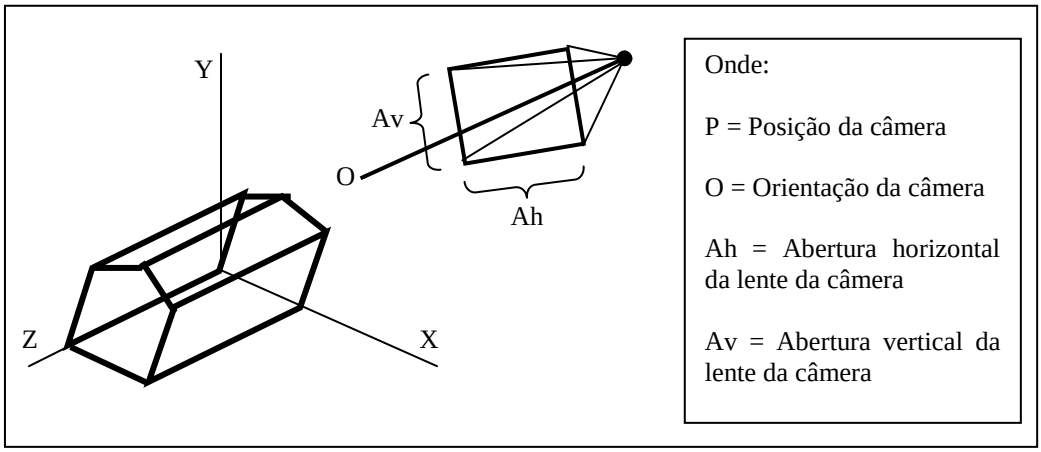
Figura 2 - Sistema de Referência Universo.



2.2.3 SISTEMA DE REFERÊNCIA DA CÂMERA (SRC)

É utilizado para definir as coordenadas dos vértices dos objetos de uma cena tridimensional, em relação ao Centro de Projeção (CP) da Pirâmide de Visualização (Figura 3). É a partir do CP, que são projetados raios visuais até cada um dos vértices dos objetos contidos na cena. As intersecções destes raios visuais com o plano de projeção, vão originar a projeção perspectiva que deverá ser exibida em um dispositivo de visualização bidimensional.

Figura 3 - Sistema de Referência da Câmera.



Fonte: (NEWMAN, 1979, p. 300).

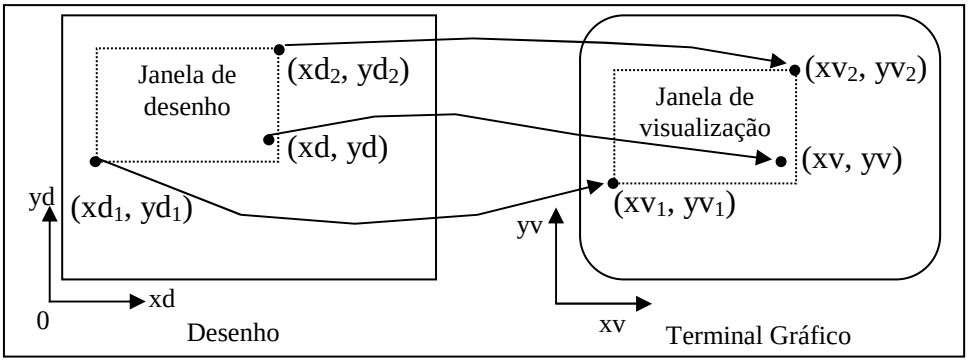
2.2.4 SISTEMA DE REFERÊNCIA DE PROJEÇÃO (SRP)

É utilizado para definir as coordenadas dos vértices dos objetos de uma cena tridimensional, projetados em um plano de projeção bidimensional. É responsável pela formação da projeção perspectiva. Para facilitar os cálculos, o tamanho do plano de projeção deve ser unitário, em relação aos eixos do SRP.

2.2.5 SISTEMA DE REFERÊNCIA DE DISPOSITIVO (SRD)

É utilizado para transferência da imagem bidimensional formada no SRP, para um dispositivo de visualização bidimensional (monitor, impressora, *plotter* etc.)(Figura 4).

Figura 4 - Sistema de Referência de Dispositivo.



Fonte: (TORI, 1987, p.88).

2.3 OUTRAS PROPRIEDADES

Outras propriedades importantes de uma câmera sintética são:

- Afastamento Angular (AF): é o ângulo horizontal da rotação da câmera em relação à origem do SRU.
- Altura Angular (AL): é o ângulo vertical da rotação da câmera em relação à origem do SRU.
- Ângulo de Visualização (AV): corresponde à abertura da lente da câmera sintética. Podemos fixar este ângulo em 80 graus. Se este ângulo for reduzido, obtém-se o efeito *Zoom In*, se for incrementado obtém-se *Zoom Out*. Portanto, o AV pode ser utilizado para o controle de *Zoom* da câmera.

3 OPENGL

3.1 O QUE É OpenGL

É uma biblioteca gráfica de modelagem e exibição tridimensional, bastante rápida e portável para vários sistemas operacionais. Seus recursos permitem ao usuário criar objetos gráficos com qualidade próxima à de um *raytracer*, de modo mais rápido que este último, além de incluir recursos avançados de animação, tratamento de imagens e texturas (BRITO, 2001, p. 1).

A biblioteca OpenGL (*Open Graphics Library*) foi introduzida em 1992 pela Silicon Graphics, no intuito de conceber uma API (Interface de Programação de Aplicação) gráfica independente de dispositivos de exibição. Com isto, seria estabelecida uma ponte entre o processo de modelagem geométrica de objetos, situadas em um nível de abstração mais elevado, e as rotinas de exibição e de processamento de imagens implementadas em dispositivos (hardware) e sistemas operacionais específicos. A função utilizada pelo OpenGL para desenhar um ponto na tela, por exemplo, possui os mesmos nomes e parâmetros em todos os sistemas operacionais nos quais OpenGL foi implementada, e produz o mesmo efeito de exibição em cada um destes sistemas (BRITO, 2001, p. 1).

Diante das funcionalidades providas pelo OpenGL, tal biblioteca tem se tornado um padrão amplamente adotado na indústria de desenvolvimento de aplicações. Este fato tem sido encorajado também pela facilidade de aprendizado, pela estabilidade das rotinas, pela boa documentação disponível e pelos resultados visuais consistentes para qualquer sistema de exibição concordante com este padrão (BRITO, 2001, p. 1).

Diversos jogos, aplicações científicas e comerciais têm utilizado OpenGL como ferramenta de apresentação de recursos visuais, principalmente com a adoção deste padrão por parte dos fabricantes de placas de vídeo destinadas aos consumidores domésticos (BRITO, 2001, p. 1).

3.2 RECURSOS GRÁFICOS

Entre os recursos gráficos disponíveis pelo OpenGL, podem ser destacados os seguintes (BRITO, 2001, p. 1):

a) Modos de desenho de pontos (escrita/xor/transparência);

b) Ajuste de largura de linhas;

c) Aplicação de transparência;

d) Ativação/desativação de serrilhamento (*aliasing*);

e) Mapeamento de superfícies com textura;

f) Seleção de janela de desenho;

g) Manipulação de fontes/tipos de iluminação e sombreamento;

h) Transformação de sistemas de coordenadas;

i) Transformações em perspectiva;

j) Combinação de imagens (*blending*);

k) Entre outras.

3.3 VANTAGENS VOLTADAS AOS DESENVOLVEDORES

Existem muitos itens que devem ser tratados na escolha de uma biblioteca gráfica, tais como ser padrão de indústria, estabilidade, fidedignidade, portabilidade, escalabilidade e facilidade no uso.

3.3.1 PADRÃO DE INDÚSTRIA

Um consórcio independente, a Comissão de Diretores de Pesquisa da Arquitetura OpenGL, guia a especificação do OpenGL. Com amplo apoio da indústria, OpenGL é o único padrão gráfico multiplataforma verdadeiramente aberto, independente de fornecedores (OPENGL, 2001, p. 1).

3.3.2 ESTABILIDADE

Implementações com OpenGL têm estado disponíveis há mais de sete anos em uma grande variedade de plataformas. Inclusões em sua especificação são bem controladas, e atualizações propostas são anunciadas a tempo para que os desenvolvedores adotem as mudanças. Exigências de compatibilidade asseguram as aplicações existentes a não ficarem obsoletas (OPENGL, 2001, p. 1).

3.3.3 FIDEDIGNIDADE E PORTABILIDADE

Todas as aplicações OpenGL produzem exibições visuais consistentes em qualquer OpenGL API, apesar do sistema operacional ou da compatibilidade de hardware (OPENGL, 2001, p. 1).

3.3.4 ESCALABILIDADE

Aplicações baseadas na API OpenGL podem ser executadas em sistemas que variam de produtos eletrônicos a PCs, estações de trabalho e supercomputadores. Como resultado, aplicações podem escalar para qualquer classe de máquina que o desenvolvedor deseja atingir (OPENGL, 2001, p. 1).

3.3.5 FACILIDADE DE USO

OpenGL é bem estruturado, com um desenho intuitivo e comandos lógicos. Rotinas eficientes de OpenGL tipicamente resultam em aplicações com menos linhas de código que as aplicações geradas utilizando outros pacotes ou bibliotecas gráficas. Além disso, os *drivers* OpenGL encapsulam informação sobre o hardware subjacente, livrando o desenvolvedor da aplicação de ter que projetá-la para características de hardware específicas (OPENGL, 2001, p. 1).

3.4 O CANAL DE PROCESSAMENTO DOS DADOS NO OpenGL

As rotinas OpenGL simplificam o desenvolvimento de software gráfico — de renderização de um simples ponto geométrico, linha, ou polígono preenchido para a mais complexa superfície curva, iluminada e texturizada. OpenGL dá ao desenvolvedor de software acesso a primitivas geométricas e de imagem, listas de exibição, transformações de modelagem, iluminação, anti-aliasing, mistura de cores, e muitas outras características.

O padrão bem especificado de OpenGL tem ligações de linguagens para C, C++, Fortran, Ada e Java.

Todas as implementações licenciadas de OpenGL vêm de uma única especificação e documentação de ligação de linguagem são exigidos para passar um conjunto de testes de conformidade.

As aplicações que utilizam funções OpenGL são facilmente portáveis entre a ampla gama de plataformas para maximizar a produtividade do programador e reduzir o tempo para que o produto alcance o mercado.

Todos os elementos do estado OpenGL - até mesmo os conteúdos da memória de textura e do *buffer* de tela podem ser obtidos por uma aplicação de OpenGL. OpenGL também suporta aplicações de visualização com imagens 2D tratadas como tipos de primitivas que podem ser manipuladas exatamente como objetos geométricos 3D.

OpenGL executa em todos os principais sistemas operacionais, incluindo Mac OS, OS/2, UNIX, Windows 95/98, Windows 2000, Windows NT, Linux, OPENStep e BeOS; também trabalha na maioria dos sistemas de janelas, incluindo Win32, MacOS, Gerente de Apresentação, e Sistema X-Window. OpenGL é acessível as linguagens Ada, C, C++, Fortran, Python, Perl e Java e oferece total independência de protocolos e de topologias de rede (OPENGL, 2001, p. 1).

3.5 INOVAÇÃO CONTÍNUA E APLICAÇÕES

O padrão OpenGL está em constante evolução. Revisões formais ocorrem em intervalos periódicos, e as extensões permitem aos desenvolvedores de aplicações acessar os últimos avanços de hardware através do OpenGL. Tão logo as extensões sejam amplamente aceitas, elas são consideradas nas inclusões do núcleo padrão do OpenGL. Este processo permite ao OpenGL estar envolvido em um controle de modo a ser ainda mais inovativo.

OpenGL é o penetrante padrão para consumidores 3D e aplicações profissionais através da maioria de plataformas de sistemas operacionais, conforme segue:

- Aplicações Windows;
- Animação 3D, modelagem e renderização;
- CAD / CAM / Prototipação Digital;
- Kits de ferramentas e bibliotecas para desenvolvedores;
- Jogos;
- Criação e Visualização VRML;
- Utilitários: Protetores de Tela, Conversores de Formatos, Medidores, etc;
- Simulação e Visualização;
- Análise de Dados Científicos e Mapas Geográficos;

- Aplicações LINUX: todas as categorias;
- Aplicações Mac: todas as categorias.

3.6 FUNÇÕES GRÁFICAS

Como funções gráficas, o OpenGl apresenta os seguintes itens:

- *Buffer* de Acumulação: Um *buffer* no qual múltiplos quadros renderizados podem ser compostos para produzir uma única imagem misturada. Usado para efeitos como profundidade do campo, movimento obscuro, e *anti-aliasing* de cena;

- Mistura de Alfa: Provê meios para criar objetos transparentes. Usando Informação de alfa, um objeto pode ser definido com qualquer nível de transparência, desde totalmente transparente até totalmente opaco;

- *Anti-aliasing*: Um método de renderização usado para alisar linhas e curvas. Esta técnica calcula a média da cor dos pixels adjacentes para a linha. Tem o efeito visual de abrandamento da transição dos pixels na linha e dos adjacentes à linha, provendo assim uma aparência mais lisa;

- Modo de Índice de Cor: *Buffers* de cores armazenam índices de cor particularmente como vermelho, verde, azul, e componentes alfa de cor;

- Lista de Exibição: Uma lista nomeada de comandos OpenGL. Os conteúdos de uma lista de exibição podem ser pré-processados e poderiam executar mais eficientemente que o mesmo conjunto de comandos OpenGL executados em modo imediato;

- Bufferização Dobrada: Usado para prover animação lisa de objetos. Cada cena sucessiva de um objeto em movimento pode ser construída na parte de trás ou "*buffer* escondido" e então pode ser exibido. Isto permite só exibir sempre imagens completas na tela;

- Realimentação: Um modo onde OpenGL devolverá para a informação geométrica processada (cores, posição do pixel, e assim por diante) para a aplicação como comparado à renderização do mesmo dentro do *buffer* de tela;

- Sombreamento Gouraud: Lisa interpolação de cores em um polígono ou segmento de linha. Cores são nomeadas a vértices e linearmente são interpoladas pela primitiva para produzir uma variação relativamente lisa da cor;

- Modo Imediato: Execução de comandos OpenGL quando eles são chamados, em lugar de uma lista de exibição;

- Iluminação de Materiais: A habilidade para computar a cor de qualquer ponto com precisão dada as propriedades materiais da superfície;

- Operações de Pixel: Armazenamento, transformação, traçando, ampliação ou redução;

- Avaliador Polinomial: Para suportar B-splines racionais e não uniformes (NURBS - *Non Uniform Rational B-Spline*);

- Primitivas: Um ponto, linha, polígono, mapa de bits, ou imagem;

- Primitivas Raster: Mapas de bits e matrizes de pixel;

- Modo RGBA: *Buffers* de cores armazenam vermelho, verde, azul, e componentes de cor alfa, no lugar de índices;

- Seleção e Escolha: Um modo no qual OpenGL determina se são feitas certas primitivas gráficas identificadas pelo usuário em uma região de interesse no *buffer* da tela;

- Cópias da Superfície: Um *buffer* que pode ser usado para mascarar pixels individuais no *buffer* de cores da tela;

- Mapeamento da Textura: O processo de aplicar uma imagem em uma primitiva gráfica. Esta técnica é usada para gerar realismo em imagens. Por exemplo, um desenho como uma superfície de uma mesa retangular pode ter uma textura de madeira aplicada a ela para fazê-la parecer mais realista;

- Transformação: A habilidade para mudar a rotação, tamanho, e perspectiva de um objeto em um espaço de coordenada 3D;

- Tamponamento Z: O tamponamento Z é usado para ver se uma parte de um objeto está mais perto do observador do que outra. É importante na remoção de superfícies escondidas.

4 MODELAGEM TRIDIMENSIONAL

Existem várias técnicas de modelagem de objetos tridimensionais, cada técnica apresenta vantagens e desvantagens quando são considerados aspectos como espaço requerido para armazenamento dos modelos, precisão da representação, facilidade de modelagem etc. A escolha de qualquer uma destas técnicas está diretamente relacionada ao tipo de recurso que uma aplicação possa necessitar (VIEIRA, 1994, p.14).

Tal como nas transformações bidimensionais, são adotados dois pontos de vista complementares: o objeto (ou a imagem) é manipulado diretamente através da utilização de transformações geométricas ou o objeto permanece fixo e o sistema de coordenadas do observador é deslocado pela utilização de transformações de coordenadas. Alem disso, a construção de objetos (ou imagens) complexos é facilitada pela utilização de transformações ou instanciação as quais combinam ambos pontos de vista (PLASTOCK, 1986, p.157).

As principais técnicas para modelagem tridimensional são:

- Representação por fronteira;
- Geometria sólido-construtiva;
- Instanciação de primitivas;
- Particionamento espacial e
- *Sweeping*.

4.1 REPRESENTAÇÃO POR FRONTEIRA

Segundo TORI (1987, p.265), a descrição mais natural de um objeto é por meio da definição de seus primitivos geométricos: os pontos (vértices), linhas (arestas) e planos (faces). Estas entidades agregam-se arcos e superfícies curvas.

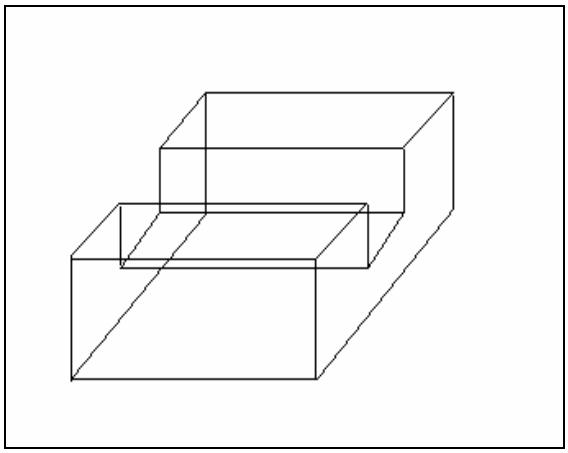
Para se definir uma estrutura tridimensional, não basta uma simples lista de vértices: é preciso definir quais dos vértices se unem, formando as superfícies do sólido. Surge, portanto, o conceito de topologia ao lado de geometria.

Por geometria do objeto entende-se o conjunto das informações acerca de suas dimensões e localizações. Assim um objeto é definido geometricamente em termos de seus pontos, linhas e planos. Se o objeto possui superfícies curvas, estas podem ser aproximadas convenientemente por uma série de polígonos planos. Esta técnica é freqüentemente utilizada

nos estudo do comportamento físico do objeto: por exemplo, no cálculo das deformações de uma estrutura.

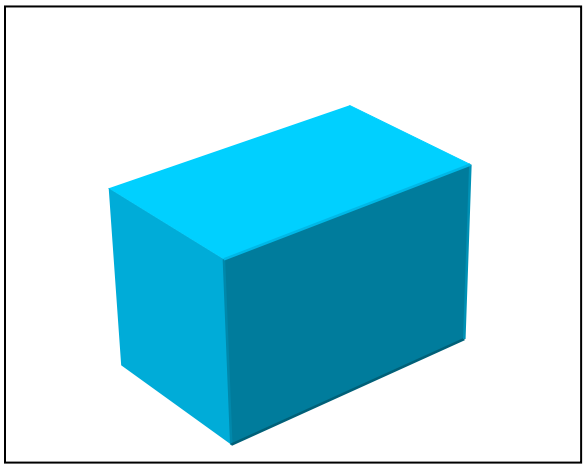
Normalmente não é necessário guardar informações sobre todas as faces e arestas do sólido, à medida que da lista de vértices se extraem as faces e as arestas, e vice-versa. Qual das classes de entidades armazenar é uma decisão dependente da aplicação. Em algumas fazes é interessante que se trabalhe sobre a representação *wire-frame* (Figura 5), a descrição em torno de vértices ou arestas permite o acesso mais rápido ao modelo do sólido.

Figura 5 – Representação *wire-frame* de um sólido.



Por outro lado, quando se trata de avaliar a intersecção entre sólidos, ou, ainda, de mostrar o sólido com realismo, a definição de superfícies é essencial (Figura 6).

Figura 6 – Sólido com definição de superfície.



Para que um conjunto de superfícies delimite um sólido real, as superfícies devem obedecer a algumas condições: devem ser limitadas (isto é, devem ocupar um espaço finito) e conectadas (isto é, existir sempre um caminho sobre as superfícies que interligue quaisquer dois de seus pontos) (TORI, 1987, p.265).

A representação por fronteiras tem como vantagens o grande poder de expressão, nela o espaço para modelagem depende exclusivamente da primitiva selecionada para uso, podendo esta técnica representar modelos de um espaço mais abrangente do que os representados por CSG[2] (VIEIRA, 1994, p.15).

Desvantagens que podem ser apontadas são:

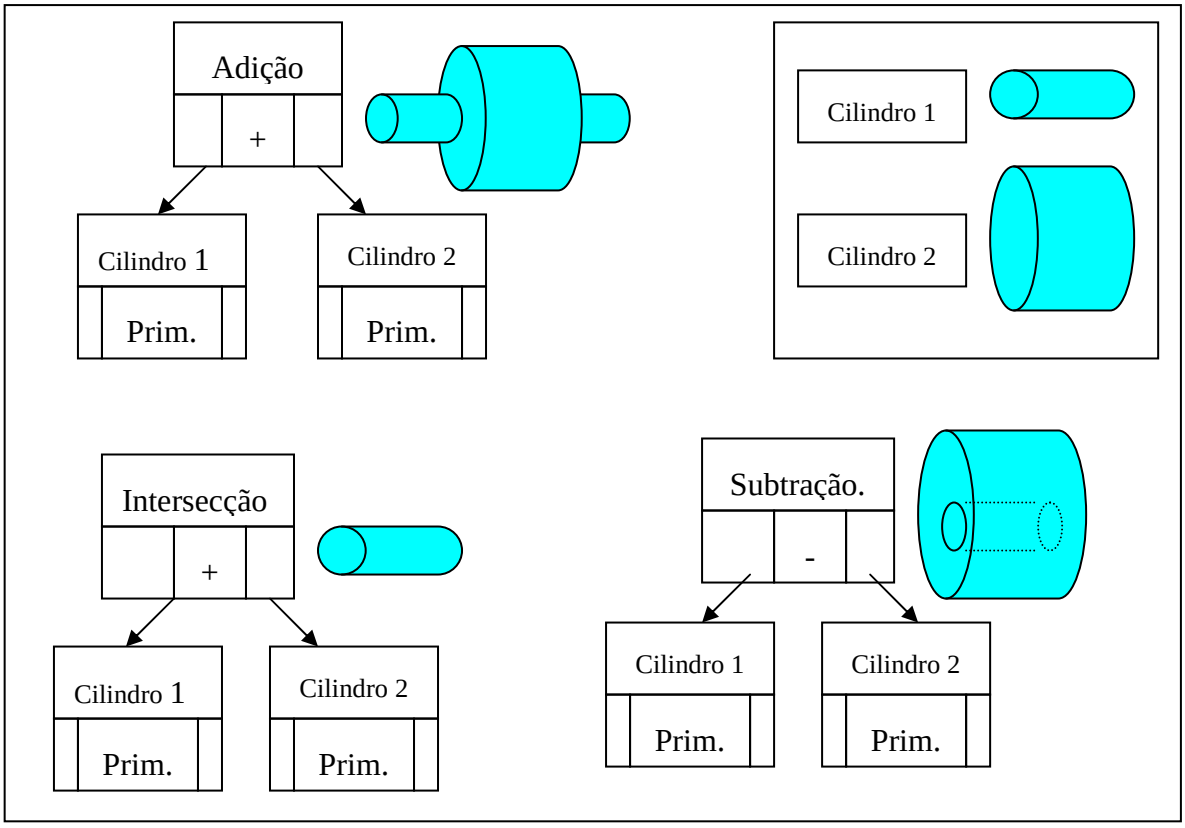
- Neste modelo, o endereçamento espacial é complicado, necessitando de algoritmos complexos para operações simples, como por exemplo, a localização de um ponto no espaço.
- Necessidade de dois tipos de validação, topológica e geométrica;
- Existem modelos representados por meio de CSG que não podem ser representados por fronteiras;
- Compõem-se basicamente de vértices e arestas, tornando difícil a descrição dos modelos.

4.2 GEOMETRIA SÓLIDO-CONSTRUTIVA

Como a representação de sólidos com base nas superfícies limitantes cria constantemente objetos absurdos e construções ambíguas, além de requerer o armazenamento de um excesso de informações, pode-se utilizar um novo tipo de representação geométrica, com base em primitivos e operadores. Assim cada objeto é descrito por uma árvore onde as folhas são os sólidos primitivos (paralelepípedos, cones, cilindros, esferas, etc.), e os nós são os operadores de conjunto (adição, subtração, intersecção). Desta forma, a cada nó tem-se um sólido completo e válido, como pode ser visto na figura 7 (TORI, 1987, p.267).

[2] CSG é abreviatura de *Constructive Solid Geometry* (Geometria sólido-construtiva).

Figura 7 - Operadores na geometria construtiva.



Fonte: TORI, 1987, p.267.

Além da validade natural dos objetos, o modelo de sólidos componentes se converte facilmente no modelo de superfícies limitantes, permitindo que se usem os algoritmos desenvolvidos para este último modelo. Cada nó da árvore de composição[3] do sólido, na geometria construtiva, tem a seguinte estrutura (TORI, 1987, p.268):

1. Operação (união, diferença ou interseção);
2. Ponteiro ao nó da direita;
3. Ponteiro ao nó da esquerda;
4. Descrição do sistema de coordenadas locais, em relação ao global.

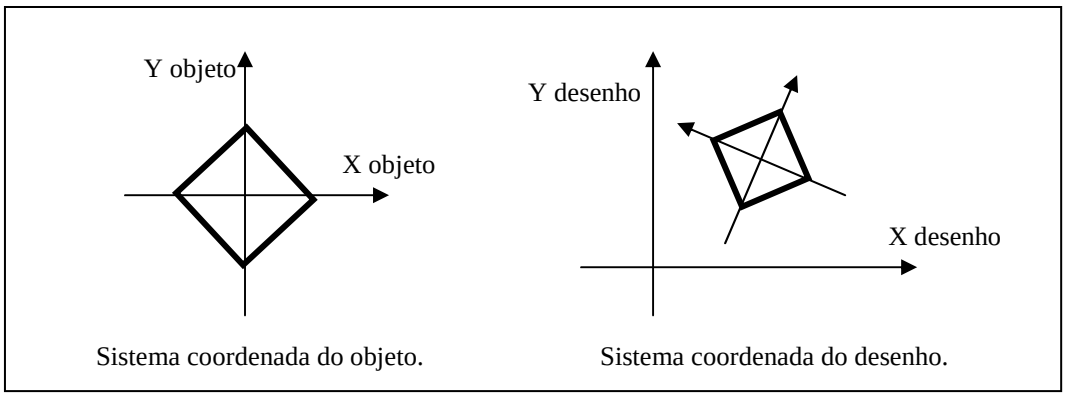
4.3 INSTANCIAÇÃO DE PRIMITIVAS

Freqüentemente, uma imagem (ou um desenho) é composta por muitos objetos que se repetem. Por sua vez, estes objetos podem ainda ser compostos por outros símbolos e outros objetos. Supondo que cada objeto é definido, independentemente do desenho, no próprio

[3] Conjunto de nós contendo informações para geração de um objeto a partir de outros dois.

espaço coordenado, designado por espaço coordenado do objeto. Querendo juntar estes objetos para formar um desenho ou pelo menos parte de um desenho, também designado por sub-desenho (Figura 8). Isto poderá ser obtido através da transformação de coordenadas, designada por transformação de instanciação, a qual converte as coordenadas do objeto em coordenadas do desenho, assim como coloca ou cria uma instancia (imagem) do objeto no sistema coordenada do desenho (PLASTOCK, 1986, p.105).

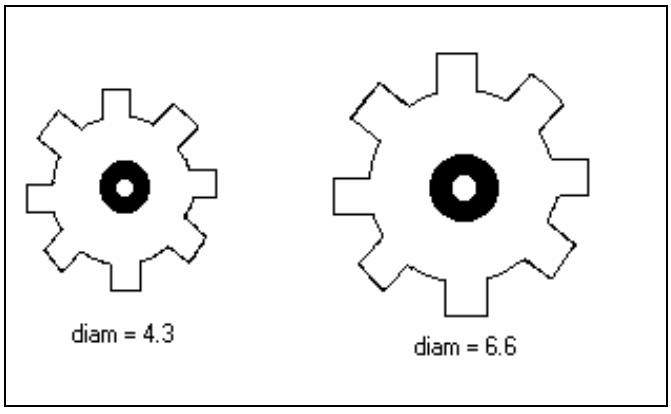
Figura 8 - Objeto instanciado.



Fonte: PLASTOCK, 1986, p.105.

Com a utilização de diferentes transformações de instanciação, o mesmo objeto pode ser colocado em diferentes posições, com diferentes tamanhos e orientações dentro de um sub-desenho (Figura 9).

Figura 9 - Engrenagens definidas por seu diâmetro.



Cada primitiva gráfica de saída é caracterizada por uma geometria e por sua aparência na superfície de exibição do dispositivo de saída. Seu aspecto é controlado por um conjunto

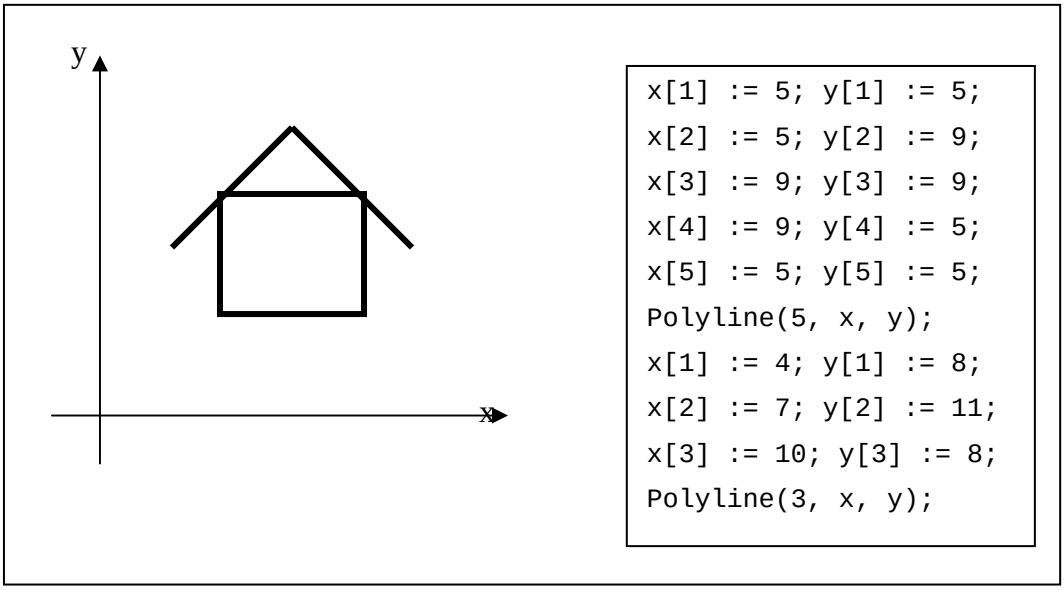
de atributos: um conjunto de propriedades que determinam a aparência da primitiva no dispositivo. Exemplos de atributos de primitivas são: a cor de um segmento de reta, o tamanho de um caractere e as coordenadas dos vértices de um polígono (PERCIANO, 1989, p. 88).

Um exemplo simples de primitiva, é o `Polyline,` uma primitiva do GKS[4], ela gera uma poligonal unindo uma seqüência dada de pontos do plano por segmentos de reta. A poligonal é traçada em resposta a invocação da função `Polyline` cujos parâmetros são os atributos geométricos da primitiva, ou seja, a seqüência de pontos:

```
Procedure Polyline (n: integer; x,y: coordenadas);
```

onde `n` é o número de pontos da seqüência e `x` e `y` são arranjos contendo as coordenadas reais de cada ponto da seqüência. A figura 10 ilustra a utilização desta primitiva na preparação de um desenho bastante simples.

Figura 10. A primitiva Polyline.



Fonte: PERCIANO, 1989, p. 89.

[4] GKS é abreviatura de *Graphical Kernel System* (Núcleo de Sistema Gráfico), um pacote gráfico cuja especialização foi aprovada como padrão internacional pela ISO (*International Standard Organization)* em 1984 e por diversas outras entidades nacionais de padronização.

A `Polyline` é a primitiva básica do GKS para o traçado de linha. Como a seqüência de pontos da `Polyline` é integrada por segmentos de reta, o traçado de curvas, tais como arcos de circunferência ou de elipses, pode ser obtido aproximando-as por uma `Polyline` formada por uma série de pequenos segmentos (PERCIANO, 1989, p. 89).

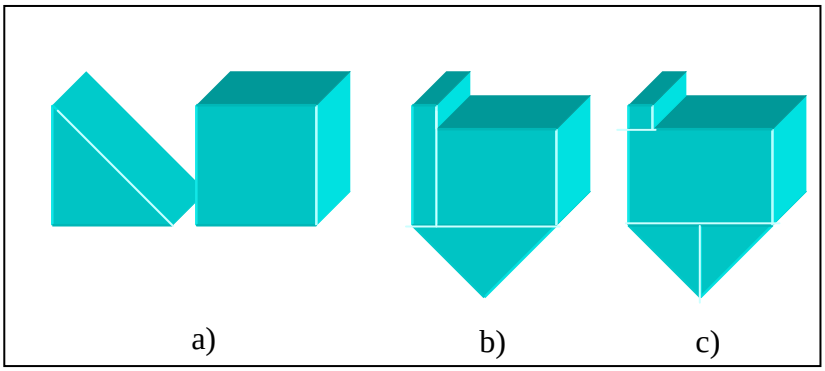
4.4 PARTICIONAMENTO ESPACIAL

Nesta técnica, um sólido é decomposto em uma coleção de sólidos mais simples, embora não necessariamente do mesmo tipo do modelo original. As primitivas podem variar em tipo, tamanho, posição, parametrização e orientação (FOLEY, 1990, p.548).

Três processos que se destacam são, decomposição celular, enumeração por ocupação espacial e *Octrees*:

a) Decomposição celular, um mesmo objeto pode ser decomposto de várias formas (Figura 11);

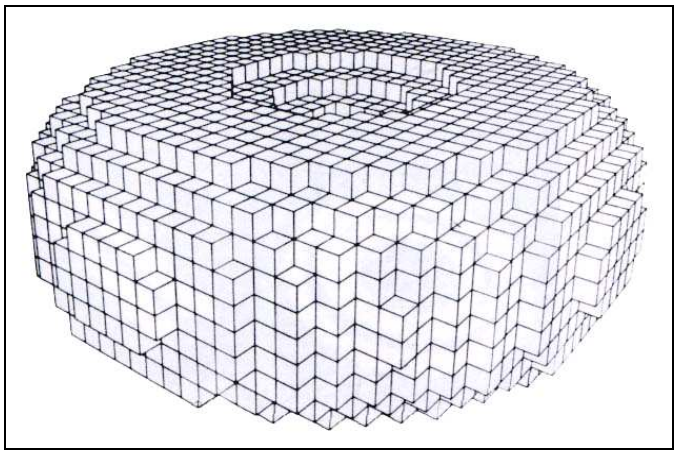
Figura 11 - As células em (a) podem se transformar para construir o mesmo objeto (b) e (c).



Fonte: FOLEY, 1990, p. 549

b) Enumeração por ocupação espacial, neste processo o sólido é decomposto em células idênticas (cubos) organizadas em uma grade regular fixa. O tamanho da célula determina a resolução e o espaço para o armazenamento do objeto (Figura 12);

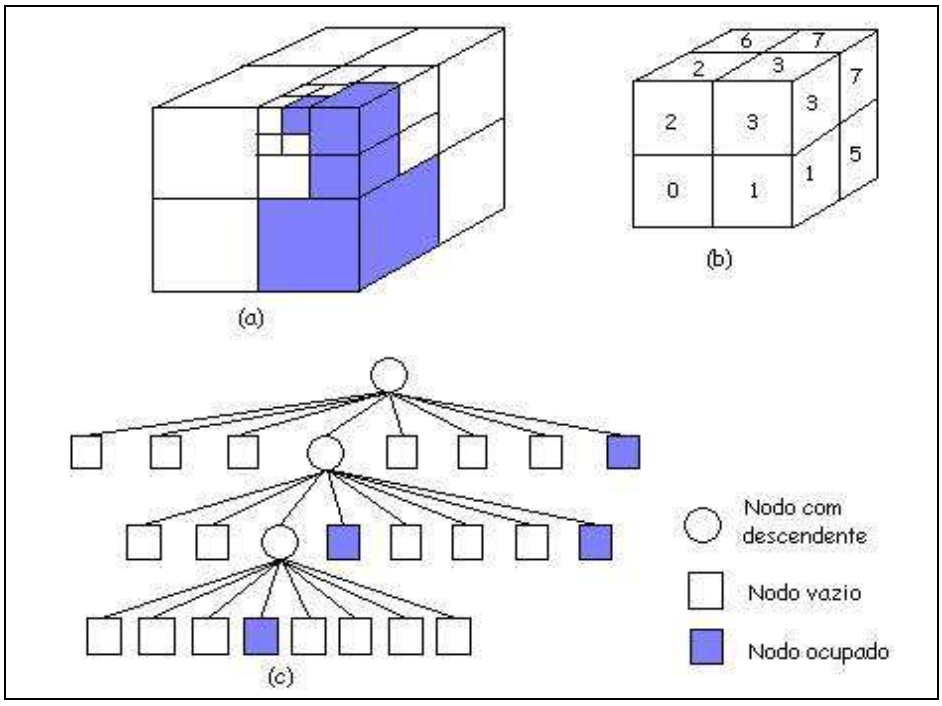
Figura 12 - Representação por ocupação espacial.



Fonte: FOLEY, 1994, p. 548.

c) *Octrees*, é uma variação da enumeração por ocupação espacial que reduz o espaço de armazenamento, consiste na formação de uma estrutura em árvore, onde, nos nodos internos encontram-se blocos parcialmente ocupados e nas folhas estão os blocos totalmente vazios ou preenchidos (MORTENSON, 1985, p.452) (Figura 13).

Figura 13 - Árvore formada na representação por *Octree*.



Fonte: MORTENSON, 1985, p.453.

4.5 SWEEPING

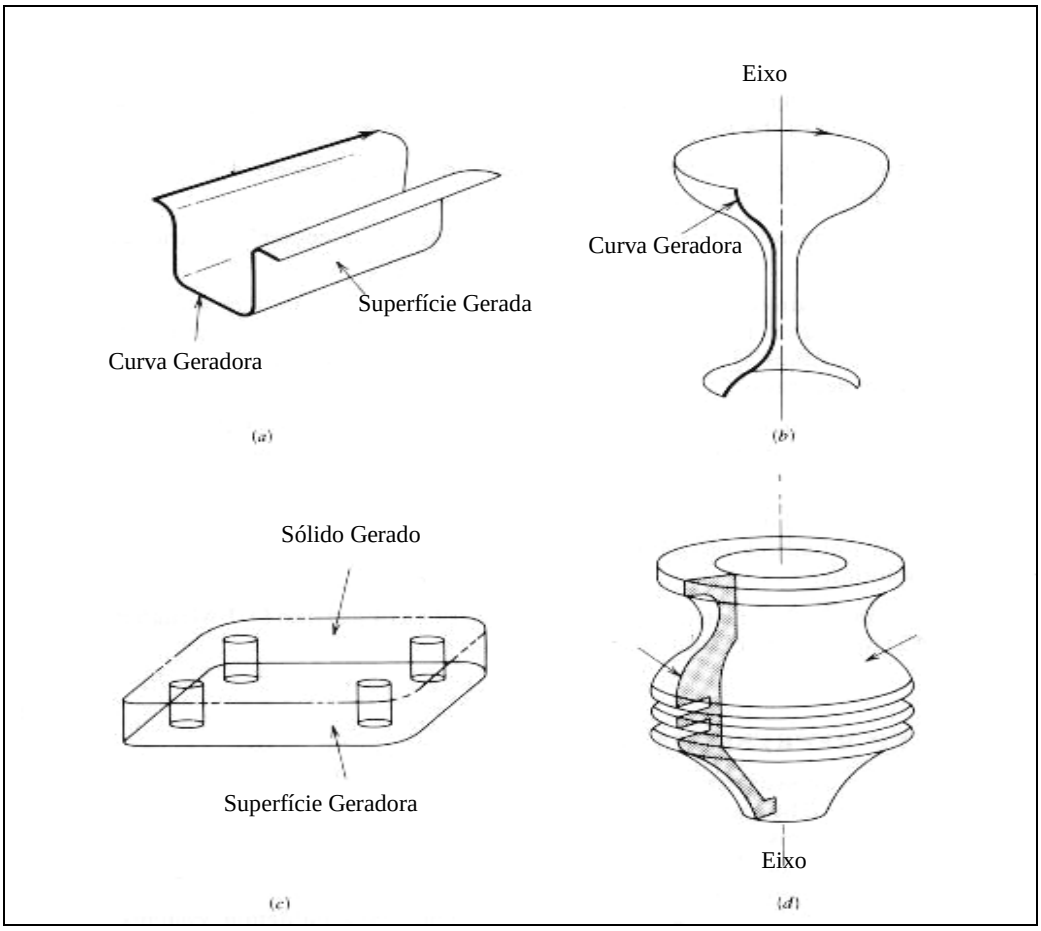
Representações *sweep* são baseadas na noção de mover um ponto, curva ou superfície ao longo de um caminho. O conjunto de pontos gerados por este processo define um objeto uni-, bi-, ou tri-dimensional. Representações *sweep* para modelagem de sólidos são simples de compreender e ainda oferecem um vasto campo para desenvolvimento de novos métodos. Este tipo de representação é muito utilizado nos sistemas de modelagem contemporâneos e tem se mostrado prático e eficiente para a modelagem de peças mecânicas de seções constantes (simetria radial) (MORTENSON, 1985, p. 455).

Sweeping é um caminho fácil para desenhistas criarem formas tridimensionais. Este processo requer que o desenhista faça um perfil ponto-a-ponto do elemento desejado. Quando ele é terminado, o computador mostra o elemento como um plano achatado tridimensional. A espessura é adicionada pelo "*sweep*" do perfil de uma posição para outra (LANSDOWN, 1989, p. 209).

Para modelagem de sólidos, duas coisas são requeridas – um objeto a ser movido e uma trajetória para movê-lo. O objeto pode ser uma curva, superfície ou sólido, e a trajetória é um caminho definível analiticamente. Ocasionalmente o termo gerador é usado para denominar o objeto, e o termo diretor é usado para denominar a trajetória.

Na figura 14 são descritos dois exemplos de cada um dos dois principais tipos de trajetória de *sweeping*: translação e rotação. Nota-se que a trajetória curva não é um elemento necessário no objeto de *sweep* (MORTENSON, 1985, p. 454).

Figura 14 - Exemplos de representações de *sweep*

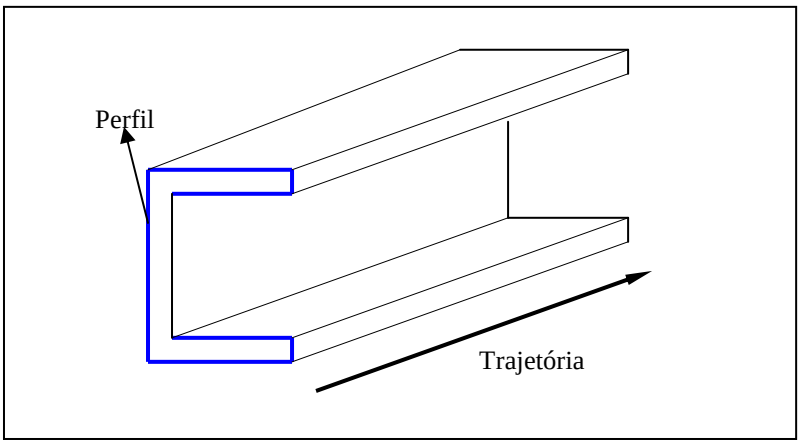


Fonte: MORTENSON, 1985, p. 456

4.5.1 *SWEEPING* TRANSLACIONAL

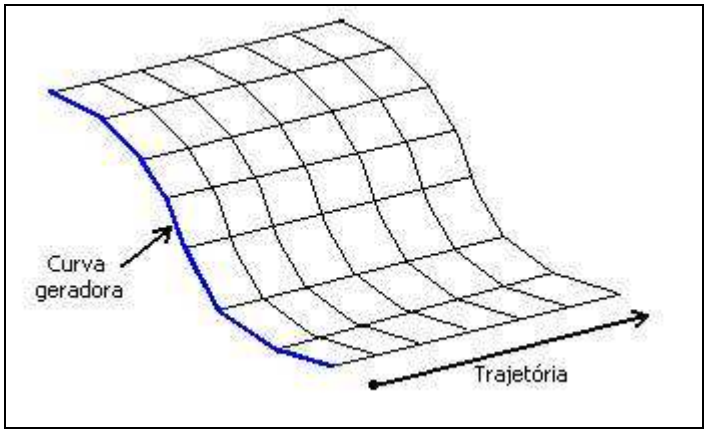
A trajetória pela qual o perfil é deslocado é descrita por equações, as quais definirão o aspecto do objeto a ser gerado. Se, por exemplo, forem aplicadas equações que descrevam uma trajetória reta (Figura 15), então, o objeto resultante terá aspecto de uma barra linear e de secção transversal igual ao perfil (VIEIRA, 1994, p.24).

Figura 15 - Objeto gerado por *sweeping* translacional.



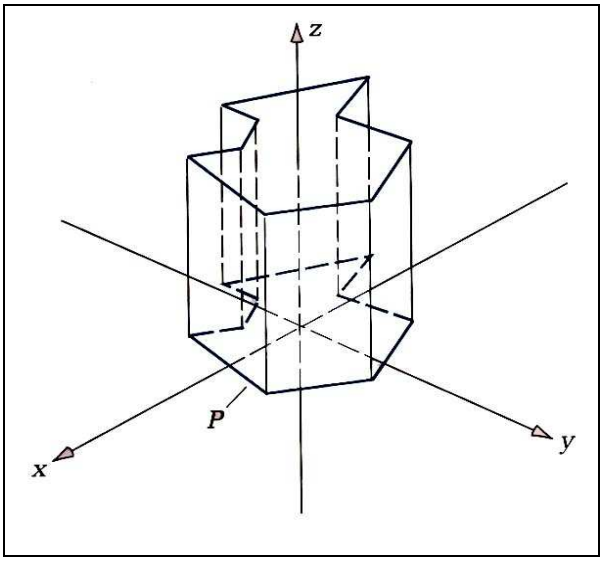
O *sweeping* translacional de uma curva cria uma superfície com duas margens bidimensionais ligadas (Figura 16). Essas regiões são conectadas por uma estrutura unidimensional (MORTENSON, 1985, p. 456).

Figura 16 - Superfície gerada por *sweeping* translacional.



Um poliedro como um prisma pode ser gerado através de *sweeping*, com sua base inicialmente situada no plano `x,y,` ao ser movida ao longo do eixo `z` até uma determinada distância (Figura 17). Os lados são criados por cada um dos pontos ao longo do intervalo movido (HILL, 1990, p. 315).

Figura 17 - Modelagem de um prisma.

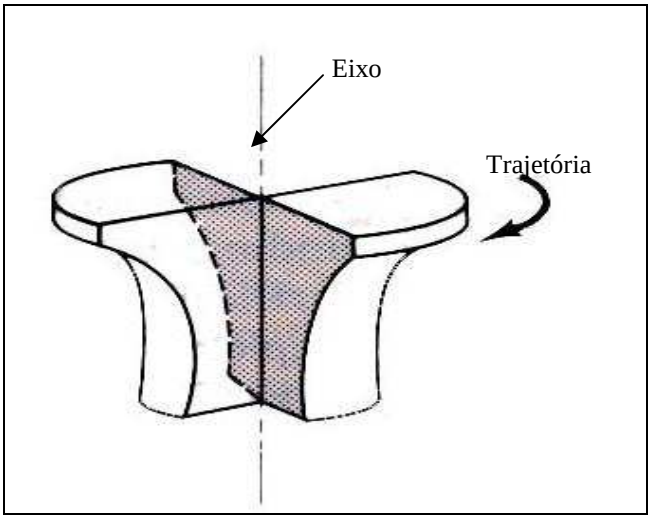


Fonte: HILL, 1990, p. 316.

4.5.2 *SWEEPING* ROTACIONAL

A trajetória de um *sweeping* rotacional é gerada por um algoritmo que move cada ponto na trajetória ao longo um arco circular em um plano através do ponto perpendicular ao eixo de rotação e com um raio definido como a distância perpendicular do ponto ao eixo (Figura 18) (MORTENSON, 1985, p. 456).

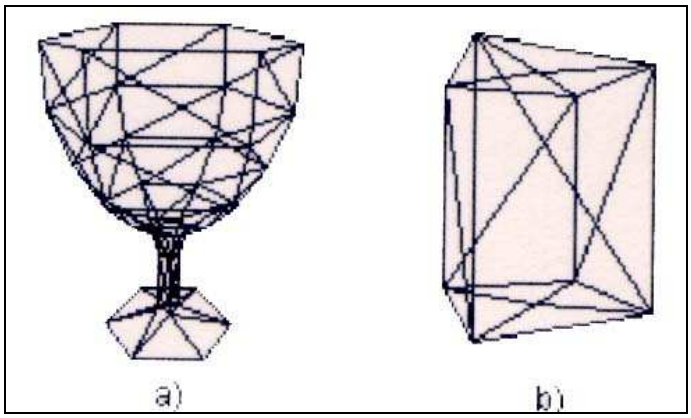
Figura 18 - Exemplo de *sweeping* rotacional.



Fonte: MORTENSON, 1985, p. 457.

Compondo, por exemplo, o perfil de uma taça a partir do traçado de alguns segmentos e rotacionando-os 360° em torno de um eixo, obtém-se a representação tridimensional da taça (Figura 19). A manipulação de uma variável para o controle do número de passos interfere diretamente no aspecto do objeto obtido, ou seja, desejando-se uma taça com seção hexagonal, define-se o número de passos igual a 6 (Figura 16-a). Até mesmo um cubo pode ser obtido por rotação, quando é utilizado o modelo de arestas para sua construção, basta amostrar pontos em incrementos de 90° (Figura 16-b).

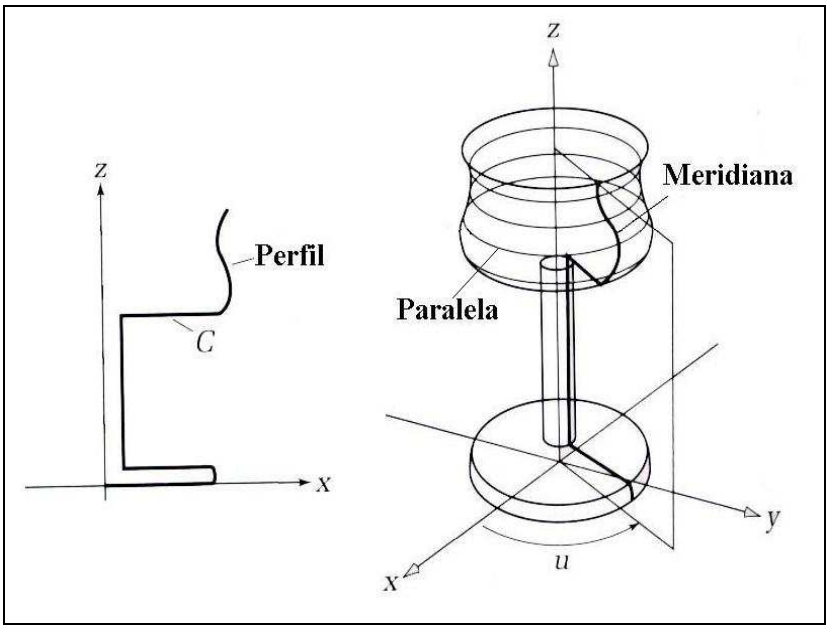
Figura 19 - *Sweeping* rotacional utilizando modelo de arestas.



Fonte: VIEIRA, 1994, p. 24.

A figura 20 mostra um exemplo em que C esta situado no plano x,y é rotacionado em torno do eixo z. O contorno da superfície C é chamado repetidas vezes, e é dado parametricamente por $c(v)=(x(v),y(v))$ à medida que v é modificada no decorrer de uma linha $(v_{inicial},v_{final})$ (HILL, 1990, p. 294).

Figura 20 - Perfil em evolução.



Fonte: HILL, 1990, p. 294.

Para a superfície em rotação, cada ponto `(x(v),z(v))` do perfil é arrastado em torno do eixo controlado pelo parâmetro `u`, com `u` especificando o ângulo que cada ponto tem ao ser rotacionado em torno do eixo. As diferentes posições da curva `c` em torno do eixo são chamadas meridianas. O *sweeping* é completado em torno do eixo, gerando um círculo completo, e então os contornos da constante tornam-se círculos. Chamadas paralelas da superfície. Desta forma os pontos da superfície no vetor são dados pela equação (HILL, 1990, p. 294).

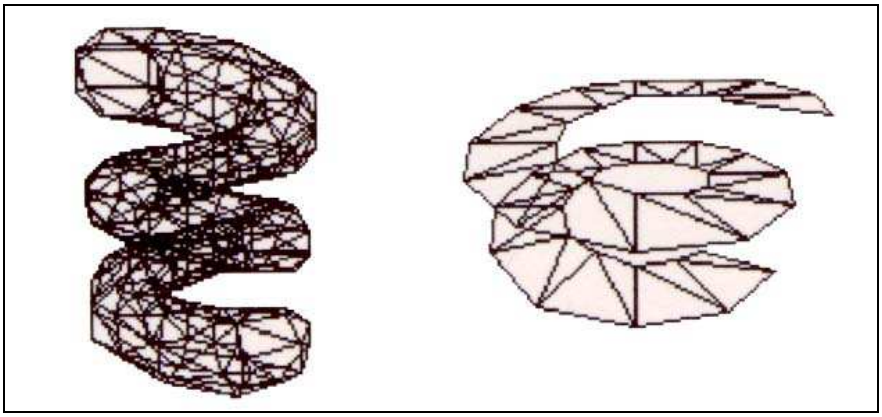
```
p(u,v) = (x(v)cos(u), x(v) sen(u), z(v))
```

4.5.3 *SWEEPING* HELICOIDAL

A combinação das duas técnicas anteriores, sweeping translacional e sweeping rotacional, resulta na descrição de um modelo helicoidal, ou seja, um perfil rotacionado em torno de um eixo ao longo de uma trajetória (VIEIRA, 1994, p. 24).

Girando a superfície de uma maneira similar a varredura, quaisquer combinações destes efeitos (*sweep* translacional e rotacional) podem ser utilizadas para desenvolver uma grande variedade de formas básicas (Figura 21).

Figura 21 - *Sweeping* helicoidal.



Fonte: VIEIRA, 1994, p. 25.

Desta forma os pontos da superfície no vetor são dados pela equação abaixo (HILL, 1990, p. 294), acrescidos pelo valor do deslocamento translacional.

$$p(u,v) = (x(v)\cos(u),\ x(v)\ \mathrm{sen}(u),\ z(v))$$

5 DESENVOLVIMENTO DO PROTÓTIPO

Para o desenvolvimento do protótipo utilizou-se a ferramenta de desenvolvimento Borland Delphi 5.0, com incorporação da biblioteca gráfica OpenGL.

5.1 ESPECIFICAÇÃO

Neste tópico pode-se observar os diagramas de contexto e hierárquico funcional. Estes diagramas possibilitam uma visão geral do funcionamento do protótipo, utilizando a representação gráfica.

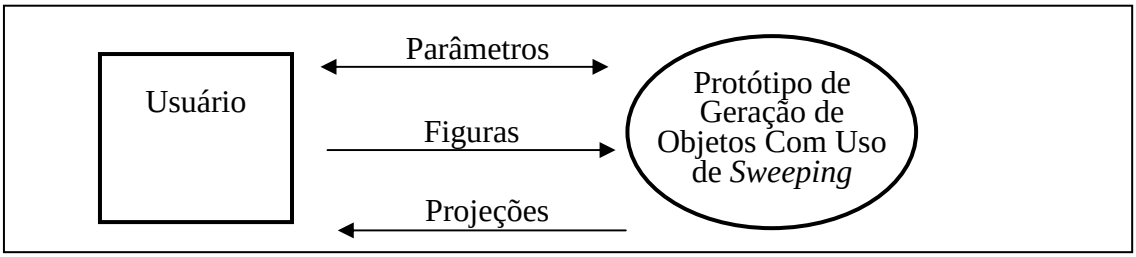
Prototipação ou prototipagem é um conjunto de técnicas e ferramentas de software para o desenvolvimento de modelos de sistemas. Seu principal objetivo é, portanto, antecipar ao usuário final uma versão (modelo) do sistema para que ele possa avaliar sua funcionalidade, identificar erros e omissões, mediante sua utilização (MELENDEZ, p.1, 1990).

Neste trabalho optou-se pela metodologia de prototipação de refinamento, que tem como passos: elaboração do DFD do sistema como um todo, elaboração da lista de elementos de dados, análise das entidades de dados, redesenho do DFD, divisão do DFD em unidades procedimentais e por último, ciclo genérico da prototipação.

5.1.1 DIAGRAMA DE CONTEXTO

Nesta seção tem-se a visão macro do funcionamento do protótipo e suas interações com o meio (Figura 22).

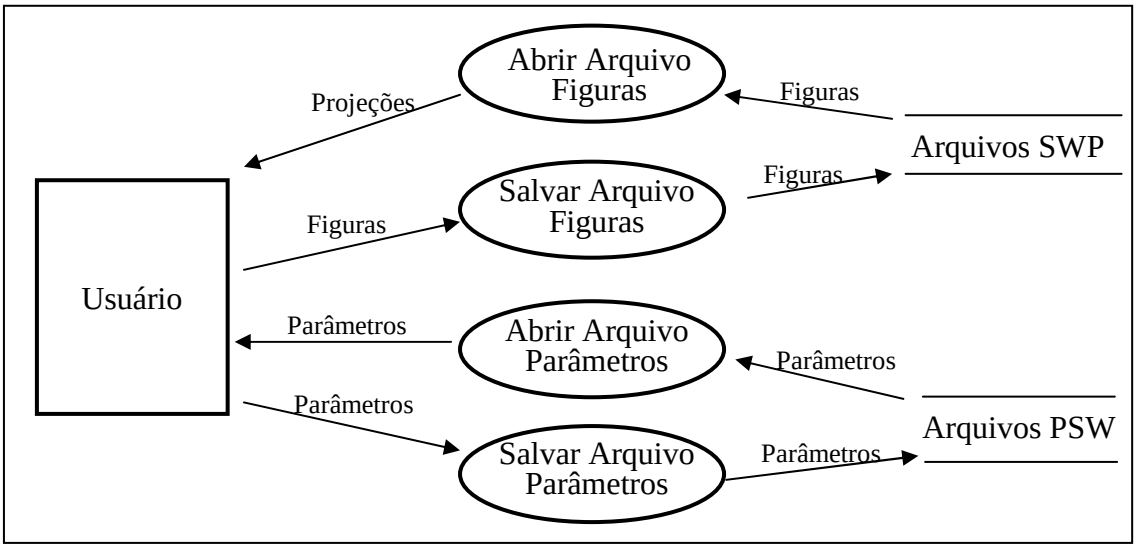
Figura 22 - Diagrama de Contexto.



5.1.2 DIAGRAMA DE FLUXO DE DADOS

Neste tópico pode-se observar o diagrama de fluxo de dados, que possibilita uma visão geral do funcionamento do protótipo, utilizando a representação gráfica (figura 23).

Figura 23 - Diagrama de Fluxo de Dados (DFD).



5.1.3 DICIONÁRIO DE DADOS

Na Tabela 1 pode ser observado o dicionário de dados (DD).

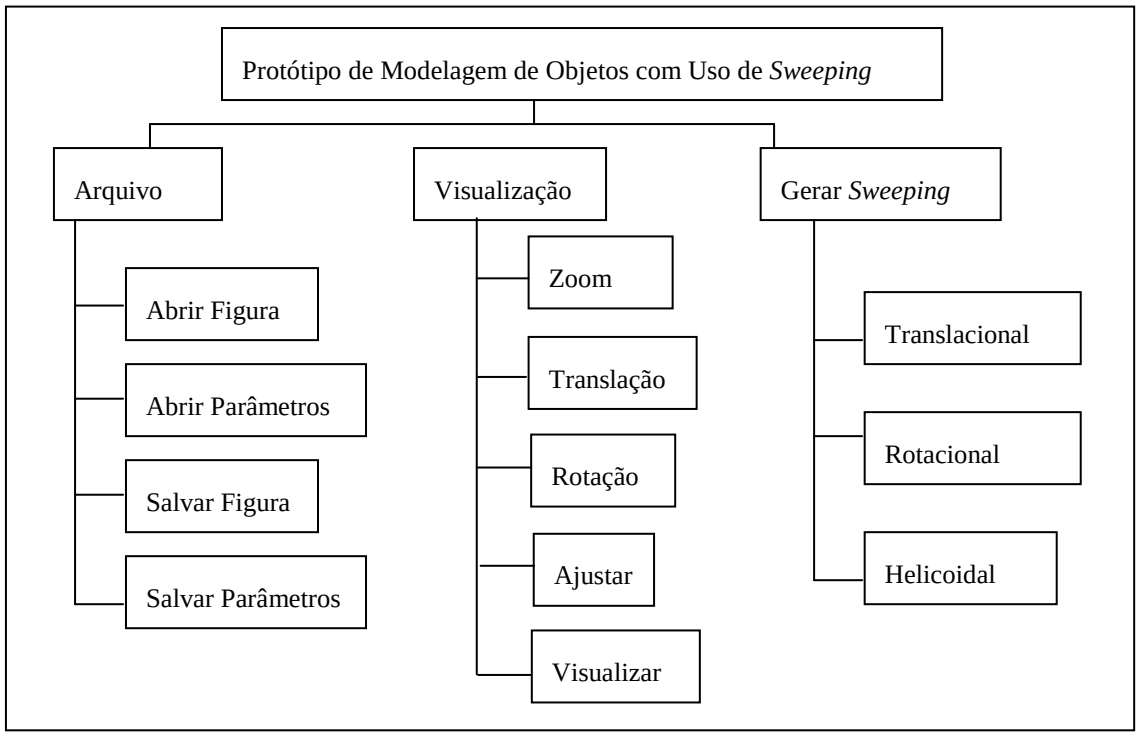
Tabela 1 - Dicionário de Dados

Nome	Tipo	Tamanho	Descrição
[Translacional]			
Deslocacao X	Caractere	4	Deslocação no eixo X
Deslocacao Y	Caractere	4	Deslocação no eixo Y
Deslocacao Z	Caractere	4	Deslocação no eixo Z
Numero Perfis	Caractere	3	Número de perfis a serem gerados
[Rotacional]			
Eixo Rotacao	Caractere	1	Eixo no qual o perfil será rotacionado
Numero Perfis	Caractere	3	Quantidade de perfis a serem gerados
Angulo Rotacao	Caractere	5	Ângulo da rotação
[Helicoidal]			
Deslocacao X	Caractere	4	Deslocação no eixo X
Deslocacao Y	Caractere	4	Deslocação no eixo Y
Deslocacao Z	Caractere	4	Deslocação no eixo Z
Eixo Rotacao	Caractere	1	Eixo no qual o perfil será rotacionado
Numero Perfis	Caractere	3	Quantidade de perfis a serem gerados
Angulo Rotacao	Caractere	5	Ângulo da rotação
GEOMETRIA			
Indice	Caractere	4	Índice do ponto
Coordenada Eixo X	Caractere	4	Coordenada do ponto no eixo X
Coordenada Eixo Y	Caractere	4	Coordenada do ponto no eixo Y
Coordenada Eixo Z	Caractere	4	Coordenada do ponto no eixo Z
TOPOLOGIA	Caractere	4xN	Indica as ligações entre os pontos

5.1.4 DIAGRAMA HIERÁRQUICO FUNCIONAL

Nesta seção é apresentado o diagrama hierárquico funcional, que possibilita a visão das funções principais do sistema (figura 24).

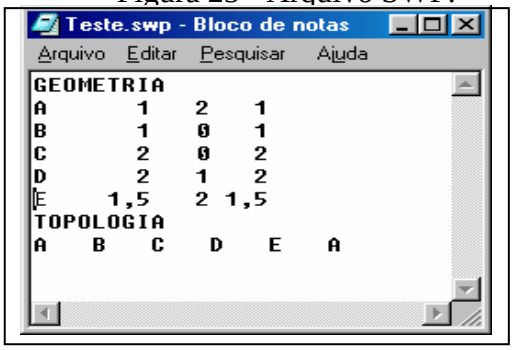
Figura 24 - Diagrama Hierárquico e Funcional.



5.1.5 ARQUIVOS

O arquivo SWP, tem como função armazenar os dados das figuras a terem o *sweep* gerado. Ele foi montado de tal forma, que os valores das coordenadas dos pontos que formam a figura, ficassem dispostos seqüencialmente e com um tamanho fixo. O arquivo está dividido em duas partes distintas, GEOMETRIA e TOPOLOGIA, como pode ser visto na figura 25, que mostra um exemplo editado no Bloco de Notas.

Figura 25 - Arquivo SWP.



A seção GEOMETRIA contém o conjunto de pontos da figura com seu índice e suas coordenadas (x, y e z), com tamanho fixo de 5 caracteres para o índice e 4 caracteres para o valor de cada eixo.

A seção TOPOLOGIA contém o conjunto de ligações entre os pontos para que formem a figura, com tamanho fixo de 5 caracteres para cada índice de ponto.

O arquivo PSW (Parâmetros do S*weeping*) é gerado no formato de arquivo utilizado pelo Delphi para geração de arquivos INI[5], criado a partir de uma variável do tipo `TiniFile`, para o uso do mesmo é necessário a biblioteca `IniFiles`. Este arquivo guarda os parâmetros utilizados para geração de um objeto de *sweeping*, como número de perfis, ângulo, translação, etc., podendo ser utilizado posteriormente para geração de objetos em outros arquivos (Quadro 2).

Quadro 2 - Formato dos arquivos PSW.

```
[Translacional]
Deslocacao X=2
Deslocacao Y=2
Deslocacao Z=2
Numero Perfis=4

[Rotacional]
Eixo Rotacao=1
Numero Perfis=30
Angulo Rotacao=360

[Helicoidal]
Deslocacao X=0
Deslocacao Y=1
Deslocacao Z=0
Eixo Rotacao=1
Numero Perfis=30
Angulo Rotacao=360
```

[5] Tipo padrão de arquivos onde aplicações armazenam informação de configuração, usualmente especificados com a extensão .INI.

Os conjuntos de dados são divididos em seção Translacional, seção Rotacional e seção Helicoidal (Quadro 2).

A seção Translacional contém o conjunto de parâmetros associadas ao *sweeping* de translação: deslocamento nos eixos x, y e z, e número de perfis gerado.

A seção Rotacional contém o conjunto de parâmetros associadas ao *sweeping* de rotação: eixo de rotação(`0=x; 1=y; 2=z`), número de perfis gerado e ângulo de rotação.

A seção Helicoidal contém o conjunto de parâmetros associadas ao *sweeping* helicoidal: deslocamento nos eixos x, y e z, eixo de rotação(`0=x; 1=y; 2=z`), número de perfis gerado e ângulo de rotação.

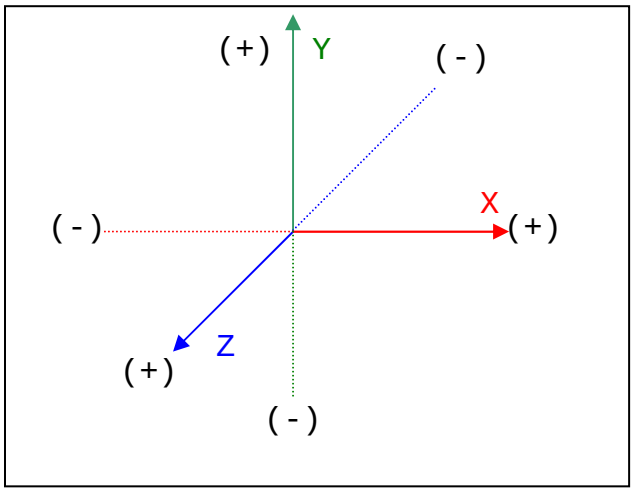
5.2 IMPLEMENTAÇÃO

Este capítulo descreve a implementação do protótipo para modelagem de objetos tridimensionais com base na técnica de *sweeping*, utilizando-se três modelos que são o translacional, rotacional e helicoidal. Nele, estão descritos os formatos das estruturas de dados utilizadas, a interface e os algoritmos utilizados para geração dos modelos.

Além dos componentes da biblioteca de visualização, foram utilizados componentes padrões do ambiente Delphi, como `ScrollBar` para alterar valores da visualização da câmera, como zoom e translação, caixas `Edit` para visualizar os valores da visualização e para receber os parâmetros da geração do *sweeping*, caixas `Memo` para possibilitar a visualização do *script* dos objetos e do *sweep*, `OpenDialog` para possibilitar a abertura de arquivos e `SaveDialog` para o salvamento dos mesmos.

O sistema de coordenadas utilizado é o da mão-direita, sistema que obedece a seguinte ordem: o eixo X cresce para a direita, a eixo Y cresce para cima e o eixo Z perpendicular ao papel e se aproximando do observador (Figura 26).

Figura 26 - Sistema mão-direita.



5.2.1 UTILIZANDO DELPHI E OPENGL

A implementação da aplicação utilizando-se do OpenGL no ambiente Delphi requer apenas que se cumpram apenas os dois seguintes passos:

- No diretório \WINDOWS\SYSTEM devem estar gravados os arquivos GLU32.DLL e OPENGL32.DLL, que são as bibliotecas necessárias para a execução de aplicações que se utilizem do OpenGL;
- No ambiente Delphi, deve-se instalar o pacote de componentes *SignSoft Visit Components 2.0* (Signsoft, 2001).

Após a execução do segundo passo acima citado, estarão disponíveis as pastas Visit e Visit Obj no ambiente Delphi (Figuras 27 e 28).

Figura 27 – Pasta de componentes Visit.

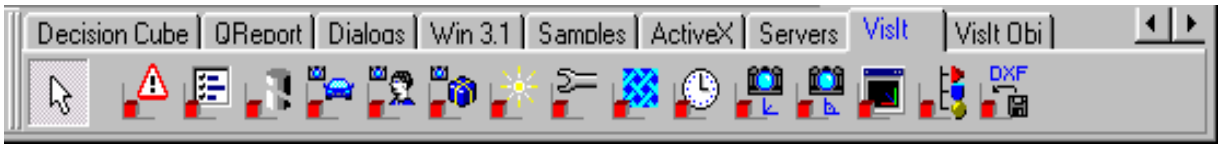
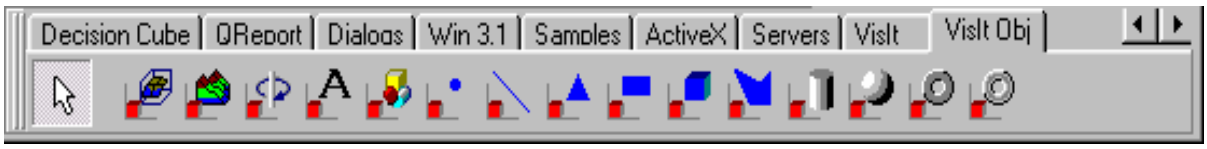


Figura 28 – Pasta de componentes Visit Obj.



5.2.2 VISUALIZAÇÃO DO OPENGL

Para visualização dos objetos, foram utilizados os componentes da biblioteca OpenGL, `VisView`, que é um componente básico necessário para renderizar a imagem desejada. Seu comportamento é similar a controles Windows comuns. Todas as ações irão ocorrer dentro deste componente; o `VisPerspectiveCamera`, que simula uma câmera, podendo-se definir a posição de visualização de uma imagem; `VisLight,` que serve para controle de luz; `VisAttributes, VisMaterial` que servem para definir atributos dos objetos, como a cor e por último o `VisTimer,` que serve para temporizar a atualização do que esta sendo exibido no `VisView`.

Utilizou-se para o armazenamento dos pontos, com três coordenadas (x, y, z), uma estrutura dinâmica como pode ser observado no quadro 3. Para o armazenamento da topologia uma estrutura dinâmica bidimensional, onde são guardados os índices dos pontos e cada linha desta matriz determina um segmento.

Quadro 3 - Estrutura de armazenamento dos pontos.

```
TPonto = record
   indice : string[4];
   x,y,z : single;
end;
TPontos = array of TPonto;
TLinhaTopo = array of string;
TTopologia = array of TLinhaTopo;
```

O componente `VisTimer` é disparado quando habilitado e executa suas linhas de código depois do tempo determinado na propriedade `Intervall`. Neste caso, é executado depois de 50 milisegundos, este tempo foi determinado para que o `VisView` atualize a tela e a atualização seja imperceptível pelo usuário. A `procedure Timer` possui a chamada de uma `procedure` para atualização da tela, como pode ser visto no quadro 4.

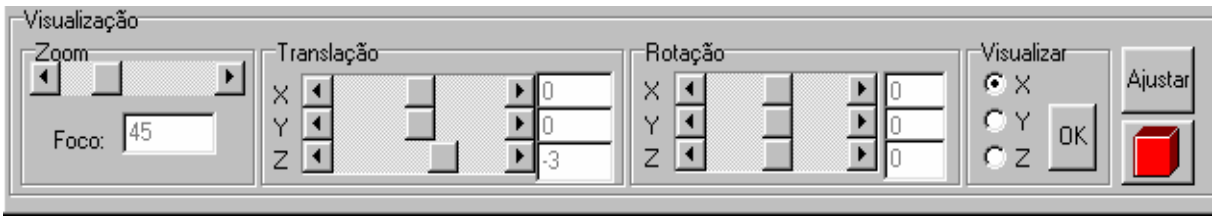
Quadro 4 - Procedure TimerTimer.

```
procedure TFrSweep.TimerTimer(sender: TObject; lagcount: Integer);
begin
 MeuView.InvalidateGL;
end;
```

A chamada a procedure para a atualização da tela, na qual devem estar todas as instruções para a geração de objetos, deverá estar no evento `OnRender` do componente `VisView`, sem o qual os objetos não serão exibidos. Nesta procedure também deverão estar as chamadas aos eventos `OnRender` dos componentes `TvisLight`, `TvisAttributes` e `TvisMaterial`.

Para facilitar a manipulação dos objetos, foram criados alguns componentes de controle das propriedades da câmera: zoom, rotação e translação. Além dos botões para ajustar a imagem e visualização (Figura 29).

Figura 29 - Controles de Visualização.



a) b) c) d)

A opção zoom (Figura 29.a), modifica o valor do ângulo de abertura da lente da câmera (ou foco), fazendo assim, com que o observador tenha a impressão que o objeto esta se aproximando.

A opção translação (Figura 29.b) muda o posicionamento da câmera em relação ao universo observado, alterando suas coordenadas. Para criar um efeito mais suave na mudança dos valores `x`, `y` e `z`, através da `ScrollBar`, dividiu-se o valor da sua posição por 10.

A opção rotação (Figura 29.c) modifica ângulo de rotação da câmera em relação aos eixos x, y e z, criando a impressão que o objeto esta sendo rotacionado.

A opção ajusta (Figura 29.d) tem como função alterar os valores de translação da câmera, de modo que qualquer objeto que esteja sendo visualizado possa ser observado por inteiro, e também no caso do mesmo ter saído do campo de visão do usuário devido a alguma alteração de posicionamento da câmera. Para tanto, utilizou-se o procedimento mostrado no Quadro 5.

Quadro 5 - Calculo de Ajuste de Visualização.

```
i ) Achar mínimos e máximos  (x, y e z);
ii) Calcular as dimensões;
  dimX := (Xmax – Xmin);
  dimY := (Ymax – Ymin);
  dimZ := (Zmax – Zmin);
iii) Encontrar dimensões;
iv) Xcamera :=  -1 * (Xmin + (dimX/2));
     Ycamera  :=  -1 * (Ymin + (dimY/2));
     Zcamera  :=  -2 * (Xmax + maiorDimensao);
```

A opção visualizar (Figura 26.d), faz a câmera dar um giro de 360° em torno do eixo definido pelo usuário, para que o observador possa ter uma noção melhor do objeto gerado, visualizando o mesmo, de vários ângulos. Para isso, calculou-se o raio inicial, e depois foi-se incrementando o ângulo e recalculando as novas posições de `x,y` e `z` a cada novo ângulo gerado.

5.2.3 *SWEEPING* TRANSLACIONAL

Optou-se pelo armazenamento dos pontos e da topologia gerada pelo *sweeping* translacional, em estruturas separadas das estruturas dos pontos e topologia do perfil original da figura, preservando assim as informações originais.

Tanto a quantidade de perfis a serem gerados, como os valores da translação (`x`, `y` e `z`) são definidos pelo usuário, sendo que o número de perfis é dois no mínimo, considerando o perfil original incluso neste valor. Os novos pontos gerados, necessários para a montagem do *sweep*, foram gerados somando-se valores da translação aos pontos do perfil original, tantas vezes quanto forem o número de perfis. O valor somado a cada geração de perfil é obtido dividindo-se o valor total da translação do eixo em questão, pelo número de perfis, menos um, multiplicado pelo numero do perfil atual (Quadro 6).

Quadro 6 - Algoritmo para geração dos pontos do *sweeping* translacional.

```
para n=1 até Total_Perfis faça
inicio
  {Calcula o valor do índice do deslocamento}
  u := n / (Total_Perfis-1);
  para i:=1 até Total_Pontos faça
  inicio
    {Acrescenta uma posição na lista de pontos de sweep}
    TamListaDePontos := TamListaDePontos +1;
    {Encontra indice do novo ponto}
    PontosSweep[ultimo].indice :=Pontos[i].indice+intParaString(n);
    {Calcula posições de X,Y,Z do novo ponto}
    PontosSweep[ultimo].X :=Pontos[i].X +(PtoTranslacao.X *u);
    PontosSweep[ultimo].Y :=Pontos[i].Y +(PtoTranslacao.Y *u);
    PontosSweep[ultimo].Z :=Pontos[i].Z +(PtoTranslacao.Z *u);
  fim
fim
```

A topologia do *sweep* foi obtida, criando-se cópias da topologia original, sucessivamente a cada novo perfil, esta nova topologia então, tem os índices dos pontos originais substituídos pelos índices dos pontos equivalentes nos novos perfis, mas mantendo a mesma configuração. A figura 30 mostra o perfil de um objeto, o mesmo objeto pode ser visto com o *sweep* translacional na figura 31.

Figura 30 - Perfil original de um objeto.

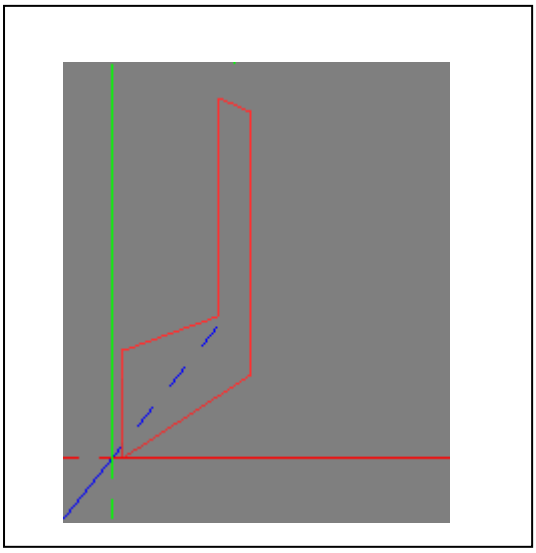
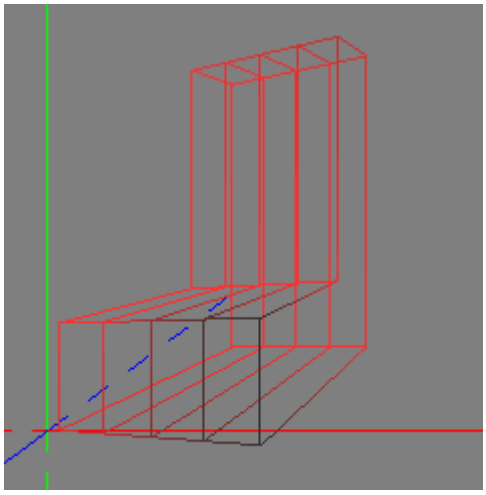


Figura 31 - Sweeping translacional.



```
Valores da translação:
X := 2;
Y := 0,2;
Z := 1;
Número de perfis: 5.
```

5.2.4 *SWEEPING* ROTACIONAL

Neste modelo, da mesma forma que no *sweeping* translacional, optou-se pelo armazenamento dos pontos e da topologia gerada pelo *sweeping* translacional, em estruturas separadas das estruturas dos pontos e topologia do perfil original da figura, preservando assim as informações originais.

Tanto a quantidade de perfis a serem gerados, como o ângulo da rotação e o eixo a ser usado na rotação (X, Y ou Z) ficam definidos pelo usuário, sendo que o número de perfis é dois no mínimo, considerando o perfil original incluso neste valor. Os novos pontos gerados, necessários para a montagem do *sweep*, são gerados somando-se valores as coordenadas dos pontos do perfil original, quantas vezes foram o número de perfis. O valor somado a cada geração de perfil é obtido através da equação que pode ser vista no quadro 7.

Quadro 7 - Algoritmo para geração dos pontos do sweeping rotacional.

```
α -> ângulo
X' :=  (X * Cos α) - (Y * Sen α)
Y' :=  (Y * Cos α) + (X * Sen α)
```

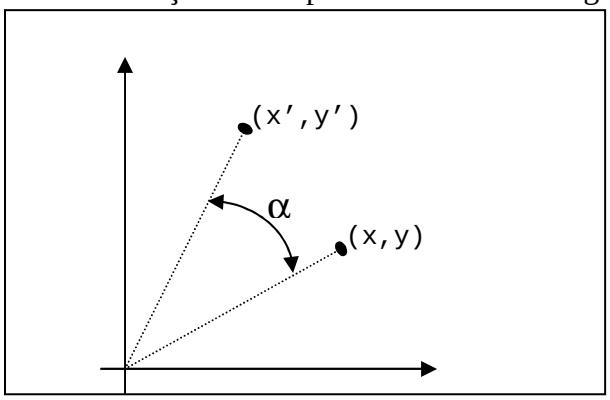
Fonte: Tori (1987, p.206)

Chama-se rotação em torno da origem a movimentação de uma figura para outra posição, de modo que todos os pontos da imagem mantenham a mesma distância da origem que possuíam antes da transformação. Para a obtenção de uma rotação em torno de um ponto qualquer que não a origem, deve-se fazer uma combinação de transformações de translação e de rotação em torno da origem (TORI, 1987, p.206).

Os parâmetros usados para a equação são o X , Y, Z e o ângulo α, que indica o deslocamento angular no sentido anti-horário.

Segundo TORI (1987, p.206) a fórmula pode ser melhor deduzida observando-se a figura 32, na qual tem-se o ponto de coordenadas (x,y), e o ponto (x′,y′) obtido pela rotação do primeiro, de um ângulo α.

Figura 32 - Rotação de um ponto em torno da origem.

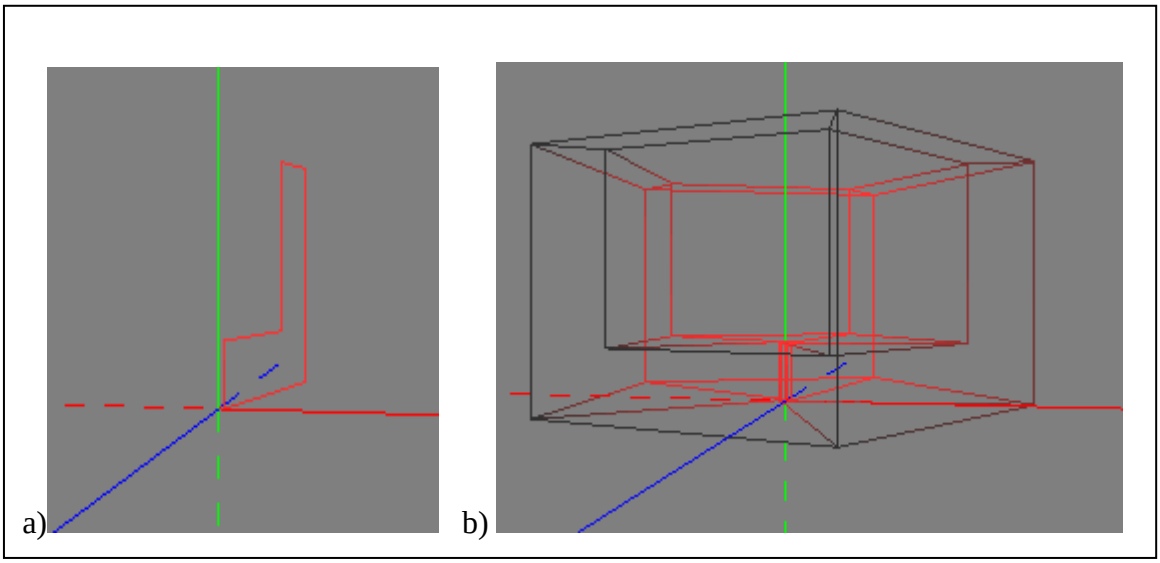


Fonte: Tori (1987, p.206)

A topologia do *sweep* foi obtida, criando-se cópias da topologia original, sucessivamente a cada novo perfil, esta nova topologia então, têm os índices dos pontos originais substituídos pelos índices dos pontos equivalentes nos novos perfis, mas mantendo a mesma configuração.

Observou-se na utilização de um ângulo de rotação igual a 360°, que o número de perfis que podem ser visualizados é igual ao número de perfis programado, menos um. Isso se deve ao fato de que o primeiro e o último perfil estão sobrepostos, como pode ser visto na Figura 33, em que o total de perfis é 6, mas só podemos visualizar 5 deles.

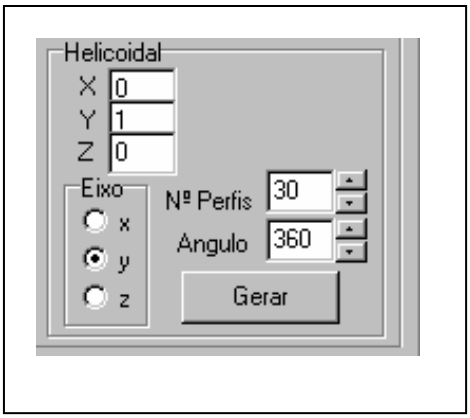
Figura 33 - *Sweeping* Rotacional.



5.2.5 *SWEEPING* HELICOIDAL.

Este modelo de *sweeping* reúne as duas técnicas anteriores, translacional e rotacional, recebendo os mesmo parâmetros (Figura 34) e gerando objetos com propriedades de ambos os modelos.

Figura 34 - Parâmetros do *sweeping* helicoidal.



A quantidade de perfis a serem gerados, valores da translação (`x`, `y` e `z`), o ângulo da rotação e o eixo a ser usado na rotação (`x`, `y` ou `z`) ficam definidos pelo usuário, sendo que o número de perfis é dois no mínimo, considerando o perfil original incluso neste valor. Os

novos pontos gerados, necessários para a montagem do *sweep*, são gerados somando-se valores as coordenadas dos pontos do perfil original, quantas vezes forem o número de perfis. O valor somado a cada geração de perfil é obtido através da equação que pode ser vista no quadro 6 (Seção 5.2.3).

5.3 FUNCIONAMENTO

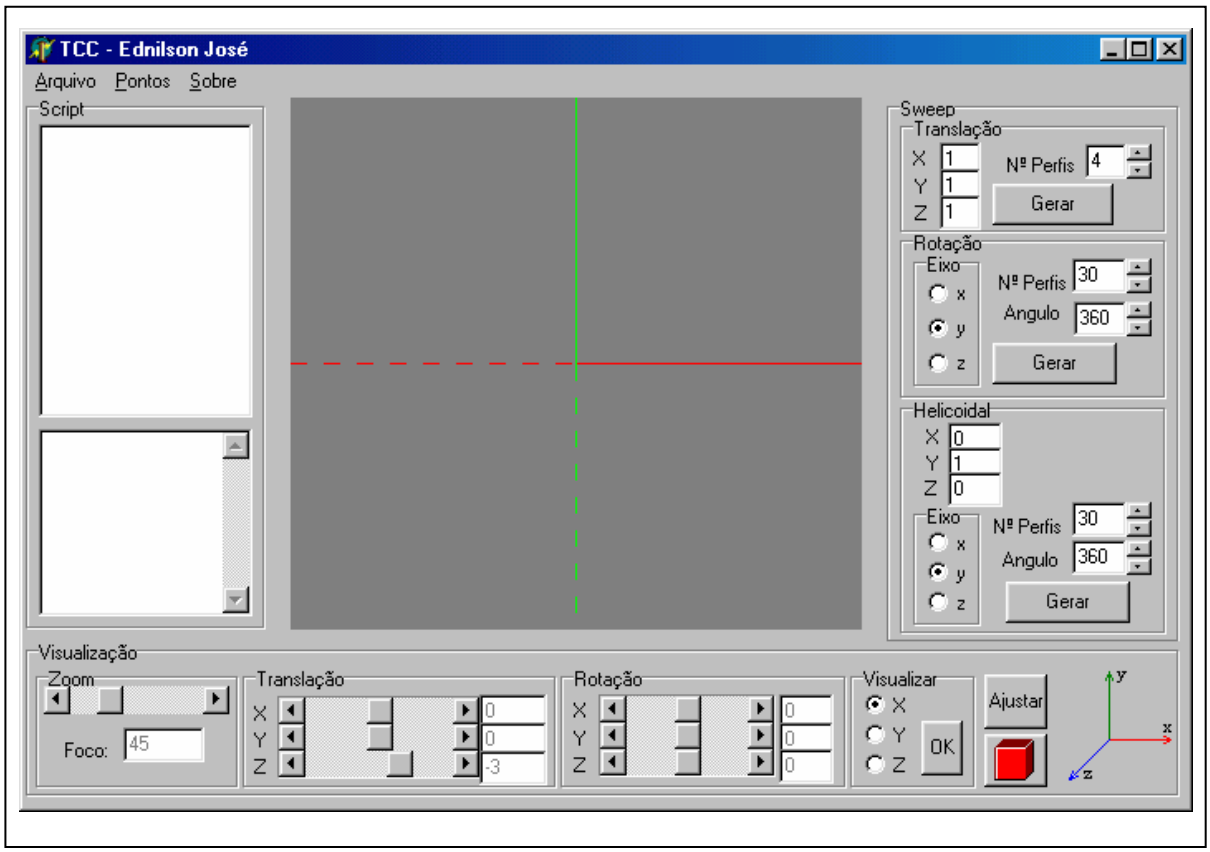
Esta seção explica todas as funções permitidas pelo protótipo para a modelagem de objetos 3D com uso de *sweeping*. O protótipo permite a construção de modelos a partir de *sweeping* translacional, rotacional e helicoidal. Os modelos gerados podem ainda ser armazenados em disco para posterior restauração. Também podem ser armazenados os parâmetros utilizados para a reutilização dos mesmos em outros objetos.

Ao executar o protótipo, é mostrada uma janela do tipo *About* do fabricante da biblioteca gráfica Signsoft (Figura 35), isto se deve ao fato de ser uma versão demonstrativa. Esta janela será fechada clicando-se no botão “OK” encontrado na mesma.

Figura 35 - Tela *About* do fabricante da biblioteca.

Após o fechamento da tela anterior, encontra-se o *layout* geral contendo o menu principal, quatro seções principais. A primeira à direita da tela contendo os *scripts* da figura, ao centro a área onde será exibida a figura, a sua esquerda a parte com os parâmetros para geração do *sweep* e abaixo, os controles de visualização (Figura 36).

Figura 36 - Tela principal do protótipo.

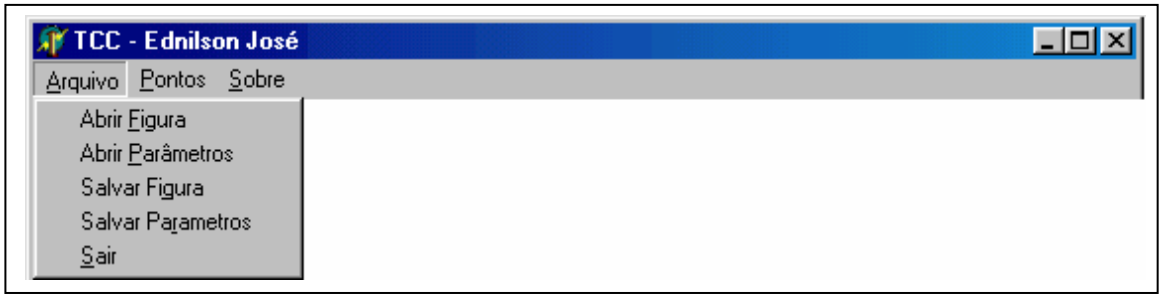


No menu principal encontram-se três submenus: o submenu Arquivo, o submenu Pontos e o submenu Sobre, como pode ser visto na figura 37.

Figura 37 - Menu Principal.

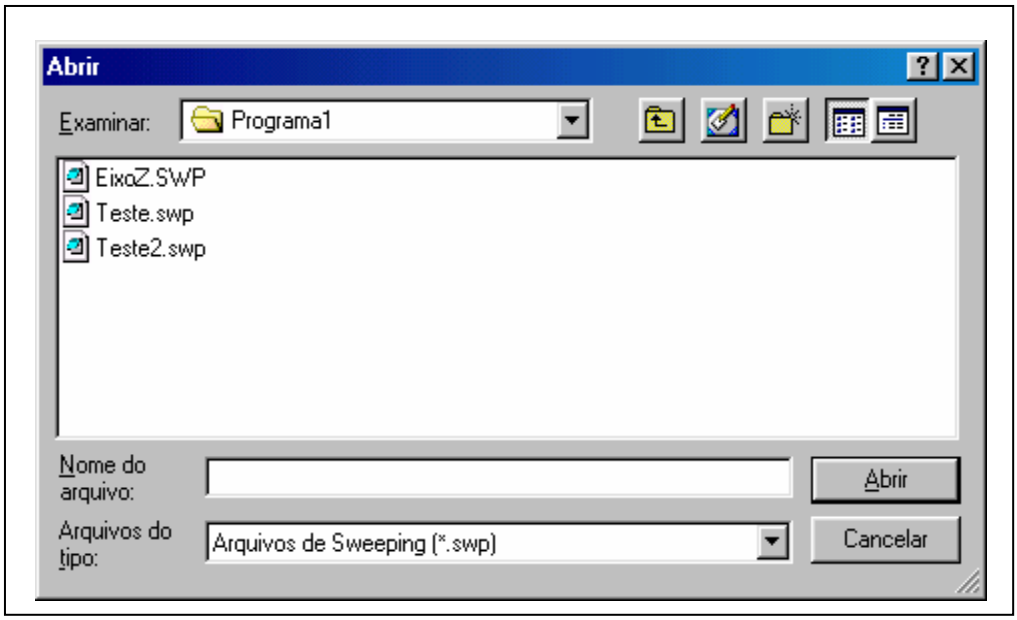
O submenu arquivo possui as opções **Abrir Figura**, **Abrir Parâmetros** **Salvar**, **Fechar** e **Sair**, mostradas na figura 38. Estas opções geralmente ficam neste submenu, segundo o padrão Windows.

Figura 38 - Submenu Arquivo.



A opção **Abrir Figura** permite abrir um arquivo de extensão ".swp" como foi explicado na seção 5.1.5. Ao executar esta opção, uma caixa de diálogo é apresentada para a localização do arquivo (Figura 39).

Figura 39 - Caixa de diálogo da opção Abrir Figura.



A opção **Abrir Parâmetros** permite abrir um arquivo de extensão ".psw" como visto na seção 5.1.5. Ao executar esta opção, é mostrada uma caixa de diálogo para a localização do arquivo semelhante a apresentada na figura 36, mas com o item "Arquivos do tipo:" com a extensão ".psw".

A opção **Salvar Figura** salva um arquivo de extensão ".swp" contendo os índices e as coordenadas da figura, ou seja, esta opção salva o perfil da figura (Figura 38). Esta opção mostra uma caixa diálogo semelhante a da opção **Abrir Figura** para a escolha do diretório onde se quer salvar.

A opção **Salvar Parâmetros** salva um arquivo de extensão ".psw" contendo os parâmetros utilizados na geração do *sweep*, podendo guardar os parâmetros dos três modelos possíveis, translacional, rotacional e helicoidal. Esta opção mostra uma caixa diálogo semelhante a da opção **Abrir Parâmetros** para a escolha do diretório onde se deseja salvar o arquivo.

No submenu Pontos está a opção: **Mostrar Pontos**, como mostra a figura 40.

Figura 40 - Submenu Pontos.

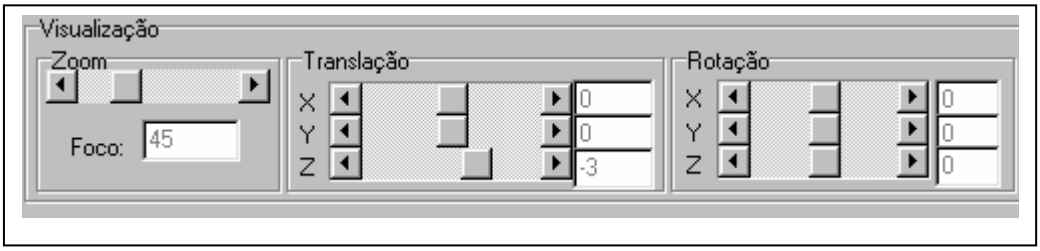
A opção **Mostrar Pontos** define se serão ocultos ou mostrados os pontos na área de visualização, os pontos somente podem ser observados se não estiverem relacionados a nenhuma topologia, caso estejam eles serão automaticamente ocultos pelo segmento formado por ele em conjunto com outro(s) ponto(s) , o protótipo é inicializado com esta opção ativa.

O submenu **Sobre** mostra uma janela com informações gerais sobre o desenvolvimento do protótipo (Figura 41).

Figura 41 - Submenu Sobre.

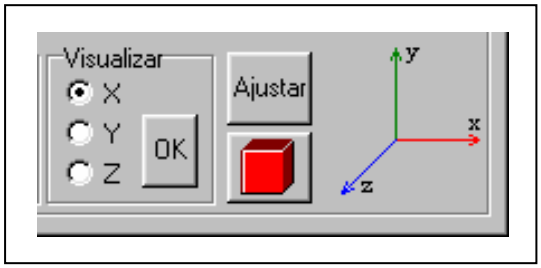
A figura 42 possui três itens da parte de controle de visualização, que permite alterar as propriedades da câmera. A esquerda da figura está o ajuste de zoom da imagem; ao centro os controles de translação e à sua direita os controles de rotação.

Figura 42 - Controles de visualização.



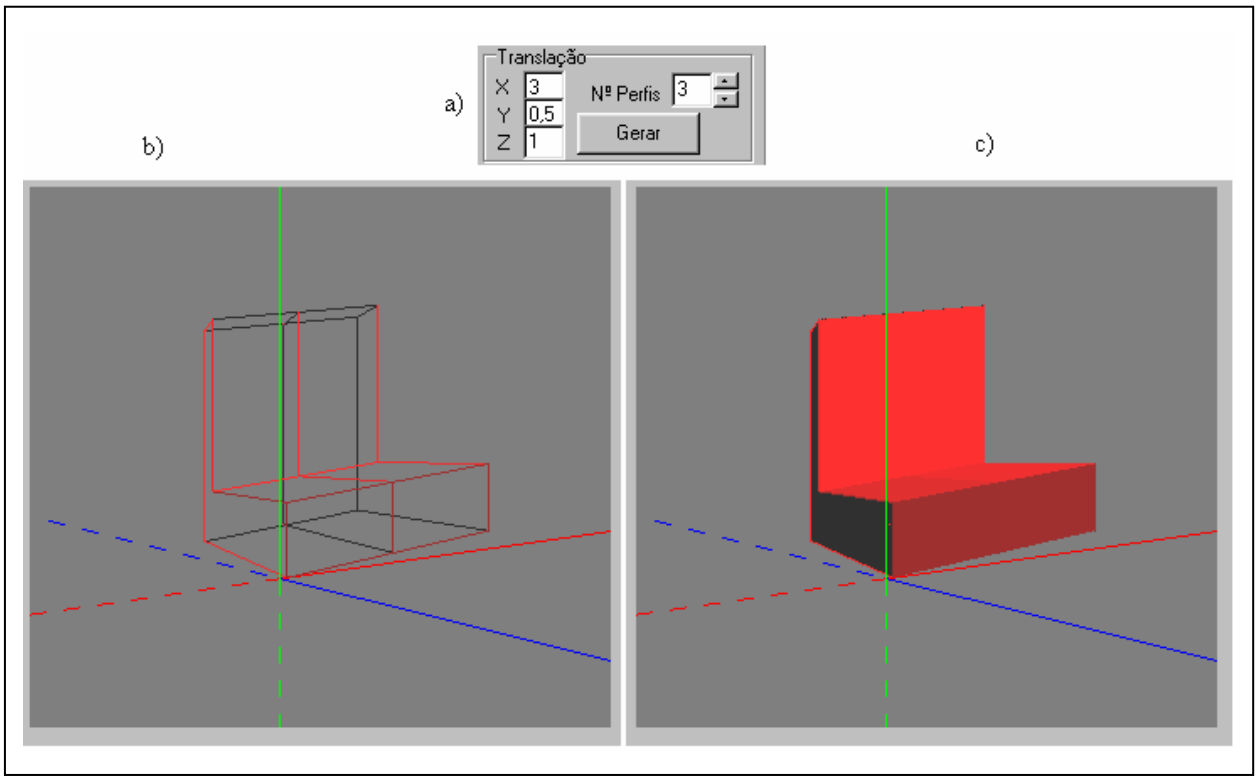
A figura 43 mostra as opções **Visualizar,** os botões **Ajuste** e de alternância entre visualização aramado/preenchido. O item **Visualizar**, faz a câmera dar um *loop* de 360° em torno dos eixos x, y ou z, dando ao usuário uma melhor perspectiva do objeto como um todo. E por último, a direita, um modelo do sistema de coordenada mão-direita, para facilitar o entendimento do usuário sobre a regra utilizada.

Figura 43 - Outras opções de visualização.



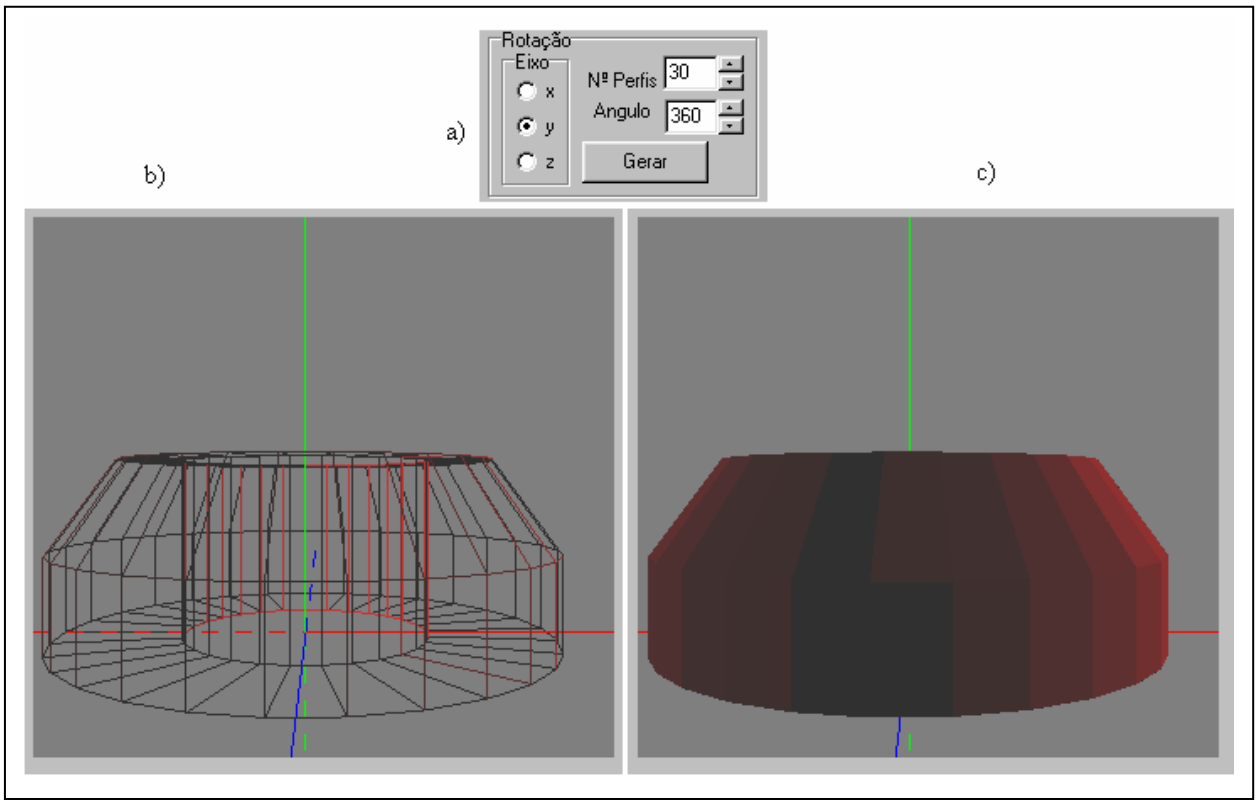
A figura 44.a mostra o item **Translação**, onde são colocados os parâmetros para a geração do *sweeping* translacional, e logo abaixo em **b)**, um exemplo de imagem gerada com a superfície em aramado e a sua direita em **c)** a mesma figura com superfície preenchida.A representação com superfície preenchida serve apenas para ilustrar melhor o resultado final do *sweep*.

Figura 44 - *Sweeping* Translacional.



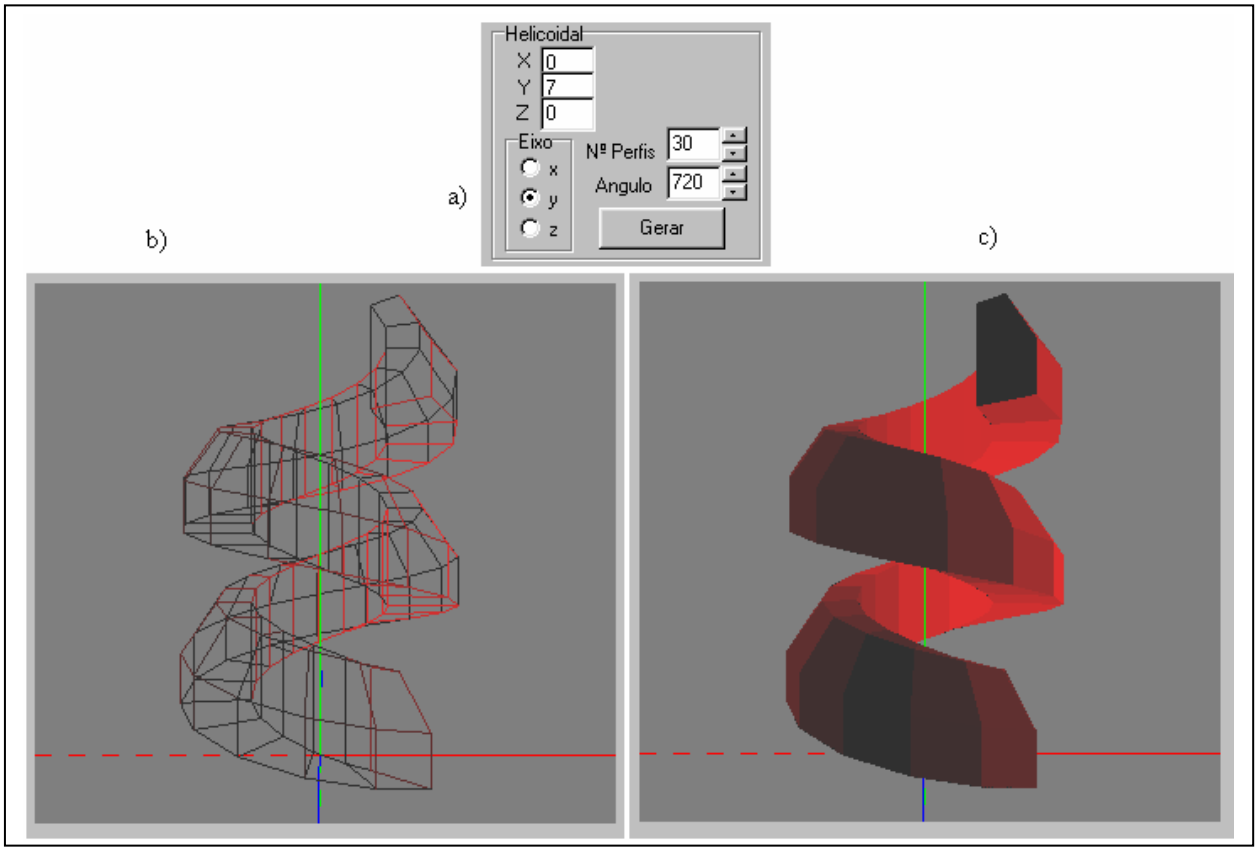
A figura 45.a mostra o item **Rotação**, onde são colocados os parâmetros para a geração do *sweeping* rotacional, e logo abaixo em **b)**, um exemplo de imagem gerada com a superfície em aramado e a sua direita em **c)** a mesma figura com superfície preenchida.

Figura 45 - *Sweeping* Rotacional.



A figura 46.a mostra o item **Helicoidal**, onde são colocados os parâmetros para a geração do *sweeping* helicoidal, e logo abaixo em **b)**, um exemplo de imagem gerada com a superfície em aramado e a sua direita em **c)** a mesma figura com superfície preenchida.

Figura 46 - *Sweeping* Helicoidal.



6 CONSIDERAÇÕES FINAIS

Nesta seção serão relatadas as conclusões obtidas com este trabalho, bem como as sugestões para trabalhos futuros.

6.1 CONCLUSÕES

A pesquisa dos conceitos viabilizou a geração de objetos tridimensionais gerados com uso da técnica de *sweeping*, através da implementação de um protótipo de software. Foram gerados os três modelos propostos: translacional, rotacional e helicoidal, com as opções de visualização aramado e preenchido, possibilitando uma melhor observação do objeto final.

O ambiente de desenvolvimento Delphi mostrou-se eficiente no que diz respeito a oferecer recursos para a manipulação das variáveis e geração dos arquivos. A biblioteca gráfica OpenGL também mostrou-se eficiente, tendo uma vasta quantidade de recursos, sendo utilizados no projeto apenas uma fração deles.

Percebeu-se que se utilizando determinados parâmetros a certos perfis inicias podem gerar objetos confusos, como por exemplo um *sweeping* rotacional em torno de um determinado eixo para um perfil inicial com todos os pontos contendo todas as coordenadas daquele eixo iguais, ou ainda, um *sweeping* helicoidal com um valor de translação muito pequeno, gerando alguns perfis que se interseccionem com outros.

Desta forma o trabalho presente obteve êxito no seu propósito, levando à aprendizagem do ambiente de desenvolvimento, da biblioteca gráfica OpenGL e das técnicas para a geração de objetos tridimensionais através da técnica de *sweeping*.

6.2 EXTENSÕES

Como extensões deste trabalho, poderiam ser citados:

- Utilização de técnicas de *spline* para calcular a trajetória produzida no *sweeping* translacional ou para suavizar as curvas geradas no modelo rotacional que aparecem

principalmente quando a quantidade de perfis não é muito grande, o que torna o processo lento;

- O *sweeping* rotacional do protótipo atual é feito sempre em torno de um dos eixos (x, y, z), em um trabalho futuro poder-se-ia fazer a rotação em torno de um ponto qualquer;

- O preenchimento das superfícies ficou com apenas um modelo, podendo ter mais opções, como preenchimento com texturas, preenchimento das extremidades o objeto, controle da luz e cores para melhor compreensão resultado obtido, etc;

- Adoção de uma interface com possibilidade de visualizar o objeto de vários ângulos simultaneamente;

- Outras formas de modelagem, encontradas nas opções não exploradas em OpenGL no trabalho atual.

REFERÊNCIAS BIBLIOGRÁFICAS

BORGES, José Antônio dos Santos. **Introdução às técnicas de computação gráfica 3D.** Rio de Janeiro: Núcleo de Computação Eletrônica, 1989.

BRITO, Agostinho de Medeiros Jr. **Introdução à computação gráfica com OpenGL**, Natal, mar. 2001. Disponível em: <http://www.leca.ufrn.br/~ambj/ele435/opengl>. Acesso em: 29 mar. 2001.

CORRIGAN, John. **Computação gráfica:** segredos e soluções. Rio de Janeiro: Ciência Moderna, 1994.

FOLEY, James D.; DAM, Andries Van. ***Computers Graphics: Principles and Practice*.** New York: Addison Wesley, 1990.

HILL, Francis S. ***Computer graphics***. New York: Macmillan Publishing Company, 1990.

LANSDOWN, John; EARNSHAW, Rae A. ***Computers in Art, Design and Animation*.** London: Springer-Verlag, 1989.

MELENDEZ, Rubem Filho. **Prototipação de Sistemas de Informações.** Rio de Janeiro: LTC - Livros Técnicos e Científicos Editora Ltda, 1990.

MORTENSON, Michael E. ***Geometric Modeling*.** New York: John Willey & Sons, Inc., 1985.

NEWMAN, Willian M.; SPROULL, Robert F. ***Principles of Interactive Computer Graphics*.** Singapore: McGraw-Hill, 1979.

OPENGL Org. ***OpenGL - The Industry's Foundation for High Performance Graphics***, EUA, mar. 2001. Disponível em: <http://www.opengl.org/developers/about/overview.html#1>. Acesso em: 29 mar. 2001.

PERCIANO, Ronaldo Cesar Marinho; OLIVEIRA, Antonio Alberto Fernandes de. **Introdução a Computação Gráfica.** Rio de Janeiro: LTC - Livros Técnicos e Científicos Editora Ltda, 1989.

PLASTOCK, Roy A.; KALLEY, Gordon. **Computação gráfica.** Portugal: McGraw-Hill, 1986.

REIS, Dalton Solano. **Análise de Algoritmos de Triangularização para Geração de Grades Regulares Retangulares.** Dissertação. Porto Alegre-RS : UFRGS – CPGCC, 1997.

SIGNSOFT. ***Signsoft GmbH - Software Development, Consulting, 3D Graphics***, Alemanha, mar. 2001. Disponível em: < http://www.signsoft.com/visit>. Acesso em: 10 mar. 2001.

SILVA, Fernanda Andrade Bordallo da. **Protótipo de um Ambiente para Geração de Superfícies 3D com uso de *Spline* Bézier.** 2000. 50 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

TORI, Romero; ARAKAKI, Reginaldo; MASSOLA, Antonio Marcos Aguirra et al. **Fundamentos de Computação Gráfica.** Rio de Janeiro: LTC - Livros Técnicos e Científicos Editora S.A., 1987.

VIEIRA, Micheus Luiz. **Construção e visualização de modelos tridimensionais utilizando a técnica de *Sweeping*.** 1994. 64 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

ZINDARS, Gerson Paulo. **Implementação de uma câmera sintética para visualização de objetos tridimensionais.** 1993. 112 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau.