MODELAGEM DE OBJETOS

POR SPLINES

Este exemplar corresponde à redação final da tese defendida por Rosane Minghim e aprovada pela Comissão Julgadora em 29/01/90

Autor: Rosane Minghim
Orientador: Clésio Luis Tozzi

jan/1990

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA

Aos meus pais,

sentido de tudo

que faço.

Agradecimentos

Ao Prof. Dr. Fernão Stella de Rodrigues Germano, pelo valioso apoio e pelo constante incentivo em todas as minhas iniciativas.

À Luisa Aparecida Spadacini Laera pela paciente colaboração na montagem final desta dissertação.

A todos os demais, que direta ou indiretamente colaboraram para tornar possível a conclusão deste trabalho.

RESUMO

Com o objetivo de tratar o problema de representação de objetos tridimensionais em computação gráfica, este trabalho apresenta um estudo do assunto, destacando a modelagem de sólidos por superfícies matemáticas, uma vez que ela representa uma alternativa útil e flexível de manipulação de objetos. Dois métodos de modelagem superficial são abordados: Bézier e B-spline. Um sistema de modelagem é implementado usando tais métodos, com atenção especial aos aspectos de interação homem/máquina, como a introdução dos dados tridimensionais necessários e visualização dos objetos projetados. O desenvolvimento do sistema é baseado em técnicas de Engenharia de Software, selecionadas para a resolução desse tipo de problema.

ABSTRACT

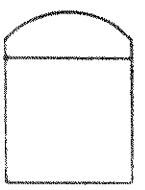
Dealing with the problem of object representation in Computer Graphics, this paper presents a review of the literature on the subject highlighting the modeling of solid objects through Mathematical Surfaces, as this shows up as a usefull and flexible alternative to solve the problem. Two methods for mathematical modeling are focused in detail: Bézier and B-spline. Using these methods a Modeling System is implemented, giving special atention to the interface aspects of man/machine interation, such as the introduction of the necessary tridimensional data and the visualisation of the objects projected. This implementation is based on Software Engineering techniques selected for solving this kind of problem.

NOTAÇÃO

Da técnica de Análise Estruturada:

CONVENÇÃO DOS DIAGRAMAS DE FLUXOS DE DADOS


depósito de dados

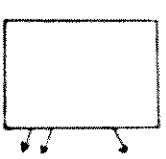

processo

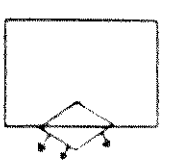

fluxo de dados

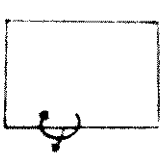

entidade externa

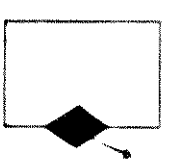
Da técnica de Projeto Estruturado:

CONVENÇÃO DOS DIAGRAMAS DE ESTRUTURA

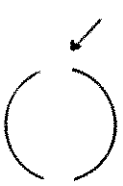

sequência


seleção


iteração


execução condicional

modulo detalhado separadamente


referência a outros módulos


módulos referenciados em outros lugares

Da técnica de Programação Estruturada:

CONVENÇÃO DA LINGUAGEM ALGORITMICA

Atribuição:

a <-- b

Teste:

Se (condição) **então**

    (bloco de comandos - condição verdadeira)

senão

    (bloco de comandos - condição falsa)

fim se

Iterações:

por contagem:

para variavel <-- valor1 **até** valor2 **faça**

     (bloco de comandos)

fim para

por condição:

enquanto (condição) **faça**

    (bloco de comandos)

fim enquanto

GLOSSÁRIO

"box"	Região delimitada do espaço, em formato de um paralelepípedo.
CAD	Computer Aided Design
cabeça da lista	Ponteiro para o primeiro elemento da estrutura de dados Lista Encadeada.
ciclo de vida do softwre	Conjunto de fases pelas quais passa o software desde a sua concepção até deixar de ser útil.
controle global	Característica do método matemático de representação, onde uma alteração nos pontos de controle provoca alterações por toda a superfície.
controle local	Característica do método matemático de representação, onde uma alteração nos pontos de controle provoca alteração da superfície apenas numa região de vizinhança limitada, possível de ser determinada.
coerência da imagem	Característica necessária da imagem de um objeto para que seja possível processá-la por determinado algoritmo de visualização.
diagrama de blocos N-S	Técnica de diagramação de programação estruturada.
domínio (de um método)	Conjunto de objetos que um modelo é capaz de representar.
funções-base	O mesmo que funções de "blending".
funções de "blending"	Funções que identificam um determinado método de representação de superfícies. Funções que combinam os pontos de controle para um determinado método.
grafo de controle	Conjunto de pontos de controle que definem a forma de uma superfície.
GRID	Conjunto de pontos sobre uma superfície matemática.
"lofting"	Método de visualização que mostra as "linhas" de uma superfcie em apenas uma das direções de variação dos parâmetros.

modelagem analítica	Modelo de sólidos baseado em funções numéricamente definidas.
modelagem matemática	Idem a modelagem analítica.
origem de aresta	Ponto inicial de uma aresta direcionada.
PAC	Projeto Auxiliado por Computador. O mesmo que CAD
"patch"	Pedaço de superfície paramétrica.
pontos de controle	Pontos que definem o formato de uma curva/superfície matemática.
quadriculado	O mesmo que Grid.
"ray tracing"	Método de produção de imagens reais.
rede	O mesmo que Grid.
rede Bézier	Grid de pontos calculados pelo método de Bézier.
rede B-spline	Grid de pontos calculados pelo método B-spline.
rede de controle	O mesmo que grafo de controle.
"rendering"	Processo de geração de imagens.
SMS	Sistema de Modelagem de Superfícies
"top-down"	Técnica descendente de abordagem de problemas (dividir para conquistar). Princípio da análise, projeto e programação estruturados.
vértices de controle	O mesmo que pontos de controle.
malha de controle	O mesmo que grafo de controle.

INDICE

APÊNDICES

1. Visualização e Perspectiva

2. Resumo das Funções da Interface de Entrada de Dados do SMS

3. Tabelas de Classificação de Quadriláteros

4. Diagramas do Módulo Mostra Quadriláteros.

5. Diagramas dos módulos Ordena por Boxes e Verifica_visibilidade.

CAPÍTULO I
INTRODUÇÃO

1.1 Considerações Iniciais

Tendo evoluído juntamente com o computador, a computação gráfica passou por várias fases, desde as primeiras aplicações, mais simples, até os sistemas gráficos de hoje, que são sistemas computacionais extremamente potentes em suas características de hardware e software.

A construção de sistemas gráficos engloba um conjunto de atividades e aplicações extremamente vasto. Nas aplicações, estão desde a utilização do computador na plotagem de dados numéricos até sistemas CAD dos mais complexos. Nas atividades referentes à construção de sistemas gráficos, estão desde a especificação do hardware adequado à aplicação até a construção de algoritmos potentes para manipulação de objetos.

Uma das atividades, na qual se concentra este trabalho, é a modelagem de objetos tri-dimensionais. Diz respeito à construção de esquema, técnicas e estruturas (modelo) para representação e manipulação de objetos bi ou tri dimensionais computacionalmente.

Existem vários modelos para representação de sólidos propostos. Dentre eles, a modelagem analítica, que usa funções matemáticas e conjunto de pontos de controle no espaço euclidiano para representação de formas. Este trabalho cuida de estudar e implementar a modelagem analítica através de dois métodos diferentes, o de Bézier e o método B-spline.

1.2 Evolução da modelagem matemática

O estudo e implementação da modelagem matemática, objeto deste trabalho, surgiu da sua necessidade em aplicações para CAD. Entretanto, pode-se empregar hoje em dia em um conjunto bastante variado de aplicações, como, por exemplo, criação de cenas artísticas, tradução e análise de dados, e outras.

Em especial, o ajuste polinomial de polígonos (e malhas) de controle evoluiu com a tecnologia de CAD/CAM. Não propriamente seu estudo matemático, que é mais antigo, inicialmente investigado no século XIX e depois em 1946, e finalmente desenvolvido entre 1966 e 1973. Entretanto, sua utilização começou, em computação gráfica, não antes da década de 60.

No projeto de peças mecânicas, por volta de 1960 eram empregadas as Curvas Francesas no desenho de bordas suavizadas para projeto de peças. Nessa época, com todo auxílio da geometria, a peça final não obedecia à forma projetada (Fa 88).

Com as máquinas de controle numérico, maior precisão da objeto era conseguida. Entretanto, para refletir essa forma o o projetista necessitava manipular um conjunto muito grande de números, cuja interpretação é bastante árdua.

O desenvolvimento dos primeiros sistemas CAD (UNISURF - projeto de automóveis) utilizavam-se apenas da Geometria como ferramenta de projeto. As primeiras curvas incorporadas em sistemas CAD eram baseadas em linhas retas, circunferências ou parábolas. Algumas curvas suavizadas eram incorporadas em sistemas deste tipo, por subdivisão em arcos de circunferência.

O uso de polinômios no projeto de curvas passou a ser observado e sugerido em função da informação gráfica resultante da sua plotagem. O uso de métodos de ajuste por polinomiais passou então a ser mais difundido e os fatores de seleção entre os métodos recaiam sobre suas características matemáticas mais desejáveis como tangente, curvatura e outras.

A introdução da interpolação cúbica em CAD ocorreu em 1964, por Ferguson. De lá para cá, a evolução dos sistemas CAD levou a uma evolução da utilidade de métodos de ajuste de superfícies por malhas de controle, em especial pela flexibilidade e informação visual que gera, sendo um auxílio para criação de projetos e geração automática dos números necessários para as máquinas de manufatura de peças e para a observação do ponto de vista do projetista.

Além dos usos em CAD, toda aplicação da modelagem de sólidos pode ser flexibilizada com a inclusão de representações de superfícies polinomiais. O emprego de um método adequado de representação matemática pode de fato superar outras técnicas convencionais, para aplicações específicas, ou mesmo trabalhar em conjunto com elas, ampliando a flexibilidade de formas dos objetos modelados.

A implementação de uma técnica de modelagem analítica, como a de qualquer outra, exige o tratamento de muitos aspectos computacionais. Dentre os mais relevantes estão a interface com o usuário e as estruturas de dados internas e de armazenamento externo. Tais aspectos formam a preocupação básica do presente trabalho.

1.3 Dos objetivos

O trabalho engloba um panorama das técnicas de modelagem, um estudo comparativo estre as técnicas de representação por funções matemáticas e a implementação de um sistema que trabalha com duas delas.

O objetivo final é tratar de aspetos computacionais de implementação de tais técnicas, permitindo a construção de módulos de programa consistentes que possam ser empregados como sub-sistemas de outros sistemas de modelagem de maior porte, ou que possam ser posteriormente adaptados a uma aplicação específica.

Os principais elementos que formam a preocupação central do trabalho são: a interface de entrada de dados, isto é, os recursos para o usuário manipular dados tri-dimensionais, em comunicação com o sistema; a visualização de superfícies matemáticas, ou seja, algoritmos e estruturas de dados para observação do objeto modelado, com algum tratamento da imagem, para o auxilio na identificação da forma obtida; a modelagem em si, ou seja, algoritmos e estruturas de dados para a avaliação da superfície projetada.

A interface de entrada é de propósito geral, fornecendo as funções principais na introdução dos dados necessários para a modelagem superficial do objeto.

A interface de saída (visualização) realiza o tratamento das linhas escondidas, como auxilio na identificação das partes do objeto projetado.

A modelagem é implementada por dois métodos diferentes, usualmente empregados : o método de Bézier e o método B-spline de representação matemática de superfícies.

Os aspectos relativos à implementação de sistemas deste tipo são discutidos e comparados.

1.4 Apresentação

O trabalho trata de aspectos computacionais da modelagem matemática superficial de sólidos tri-dimensionais.

O capítulo 2 apresenta os métodos e a bibliografia existente sobre a modelagem geométrica de objetos.

O capítulo 3 apresenta os métodos empregados no desenvolvimento do sistema.

O capítulo 4 apresenta o sistema desenvolvido, os módulos e estruturas de dados resultantes.

O capítulo 5 apresenta os resultados alcançados.

O capítulo 6 define as conclusões e sugestões para novas pesquisas.

CAPITULO II
SOBRE A MODELAGEM DE OBJETOS 3D

2.1 Considerações Iniciais

O presente capítulo tem o objetivo de fornecer uma apresentação da área de Computação Gráfica em que se insere o trabalho e seus aspectos gerais.

O capítulo não se concentra em apresentar os sitemas disponíveis para a modelagem, mas sim um apanhado do que existe na literatura sobre os aspectos que envolvem a implementação de tais sistemas, tanto da parte matemática como da parte computacional próprimente dita.

Dos modelos citados no capítulo, o trabalho concentra-se no modelo analítico. Entretanto, a modelagem empregada no trabalho define o objeto por sua superfície, enquanto que o modelo analítico citado modela o objeto pelo volume por ele ocupado.

Os tópicos abordados no capítulo podem ser encontrados em maiores detalhes nos trabalhos [Mi 89b] e [Mi 89c], desenvolvidos com o objetivo de relatar os estudos da área.

O parágrafo 2.2 apresenta a idéia geral da modelagem de objetos em computação gráfica.

O parágrafo 2.3 apresenta os aspectos específicos da modelagem superficial.

O parágrafo 2.4 discute os aspectos computacionais básicos relacionados com a implementação de um Sistema de Modelagem.

O parágrafo 2.5 comenta as técnicas de desenvolvimento de sistemas utilizadas no trabalho.

2.2 Modelagem de Sólidos

2.2.1 O que é um modelo

Os sistemas de manipulação de objetos por computador são tão antigos quanto o computador. Os primeiros sistemas possuiam uma representação do objeto com organização e estruturas de dados bastante simples, compatíveis com as aplicações.

A evolução das aplicações levou à evolução dos esquemas de representação e das estruturas de dados. Pode-se então identificar nestas aplicações a presença de modelos de construção de sólidos, através dos quais os objetos são manipulados internamente. Tais modelos geram estruturas de dados que são função de sua complexidade, e seus aspectos matemáticos são mais complexos, e envolvem geometria e topologia mais elaboradas.

Na sequência, identifica-se os principais modelos hoje aplicados a sistemas gráficos, e suas estruturas.

2.2.2 Técnicas para modelagem de sólidos

A evolução dos métodos de representação de objetos levou à elaboração de modelos com características matemáticas, topológicas e conceituais mais gerais do que simples estruturas de dados, que pudessem ser mapeadas posteriormente para uma representação computacional. Dentre elas, as mais difundidas são: [Sri 81] [Ma 88] [Mi 89c]

- DECOMPOSIÇÃO CELULAR: Compreende a definição da superfície de um objeto tri-dimensional pela justaposição de figuras planas, como triângulos (triangularização), ou quadriláteros. Tais

unidades são chamadas células e representam uma aproximação do sólido sendo modelado. São não ambíguas, embora não únicas, e facilitam o cálculo de certas propriedades dos objetos, como área e fronteira [Re 80].

- CSG (Constructive Solid Geometry): Representa um sólido como uma coleção de primitivas espaciais, por exemplo, paralelepípedos, cones, cilindros, esferas. Tais primitivas são combinadas por construções booleanas ou por operadores regulares. Assim, pode-se gerar um objeto pela união, intersecção ou subtração de vários desses sólidos primitivos. A utilização deste modelo é muito difundida [Re 80] [To 86] [Ma 88]. Os modelos são não ambíguos, mas não únicos. O conjunto de objetos possíveis de serem representados (domínio) é dependente do conjunto de sólidos primitivos e de operações admitidas. Uma limitação de CSG reside na determinação da fronteira do objeto que, não sendo incorporada diretamente no modelo, necessita de algoritmos que processem tal informação indiretamente.

- REPRESENTAÇÃO POR VARREDURA ("SWEEP"): Representar um sólido por sweep significa definir uma figura plana, e movimentá-la através do espaço para obter o objeto 3D desejado. Há o sweeping de translação (forma plana em movimento retilíneo) e o de rotação (figura plana em rotação ao redor de um eixo fixo). Seu esquema é não ambíguo, mas não único. Seu domínio é limitado a objetos com simetria de rotação ou translação. Pode ser usado como sistemática para entrada de dados.

- REPRESENTAÇÃO POR FRONTEIRA ou B-REPS (Boundary Representation): Todo sólido é representado pela definição de suas faces. Essas são segmentadas em um número finito de

"pedaços" que possam ser definidos separadamente. O conjunto de faces possíveis de serem representadas precisa ser limitado, pela dificuldade da manipulação de um conjunto muito grande de faces diferentes. Seu domínio é bastante vasto. Determina representações não ambíguas, mas não únicas para a maioria dos casos [Re 80], mas permitem a pronta disponibilidade de dados topológicos e facilitam determinados cálculos de propriedades bastante desejáveis, como a área superficial do objeto.

Tanto o esquema CSG quanto B-reps tem como representação básica uma árvore de componentes básicos combinados.

Várias das representações aqui citadas são conversíveis umas nas outras. Por exemplo, todo esquema CSG tem um esquema B-reps correspondente. Assim, é usual um sistema comercialmente disponível admitir mais de uma representação, permitindo um armazenamento interno por um modelo e uma interação com o usuário por outro, que torne mais acessível a manipulação do objeto [Ba 79].

- Modelos Alternativos: Muitas vezes são desenvolvidos modelos geométricos alternativos, com o objetivo de implementar um modelador para aplicações específicas, ou que permitam definir objetos difíceis de se manipular por aqueles métodos até aqui apresentados.

Um exemplo é o uso da Topologia Algébrica [Con 87], que pode facilitar o problema da intersecção entre objetos, para aplicações como química, por exemplo, onde uma molécula pode ser interpretada por um conjunto de esferas que se interseccionam.

Outro modelo atualmente bastante discutido é o Estocástico,

que visa resolver o problema de se modelar objetos irregulares, como os existentes na natureza, o que pelos métodos anteriores seria uma tarefa muito difícil. Consiste basicamente em identificar um processo Estocástico com comportamento "parecido" ao do fenômeno que se quer modelar [Fou 82]. As imagens desejadas são formadas obtendo-se realizações do processo estocástico base. Nessa classe de objetos estão: mapas, árvores, ondas, fumaça, água, nuvens.

Um outro modelo de abordagem relativamente recente é o Modelo Analítico de Sólidos [Ca 85], que visa auxiliar a manipulação de objetos de geometria complexa, geralmente não cobertos eficientemente pelos métodos convencionais. Para apoiar algumas aplicações que necessitem dessa flexibilidade, propõe definir sólidos matematicamente, o que, em muitos casos, apresenta vantagens sobre as outras representações, como por exemplo a generalidade e a facilidade de cálculo de determinadas propriedades.

Como para algumas aplicações são adequados vários dos esquemas de representação, frequentemente é proposta uma combinação de vários deles, com o objetivo de extender o domínio e facilitar a manipulação por parte do usuário. Um exemplo é a combinação de esquemas "tradicionais" (CSG, B-reps) como modelagem analítica [Ca 85].

Assim, a modelagem matemática visa aumentar a flexibilidade de representação de objetos, que não podem ser facilmente definidos ou manipulados por uma representação baseada apenas em outros esquemas ou modelos. Nesse contexto aparecem os métodos de representação de superfícies, que fazem parte da classe de

modelagem analítica, porém definindo sólidos matematicamente através de suas superfícies, e não do espaço que ocupam (volume). Existem vários métodos que podem apoiar um sistema de modelagem desse tipo, e os mais difundidos são apresentados a seguir.

2.3 Modelagem Matemática de Superfícies

Consiste numa ferramenta de definição de formas mais flexíveis para objetos tri-dimensionais, adequados a diversas aplicações. Alguns métodos tradicionalmente usados na Engenharia podem ser utilizados com esse objetivo.

Os métodos usados para esse fim consistem em interpolação de superfícies por funções matemáticas, utilizando pontos no espaço como referência para o formato final a ser obtido. O objetivo é obter uma equação paramétrica em função de dois parâmetros que tenha características de modelagem de formas adequadas à aplicação desejada. A escolha entre as várias formulações deve se concentrar nas características matemáticas do método em si, no comportamento das formas resultantes segundo as aplicações, na eficiência de implementação do método, e nos dados de entrada necessários para sua utilização (interface).

Cabe ressaltar que uma vantagem de todos esses métodos é a definição paramétrica, que admite independência do sistema de coordenadas. Outra vantagem é a possibilidade de representação de uma superfície por partes ("patches"), que podem ser justapostas para obtenção de um objeto mais complexo.

Além disso, esses métodos admitem formas multiplamente

avaliadas [Ne 83].

As representações mais usuais são brevemente apresentadas a seguir.

2.3.1 Superfícies Interpoladas ("Lofted surfaces")

Um dos primeiros métodos que pode produzir formas paramétricas bivariadas seria uma interpolação entre quatro pontos adjacentes. Tal abordagem indica um mapeamento entre um quadrado (onde os quatro pontos seriam vértices) e um "pedaço" de superfície no espaço. Esses vértices podem ser combinados por conjuntos de funções, ou curvas, cujo produto formaria uma superfície. Essa é a essência do método de Coons, um dos primeiros a serem utilizados em sistemas comercializados na indústria, e apresentado a seguir.

2.3.2 Superfícies de Coons

Apresentado em [Fo 72], serviu de base para o desenvolvimento de outros métodos. O método de Coons combina quatro curvas de fronteira para criar uma superfície. Essas curvas pré-definidas são combinadas com o uso de duas funções, denominadas "funções de blending", e dos quatro pontos limitantes do trecho de superfície que se deseja modelar. Como a definição original não pode garantir continuidade nem mesmo de primeira ordem, ele é extensível para obtenção de derivada de alta ordem. Para isso, o usuário tem necessidade de fornecer as derivadas parciais nos pontos limitantes do trecho de superfície.

Simplificações existem para evitar esse inconveniente. Isso se faz pela escolha de funções adequadas que garantam a continuidade desejada [Bar 84], por exemplo , as funções-base bi-cúbicas de Hermite.

Possui controle local, continuidade da função e da derivada primeira, e necessita como dados de entrada: os quatro vértices limitantes da superfície, as funções de "blending" (duas para superfície bi-linear e quatro para bi-cúbica) e quatro funções de curvas de fronteira.

A partir daí, os demais métodos de representação de superfícies são baseados na definição de uma malha de pontos de controle no espaço, definidos por suas coordenadas tri-dimensionais e de um conjunto de funções-base, chamadas funções de "blending" paramétricas. Combinadas, determinam a forma da superfície resultante. A definição das malhas de controle permite uma maior facilidade de modelagem interativa da forma. O que difere de um método para outro são as características das funções de "blending" que ele utiliza, e as propriedades que elas imprimem ao método. Para superfícies, tais métodos normalmente são extensão de sua utilização para curvas. Em [Mi 89b], que é um estudo sobre representação de superfícies, pode-se obter maiores informações sobre tais métodos, suas formulações e características. Uma breve apresentação é dada nos próximos parágrafos.

2.3.3 Superfícies de Bézier

é o produto tensor de curvas de Bézier. Assim, um ponto da superfície é obtido como uma média ponderada dos vértices de controle calculada a partir das funções-base de Bézier.

Nesse método, pertencem à forma final da superfície apenas os "cantos" da malha , isto é, os quatro pontos extremos que delimitam o trecho de superfície sendo modelado.

Os dados de entrada para este método são os vértices da malha de controle.

As funções de "blending" para o método são as polinomiais de Bernstein, que definem a distribuição binomial de probabilidades. Tais funções formam uma base no espaço vetorial dos polinômios de grau menor ou igual a m, m sendo o grau da superfície. Possui, portanto, controle global e grau dependente do número de pontos da malha de controle. Entretanto, controle local pode ser obtido pela definição por partes da superfície desejada.

As formas resultantes são orientadas pelo polígono de controle ("convex hull"), facilitando a manipulação interativa. Além disso, permitem controle da variação, ou seja, são suavizadas, sem grandes oscilações [Gi 78]. é versátil, e possui ordem de continuidade infinita [Ne 83]. Para se obter continuidade para trechos de superfícies vizinhos, os pontos de controle da fronteira de uma superfície dever coincidir com os pontos de controle da fronteira do pedaço ("patch") vizinho. A continuidade da derivada primeira se obtém alinhando os vértices de controle de malhas vizinhas, adjacentes à linha de fronteira.

2.3.4 Superfícies B-spline

De formulação não recente [Boo 72], sua aplicação em computação gráfica tem sido mais difundida nos últimos anos.

O esquema de representação por B-spline segue os mesmos moldes que o de Bézier, isto é, há um conjunto de funções base que combina os pontos de uma malha de controle para obtenção da forma final da curva. O nome splines, cujo conceito foi primeiramente introduzido por Schoenberg, é devido aos dispositivos usados por desenhistas e construtores de navios para esboçar curvas [Bar 84].

Uma B-spline é definida analiticamente como um conjunto de polinomiais sobre um vetor de números reais, chamado vetor de nós, em ordem não decrescente. Isso, especialmente para curvas, representa uma flexibilidade a mais na manipulação das formas.

Uma spline de ordem k é matematicamente definida como uma polinomial por partes, de grau (k-1), que é C^{k-2} contínua.

As funções base B-spline são de grau limitado pelo número de pontos de controle, mas não totalmente dependente dele. Isto é, o grau de uma B-spline pode ser escolhido na sua formulação e o número de vértices de controle pode ser aumentado sem alteração do grau da superfície resultante. A alteração do grau da curva muda a forma, aproximando-a ou afastando-a da malha de controle.

Possui controle local, pois sua formulação é dedicada a limitar a região de influência de um dado nó ou vértice de controle. Possui muita flexibilidade na alteração da forma. Os dados necessários para sua manipulação são os pontos da malha de controle e a ordem. Entretanto, modificações no vetor de nós

também podem produzir alterações desejáveis na forma, aumentando a possibilidade de interação com o usuário.

Uma B-spline não passa por qualquer dos pontos de controle, o que implica num problema de fronteira que pode ser contornado de diversas formas [Bar 82].

O método combina a facilidade da manipulação polinomial com a suavidade da construção de formas.

Pela sua formulação geral, uma superfície de Bézier passa a ser uma particularização da superfície B-spline [Ro 76], sendo que uma forma pode ser convertida na outra [Bo 81]

Como Bézier, B-spline possui as propriedades de "convex-hull" , de controle da variação e independência dos eixos de coordenadas [Bar 84].

Ainda com a flexibilidade de B-spline, a manipulação de pontos e ordem para re-definir formatos gera uma dificuldade de manipulação, mesmo que computacional. Com o objetivo de contornar tais dados nem sempre intuitivos Barsky [Ba 84] propôs um novo método matemático de representação, apresentado a seguir.

2.3.5 Superfícies Beta-spline

Segundo seu criador, o esquema Beta-splines é baseado em medidas geométricas fundamentais, ao invés de quantidades algébricas abstratas.

Os dois parâmetros que definem a representação por Beta-spline de uma curva são tensão e inclinação. Tais parâmetros podem ser manipulados diretamente pelo usuário, e são mais

intuitivos que a maioria dos dados dos outros métodos, isto é, permitem maior domínio dos resultados.

Uma Beta-spline é uma polinomial cúbica por partes.

O parâmetro inclinação mede a quantidade relativa de assimetria.

O parâmetro tensão mede a quantidade de tensão simétrica aplicada sobre a curva.

Os efeitos para superfícies são de mesma característica, ou seja, para produzir "deformações" na forma, é muitas vezes suficiente manipular os dois parâmetros básicos, ao invés de alterar posição de pontos de controle ou grau da função.

A obtenção da continuidade entre partes vizinhas é discutida em [Goo 86], e para isso é necessária uma generalização do esquema.

Possui controle local. Possui a propriedade de "convex hull" e controle da variação.

De boa perspectiva para computação gráfica, a carência de literatura a respeito impede maior avaliação de determinados requisitos importantes, em especial no que se refere à implementação por computador. Entretanto, alguns resultados já foram obtidos, utilizando um caso especial de Beta-splines, chamado Beta2-splines [Bar 85].

2.3.6 Comparação entre os métodos matemáticos de representação de superfícies

Os métodos de representação convencional possuem um domínio bastante vasto de representação, adequado para muitas aplicações,

como CAD, modelagem de objetos para obtenção de imagens reais, objetivos artísticos e de propaganda, e outras. Entretanto, para qualquer uma destas aplicações, pode-se desejar modelar um sólido que não possui comportamento adequado a tais métodos. Incluem-se nesta classe de objetos aqueles que tem formato "suavizado" que não podem ser obtidos facilmente por combinação de outros sólidos mais simples. Neste contexto o trabalho apresenta uma alternativa de representação, com uso de um método matemático. O objetivo é avaliar seu emprego para representar objetos difíceis de se manipular por outros modelos, e implementar um sistema que realize a modelagem empregando tal método de representação.

Como já foi mencionado, no momento da escolha de um método de representação de superfícies, além das características matemáticas, outros aspectos devem ser considerados, a maioria deles relativos à aplicação.

Dentre os métodos apresentados, o de Coons já representa uma grande vantagem sobre as técnicas convencionais de interpolação, que normalmente necessitam de uma quantidade maior de pontos para definir uma forma. Entretanto, aqueles métodos que manipulam superfícies através de malha de controle são bastante adequados no tocante à flexibilidade de representação e à possibilidade de manipulação interativa de objetos. Além disso, possui uma base de dados bastante simples, que consiste em armazenar a malha de controle numa ordem adequada à construção da superfície. Isto é, uma vez definido o formato final da superfície, basta guardar seus pontos de controle (e, em alguns casos a ordem) para se recuperar as informações sobre a superfície.

Com relação aos aspectos matemáticos, todos os métodos aqui citados apresentam formulações paramétricas, o que admite muitas facilidades. Além disso, permitem que se represente qualquer objeto pela combinação de "pedaços" (patches) que podem ser justapostos.

Quanto ao método de Coons (Hermite), além do controle local, obriga a localização de um conjunto de polinômios adequados à aplicação com os quais se possa obter uma curva diferenciável em grau 2. Ou seja, as condições de contorno do método elevam o custo computacional e pioram a interação, pois a alternativa é determinar as derivadas parciais nos extremos da superfície. De maneira geral, o método é exaustivo do aspecto computacional, se comparado com os outros métodos [Gi 78].

Já Bézier contorna melhor o problema da interação, uma vez que as inclinações nos extremos do "patch" são derivadas do polígono de controle, e não precisam ser obtidas do usuário. A forma final é "orientada" pelo polígono de controle, o que auxilia muito ao usuário intuir a forma resultante. O mesmo se observa com B-spline ou Beta-spline. Os três também possuem a vantagem da forma obtida não possuir ondulações, ou seja, ser feita de forma suavizada. Para Bézier, o grau de continuidade é infinito. O controle da forma, entretanto, é global, ou seja, uma alteração de um ponto de controle provoca modificações em toda a superfície. Além disso, o grau da curva resultante é estritamente dependente do número de pontos de controle. O método B-spline possui vantagens sobre Bézier, já que tem controle local, e grau independente do número de pontos de controle. Também possui maior flexibilidade de definição da forma. O grau de continuidade de

uma superfície B-spline é C^{k-2}, onde k é a menor ordem entre as formulações para os dois parâmetros, mas pode ser diminuída pela duplicação de um nó paramétrico [Ro 76].

Em [Ro 76] é apresentado um conjunto de modificações que o método admite nas suas formas para curvas. Tais alterações podem ser também extendidas para superfícies. Todos os métodos são adequados para utilizações específicas, mas, do ponto de vista matemático, B-spline possui vantagens sobre os métodos anteriores. Quase todas são mantidas em Beta-spline.

Entretanto, a escolha de um método para implementar a representação não recai apenas sobre suas vantagens matemáticas. Muitas vezes a formulação inicial do método deixa em aberto questões importantes para seu uso em computação gráfica. Tais aspectos necessitam ser manipulados sobre a formulação matemática para permitir uma implementação eficiente e flexível.

O método de B-spline também é adequado neste aspecto, já que oferece uma boa cobertura para seu estudo e utilização, e também para permitir adaptação a novas condições de projeto de formas. Por exemplo, manipulações matemáticas são apresentadas [Boo 88], que otimizam sua utilização. Algoritmos como subdivisão [Cat 78] [Do 78] [Co 80] extendem as possibilidades do método, e outros, como ray-tracing [Swe 86], aumentam as possibilidades de aplicação de B-splines. Já Beta-spline apresenta os primeiros resultados [Goo 88] [Bar 86] de sua manipulação. Entretanto, o método B-spline torna um sistema melhor apoiado pela literatura disponível, o que facilita sua utilização e extensão.

Além disso B-spline apresenta um conjunto de aplicações já

desenvolvidas que constatam sua utilidade prática, como por exemplo [Ro 83], que mostra o uso de B-splines em CAD/CAM de projeto de navios, e [Na 88] que mostra sua utilização para animação facial.

Possui também, um conjunto de extensões que indicam esforços para aumento das possibilidades do método, como [Co 87], que apresenta uma técnica de definição de objetos por "sweep" baseado em B-spline e [Wo 87], que mostra uma nova técnica também baseada em B-splines cúbicas, para interpolação de pontos. Portanto, a utilização de B-spline tem se mostrado bastante popular para modelagem superficial.

As vantagens de sua utilização em comparação com os outros métodos, especialmente para uso em CAD, é comprovada.

2.4 Aspectos de implementação de uma técnica de modelagem

A maioria dos livros disponíveis de computação gráfica apresenta a representação de curvas pelos métodos matemáticos [Ne 83] [Fo 84] [Gi 78], e cita sua extensão para superfícies, sem, no entanto, cuidar de aspectos ou algoritmos particulares à manipulação de superfícies pelos mesmos metodos, com algumas exceções em publicações mais recentes como [Ha 88] que apresenta a implementação do método de Bézier.

Assim, como primeiro e mais simples passo para sua generalização, é necessário extender os algoritmos de curvas para superfícies. Na implementação de um sistema computacional, entretanto, vários outros aspectos devem ser considerados.

Alguns destes aspectos, com bibliografia associada, são apresentados a seguir.

2.4.1 Aplicações da Modelagem de Superfícies

O primeiro aspecto a ser considerado diz respeito às aplicações possíveis da modelagem de superfícies, ou seja, porque desenvolver um sistema de modelagem por B-splines, por exemplo.

A utilização dos sistemas de modelagem tem sido ampliada e implementações ou pesquisas recentes incluem: cálculo de propriedade de massa, checagem de interferência, geração de imagens reais. Além dessas aplicações, é de interesse a utilização da modelagem de superfícies para controle numérico, Método de Elementos Finitos, planejamento de processo, cinemática, robótica, e outras aplicações práticas, além de sistemas experimentais.

A aplicação da modelagem de superfícies na definição de sólidos existia mesmo antes de esquemas mais consistentes como CSG e B-reps, por exemplo. Para aplicações como CAD, modelar um sólido unicamente por B-splines ou qualquer dos métodos de descrição superficial insere alguns problemas, como a determinação de fronteiras e silhuetas. Isto gera dificuldades no uso destes métodos no processo de manufatura [Go 88]. A tendência é, portanto, a utilização da modelagem de superfícies em conjunto com técnicas como CSG ou B-reps, que são esquemas coerentes em termos das características que faltam na modelagem analítica. [Go 88] fornece estratégias básicas para executar a combinação entre

os métodos. Tal combinação deve ser apoiada por algum tipo de conversão poliédrica, como a apresentada em [Li 88], que mostra a incorporação de superfícies matemáticas num esquema B-reps.

Os tipos de superfícies a serem incorporados a um sistema de modelagem são diretamente associados à aplicação. Alguns dos métodos mais desejáveis na maioria das aplicações foram apresentados acima.

A seguir se descrevem aspectos fundamentais na implementação da representação por superfícies paramétricas.

2.4.2 Interface e armazenamento de dados

Num sistema de modelagem baseado em métodos de representação de superfícies descritas por pontos de controle, a base de dados fundamental é bastante simples. Basta armazenar a quantidade de pontos de controle e seus valores em coordenadas tri-dimensionais. A partir desses dados simples, é possível reconstruir a superfície modelada. No caso de B-spline um dado a mais é necessário, referente às ordens da superfície na direção de variação dos dois parâmetos.

Com relação à interface, um sistema de modelagem possui dados gráficos como entrada. Além disso, manipula objetos tri-dimensionais. Portanto, uma interface de entrada de dados deve se preocupar em obter e apresentar em periféricos bi-dimensionais os dados espaciais. Da mesma maneira os periféricos de saída possuem apenas duas dimensões. Para todas as aplicações são então necessários os conceitos de Visualização e Perspectiva, amplamente conhecidos [Ne 83] [Fo 84] [Pe 86], e sua

implementação deve ser transparente ao usuário. Além disso, toda a manipulação de objetos necessita ser discretizada para o nível de precisão admitido pelo periférico. Em especial, funções matemáticas se reduzem, ao nível de observação, a pontos consecutivos a serem mostrados em tela.

Vários aspectos devem ser considerados na elaboração de uma interface gráfica de entrada e saída. Tanto aspectos de comunicação homem/máquina [Ma 85] [We 85], quanto aspectos específicos do equipamento disponível (periféricos gráficos ou não gráficos, precisão da imagem obtida, versatilidade de comunicação). O objetivo da interface, para sistemas de modelagem, é deixar ao usuário as preocupações apenas com o objeto sendo modelado, facilitando ao máximo o acesso às demais funções necessárias.

No que se refere à apresentação dos resultados, os métodos usuais de apresentação se superfícies são :

- "lofting" [Ne 83] [Gi 78] [Fo 84]: Para apresentação de um trecho da superfície mostra-se as curvas limitantes na direção de variação de um dos parâmetros. Em seguida, mostra-se várias "linhas" da superfície na direção de variação do outro parâmetro.

- "quadriculado" [Ha 87]: Repete-se o processo do "lofting" para as duas direções de variação paramétrica. Como resultado, se obtem uma "rede" de pontos que indica o formato geral da superfície resultante.

Deve-se então deixar disponível um dos dois esquemas de apresentação de superfície, permitindo variação da precisão quando necessário.

Constantes esforços têm procurado melhorar a apresentação de superfícies, uma vez que, para objetos complexos, os métodos usuais mais confundem que auxiliam, pela mistura de linhas visíveis e invisíveis na observação do objeto sendo modelado. Vários métodos existem para melhorar essa apresentação, onde o objetivo é permitir um melhor reconhecimento, por parte do usuário, do objeto que está sendo elaborado. A seguir é fornecida uma visão geral dos algoritmos disponíveis para apresentação.

2.4.3 Apresentação de Superfícies Definidas Parametricamente

Uma das importantes tarefas num sistema de modelagem, qualquer que seja a aplicação a que é dedicado é a de apresentar os resultados de maneira fácil de ser entendida, e principalmente que reflita com clareza o objeto sendo modelado. Estas são as características mínimas de uma forma de apresentação de objetos.

No que se refere a objetos definidos matematicamente, são várias as técnicas de apresentação verificadas na literatura, conforme o nível de precisão de imagem desejada para a aplicação.

A maioria dos algoritmos de geração de superfícies definidas parametricamente pode ser classificada de acordo com a diretriz pela qual realiza os cálculos, como:

- algoritmos baseados no espaço paramétrico: Realizam os cálculos no espaço dos parâmetros, pela determinação da visibilidade da curva/superfície como extensão das técnicas de visibilidade para polígonos. Normalmente executam uma aproximação da superfície em regiões poligonais planares, o que é coerente, uma vez que os dispositivos de saída exigem necessariamente

uma discretização. Via de regra, tais algoritmos utilizam grande espaço de memória para dados, e são custosos computacionalmente. Pertencem a essa classe os algoritmos apresentados em [Gr 75] [Gr 78] [Oh 83].

- algoritmos de varredura ("scan-line algorithms") : é caracterizado pela ordem em que os elementos de imagem da figura são gerados na tela: a ordem é da esquerda para a direita e de cima para baixo, como uma varredura de televisão. Por trabalhar no espaço da imagem, necessita de inversão da função usada para definir a superfície, ou seja, dado um ponto na imagem, descobrir a qual ponto no espaço ele corresponde. [La 80] apresenta vários algoritmos de geração de imagem ("rendering"), aqui classificados. Outros também são descritos em [Sc 82] e [Gr 84].

- algoritmos mistos: Estão aqui incluídos algoritmos que não se pode classificar em nenhum dos outros dois itens anteriores, e que se referem a métodos do tipo varredura, mas que realizam alguns cálculos no espaço paramétrico, com o objetivo de evitar a inversão de funções, que é computacionalmente custosa. Aqui podem ser incluídos os métodos enumerados em [Pu 87] e [Rock 87].

- "ray-tracing" : Representa a extensão, para superfícies definidas matematicamente, de uma técnica já clássica na computação gráfica para geração de imagens de objetos tri-dimensionais. À parte da classificação anterior, o cálculo de ray-trace de formas paramétricas proporciona a apresentação de imagens de alta qualidade com grande realismo [Ka 82] [Swe 86]. Esses algoritmos empregam boa parte dos algoritmos de manipulação de superfícies.

Alguns dos métodos aqui citados, buscam imagens que permitam melhor reconhecimento da forma pela eliminação de linhas/superfícies escondidas [Gr 78] [Oh 83]. Outros buscam maior apuro da imagem, pela geração da superfície coberta, com sombreamento [Fo 79] [Gr 84] [La 80] [Sc 82], ou mesmo de imagens realísticas [Pu 87] [Rock 87], que se preocupa com iluminação, cor, cortes e anti-aliasing.

O tempo de processamento dos algoritmos de apresentação de imagens é uma preocupação de vários anos, que trata inclusive do desenvolvimento de hardware dedicado a aplicações gráficas [For 79]. Com o aumento da velocidade do hardware, a geração de imagens precisas num curto intervalo de tempo vem crescendo. Mas, de uma maneira geral, ainda é excessivamente lento produzir uma imagem do objeto com alto grau de realismo para um processo de projeto interativo. A apresentação por "quadriculado" é, portanto, um recurso bastante razoável para aplicações como CAD, por exemplo, cujo objetivo é a forma.

Como o equipamento dedicado é caro e, portanto, não universal, isso não descarta (ao contrário, incentiva) a otimização dos algoritmos existentes, pela criação de novas estratégias, como [Oh 83], que otimiza o algoritmo dado em [Gr 78], ou novos algoritmos, como [Sc 82]. Também se justificam algoritmos específicos, como [Roc 87] que manipula "pedaços" de superfícies de Bézier, ou [Swe 86] que gera superfícies B-spline.

Como a definição matemática é contínua, uma discretização é sempre necessária. Assim, os algoritmos de subdivisão de superfícies [Co 80], podem auxiliar em muitos algoritmos de apresentação, quadriculando ou triangularizando a superfície.

2.5 Uso de técnica no desenvolvimento do software

Independente dos algoritmos que um sistema de modelagem utiliza, o desenvolvimento do software é trabalhoso, e normalmente resulta em sistemas de grande porte.

Embora não seja usual a literatura comentar esse aspecto, o porte do software gráfico e sua complexidade habitual fazem com que se justifique o uso dos conceitos e técnicas atuais da Engenharia de Software no desenvolvimento e implementação de qualquer sistema gráfico, não apenas sistemas de modelagem de superfícies.

Em especial, as técnicas apoiadas em desenvolvimento descendente (top-down) podem ser úteis na confecção de software gráfico. Consiste basicamente em, partindo de uma especificação bem elaborada, considerar o problema como uma caixa preta, que é sucessivamente refinado em módulos, até as funções de mais baixo nível, que possam ser prontamente implementadas.

Técnicas apoiadas nesse conceito são a Análise Estruturada [Ga 83], o Projeto Estruturado [Pa 88], e a Programação Estruturada, esta última bastante difundida no estudo da computação procedural.

2.6 Considerações Finais

A aplicação de um sistema de modelagem possui um conjunto muito grande de aspectos a se considerar. Em especial, a aplicação, as estruturas matemáticas e algoritmicas disponíveis,

o equipamento objetivo e o domínio dos objetos que se quer representar, são os primeiros aspectos que limitam e direcionam o desenvolvimento de um programa que manipule a modelagem por computador.

Entre as técnicas de modelagem por superfícies matemática, objetivo do trabalho, destaca-se as vantagens do método B-spline para ajuste de curvas/superfícies.

Além disso, um sistema construído para modelar objetos por B-splines tem a mesma estrutura da modelagem por Bézier, isto é, um sistema pode opcionalmente trabalhar com qualquer um deles, uma vez que os dados de entrada e o tipo de saída obtida são semelhantes. Neste sentido, o próximo capítulo visa apresentar os aspectos relacionados com os modelos matemáticos, e que direcionam sua implementação.

CAPÍTULO III
SOBRE A TÉCNICA DE MODELAGEM POR SPLINES

3.1 Considerações iniciais

Como foi apresentado no capítulo II, a escolha de um método de representação recai sobre uma série de aspectos, especialmente ligados ao método e à aplicação.

Neste trabalho, que busca implementar um modelo de representação por definição de superfícies matemáticas, são considerados os aspectos básicos de construção de um modelador que trabalhe com tais superfícies. Em termos de aplicação, a meta é deixar a ferramenta disponível para que possa ser adaptada às várias aplicações onde seja de interesse.

Este capítulo procura descrever os conceitos, teoria e algoritmos que fundamentaram a construção do sistema.

Os métodos paramétricos de definição de superfícies são bastante úteis, principalmente pela independência dos eixos de coordenadas e pela facilidade de avaliação de pontos da superfície. Dentre eles, os que trabalham com malhas de controle são os mais recomendados, e também mais populares em computação gráfica, devido à grande flexibilidade para o projeto da forma do objeto.

Dos dois métodos mais empregados, Bezier e B-spline, este último apresenta, de acordo com as características citadas no capítulo anterior, maior vantagem, em especial pela escolha de

ordem (grau da superfície), controle local, e versatilidade de projeto. O nosso sistema foi então desenvolvido com o objetivo de aplicar B-spline ao projeto superficial de sólidos tri-dimensionais.

Muitos aspectos são importantes na construção de um sistema de modelagem por superfícies paramétricas. Em primeiro lugar, a formulação e a avaliação do método em si. Os dois próximos itens no capítulo cuidam destes aspectos.

Outro aspecto importante é a interface com o usuário, em termos de entrada de dados e também de visualização dos resultados. Esse aspecto de interface com o ambiente é um cuidado especial deste trabalho.

Com relação à apresentação da superfície resultante, dois métodos foram implementados. Um deles, de manipulação relativamente simples e execução rápida possui limitações quanto ao tipo de apresentação e tipo de objetos que é capaz de gerar. O outro foi implementado com base num algoritmo divulgado na literatura [Oh 83], e foi escolhido por se adaptar bem ao objetivo do sistema, e também por transmitir uma quantidade de informações satisfatória sobre o procedimento empregado. O algoritmo deve facilitar a observação do formato real do objeto que está sendo projetado, sem ambiguidades e com precisão possível de ser controlada. Além disso, não é objetivo a obtenção de "imagens reais", isto é, aspectos como cor e iluminação não são tratados neste trabalho, mas podem ser incluídos a nível de extensão do sistema. Além disso, outro aspecto que norteou a escolha foi a generalidade, isto é, o método empregado não é específico do método de B-spline, o que ocorria com vários outros

métodos investigados. Qualquer superfície matemática pode ser apresentada pelo método escolhido. O item 3.4 apresenta os métodos utilizados para geração da imagem da superfície ("rendering").

O item 3.4 apresenta a avaliação de uma superfície, segundo o método de Bézier.

O item 3.5 discute aspectos da interface com o sistema de modelagem, que necessita manipular dados tri-dimensionais e por isso é mais elaborada, além do fato de que deve tornar disponíveis as facilidades do método de modelagem ao usuário.

O item 3.6 discute o desenvolvimento de sistemas do ponto de vista da Engenharia de Software, no que diz respeito aos conceitos empregados no desenvolvimento do software aqui apresentado. Dois dos aspectos discutidos são o de flexibilidade e o de generalidade, que, tendo sido observados durante a construção do sistema, permitiram a possibilidade de inclusão imediata da representação de Bézier, num sistema originalmente desenvolvido para B-splines.

Por isso, o item 3.2.4 apresenta a formulação do método de Bézier, cuja implementação é facilmente apoiada pela estrutura do sistema de modelagem.

3.2 Definição do método de representação matemática

A base da representação por modelo matemático de superfícies é a formulação analítica do método utilizado para modelar os objetos. O propósito desse item é discutir tal formulação e suas

características para o método B-spline, no qual foi centralizado o desenvolvimento do trabalho.

A formulação matemática de todas as polinomiais apresentadas anteriormente como métodos de representação superficial é dada na forma paramétrica, que é definida a seguir.

3.2.1 Formulação paramétrica de curvas/superfícies

Sob muitos aspectos a formulação paramétrica de curvas e superfícies possui vantagens. Por esse motivo, toda manipulação de métodos matemáticos de representação se dá pelo uso da forma paramétrica. Para curvas, a representação paramétrica é dada na forma:

$$x = x(u)$$
$$y = y(u) \qquad u_1 <= u <= u_2$$
$$z = z(u)$$

Pode-se, assim, avaliar toda a curva variando o valor do parâmetro u entre seus dois valores extremos u_1 e u_2.

As técnicas aqui apresentadas, calculam sua forma paramétrica pela combinação linear de funções-base ("blending") e determinação dos coeficientes dessa combinação linear utilizando um conjunto de pontos de controle, pertencentes ou não à curva resultante, e que formam o chamado polígono de controle. O formato do polígono de controle é responsável por modelar a forma da curva resultante (figura 3.1). A equação geral de uma curva

definida parametricamente em função do parâmetro u é dada por:

$$Q_n(u) = \sum_{i=0}^{n} F_i(u)\, V_i$$

onde:

n+1 é o número dos vértices do polígono de controle,

V_i são os vértices de controle ,

F_i são as funções-base específicas do método sendo utilizado

Q_n é uma equação vetorial, ou seja, possui uma componente em x, uma em y e uma em z.

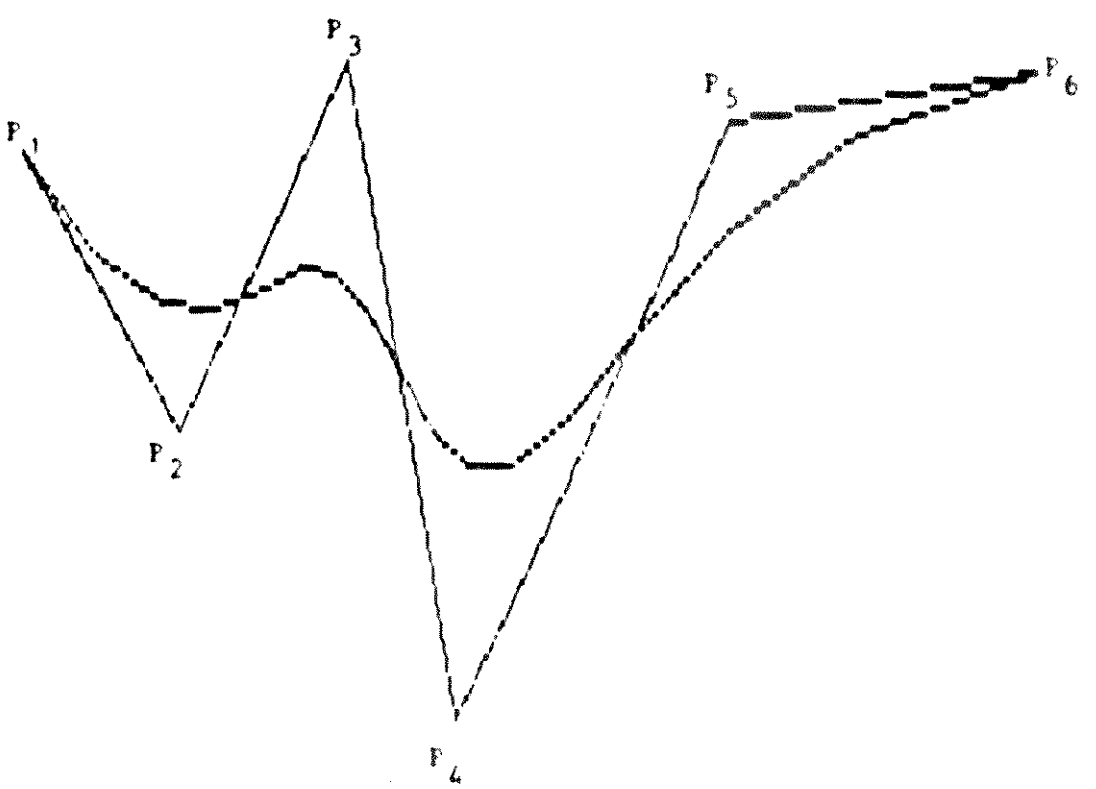


Fig. 3.1 polígono de controle de uma curva e curva resultante do polígono

Para superfícies, o que normalmente ocorre é uma generalização do processo usado para curvas. Os pontos de controle formam uma malha de controle, que é uma rede de pontos que modelam o formato final da superfície (figura 3.2). A formulação matemática de uma superfície para um dado método é obtida pelo produto tensor de duas curvas definidas pelo mesmo método.

A equação geral para superfícies é obtida em função de dois parâmetros, u e v e é dada por:

$$Q_{n,m}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} F_i(u)\, F_j(v)\, V_{i,j}$$

onde:

n + 1 é o número de pontos de controle na direção de variação do parâmetro u,

m + 1 é o número de pontos de controle na direção de variação do parâmetro v,

F_i e F_j são as funções-base B-spline, aplicadas respectivamente nas direções paramétricas u e v, e

$V_{i,j}$ são os pontos da malha de controle, na forma matricial.

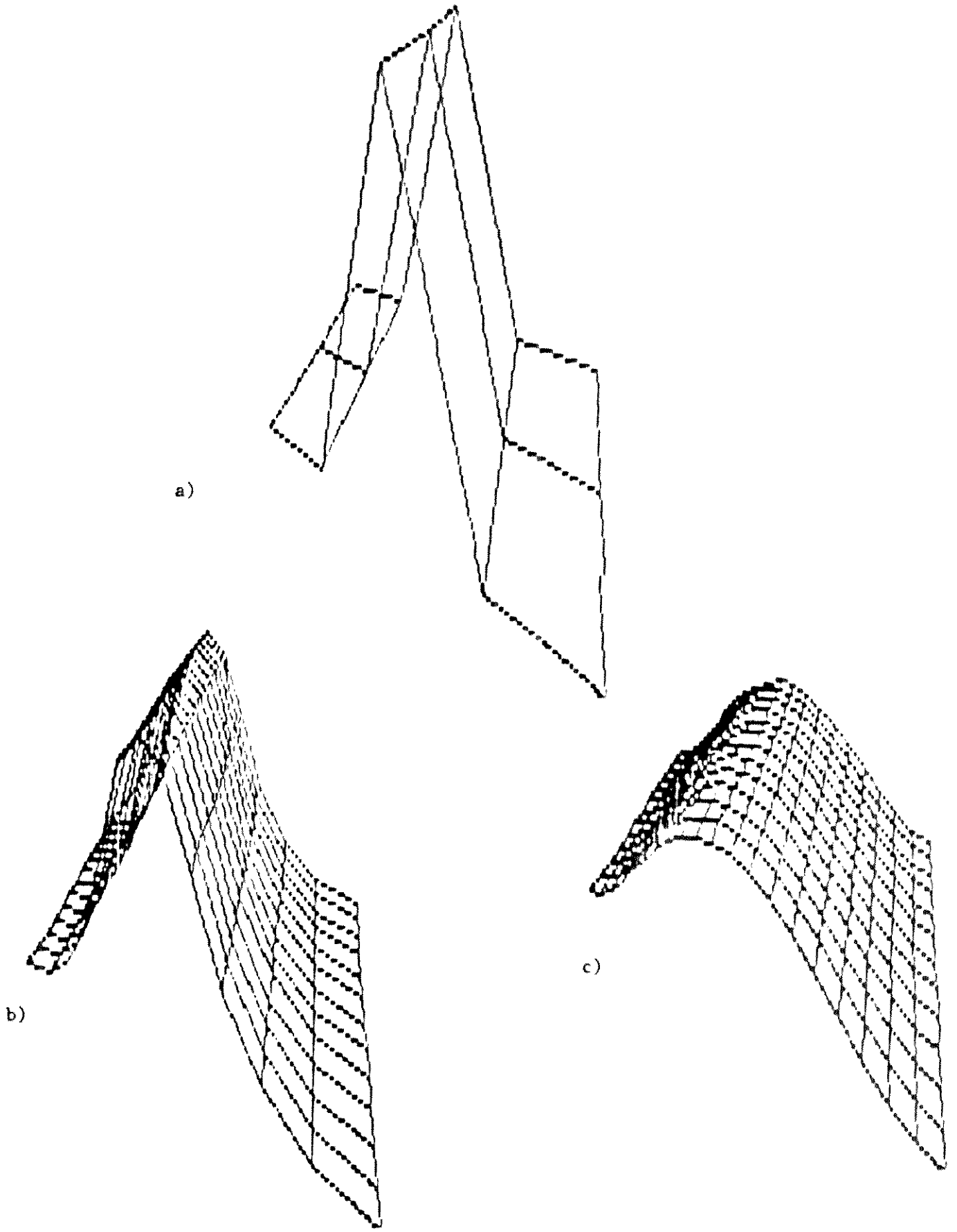


Fig. 3.2a Malha de controle de uma superfície.
b Superfície B-spline correspondente à malha da fig.3.2a, k = l = 3
c Superfície B-spline correspondente à malha da fig. 3.2a, k = 5, l = 3

Os vários métodos para representação de formas se diferenciam no que diz respeito ao tipo de funções-base utilizadas, que no caso desse trabalho , são funções polinomiais. A formulação para B-splines é fornecida a seguir:

3.2.2 Formulação do método B-splines

Uma B-spline de ordem k é formulada analiticamente como uma polinomial por partes, de grau k-1, definida sobre um vetor de nós. Um vetor de nós é um vetor de número reais quaisquer em ordem não- decrescente, ou seja:

$[u_0, u_1, \ldots, u_m]$ tal que $u_{i-1} \leq u_i$, $i = 1, \ldots, q$

Além do vetor de nós, uma B-spline é obtida como função dos pontos de controle e da ordem da curva na direção de variação do parâmetro.

Assim, a i-ésima função-base B-spline de ordem k (grau k-1), para o vetor de nós $[u_i, \ldots, u_{i+k}]$ será denotada por $N_{i,k}(u)$ e é definida pela seguinte relação recursiva:

$$N_{i,k}(u) = \frac{u - u_i}{u_{i+k-1} - u_i} N_{i,k-1}(u) + \frac{u_{i+k} - u}{u_{i+k} - u_{i+1}} N_{i+1,k-1}(u)$$

e

$$N_{i,1}(u) = \begin{cases} 1 & \text{se } u_i \leq u < u_{i+1} \\ 0 & \text{caso contrário} \end{cases} \qquad \text{(eq. 3.1)}$$

Pela formulação do método, observa-se:

- a ordem da função-base (e, portanto, seu grau) pode ser escolhida, desde que seja menor ou igual ao número de pontos de controle.

- A computação de $N_{i,k}(u)$ envolve todos os nós de u_i a u_{i+k}, mas não outros, e então a amplitude de sua influência (ou apoio) na construção da curva é a de k intervalos. Essa característica habilita o controle local da curva/superfície. O controle local implica em, modificando a posição de um ponto de controle da curva/superfície, apenas uma região limitada ao redor do ponto é alterada (figura 3.3). Essa é uma vantagem, no sentido de que permite ao usuário maior domínio do efeito de uma modificação aplicada à malha.

- As funções $N_{i,k}(u)$ formam realmente uma base. Assim, qualquer spline de ordem k ou menor, definida sobre um dado vetor de nós, pode ser escrita como combinação linear de funções base B-spline, definidas sobre o mesmo vetor de nós, extendido em ambos os extremos por k-1 nós arbitrários.

- A repetição de um ponto de controle mais de uma vez no polígono (malha) implica numa redução da continuidade da curva naquele ponto, o que é mais uma flexibilidade de projeto admitida pelo método (figura 3.4).

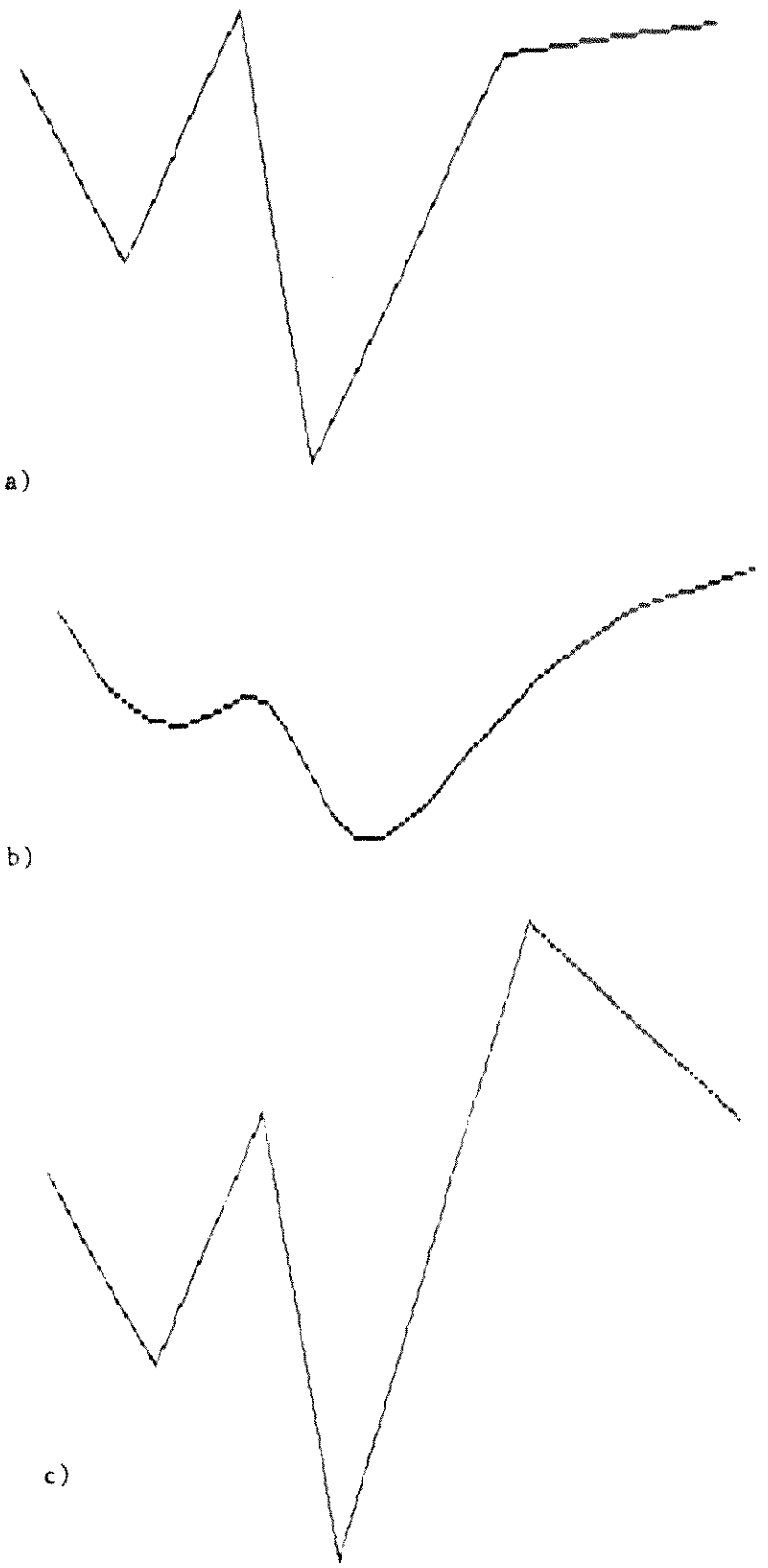
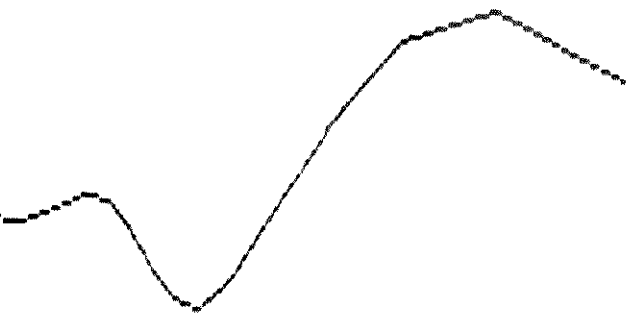
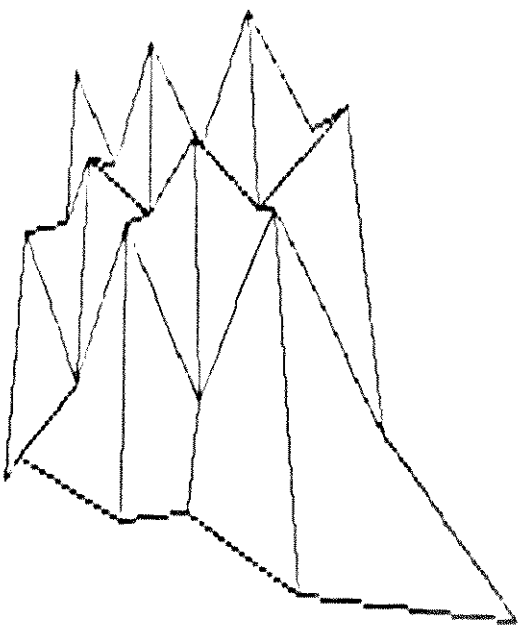


Fig. 3.3a polígono de controle
b curva resultante do polígono da fig. 3.3a
c polígono de 3.3a com um ponto de controle alterado

d)



e)



f)

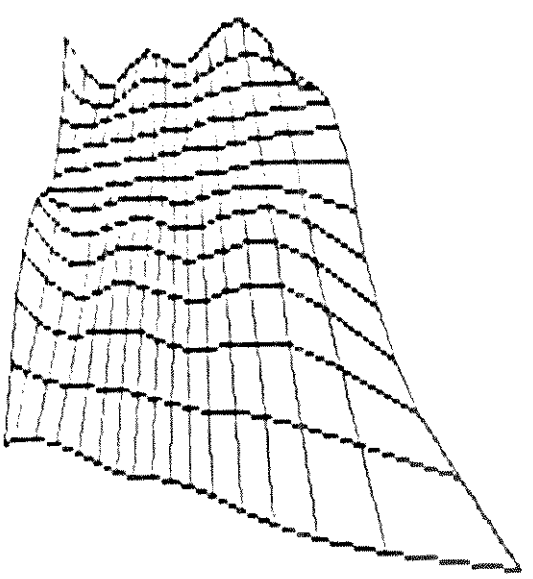
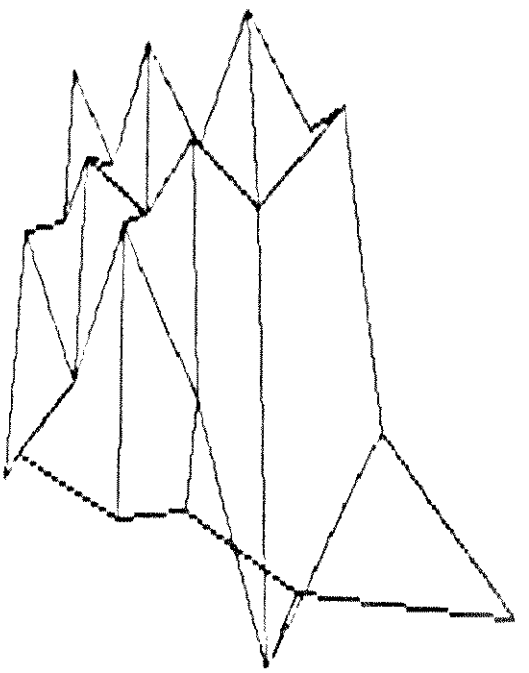
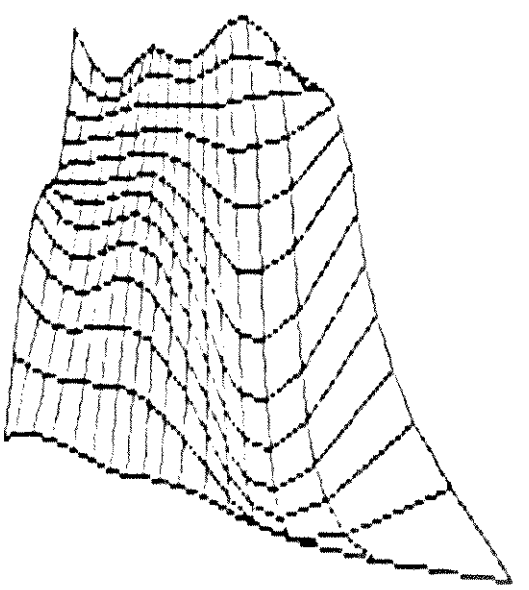


Fig. 3.3(continuação)
d curva resultante do polígono de controle modificado.
e malha de controle para superfície
f superfície resultante do polígono da fig. 3.3e



g)



h)

Fig. 3.3 (continuação)
g malha de controle da fig. 3.3e modificada
h superfície resultante da malha de controle modificada

Com as funções-base B-spline, a equação vetorial que expressa a curva, fica:

$$Q_i(u) = \sum_{i=0}^{n} N_{i,k}(u)\, V_i$$

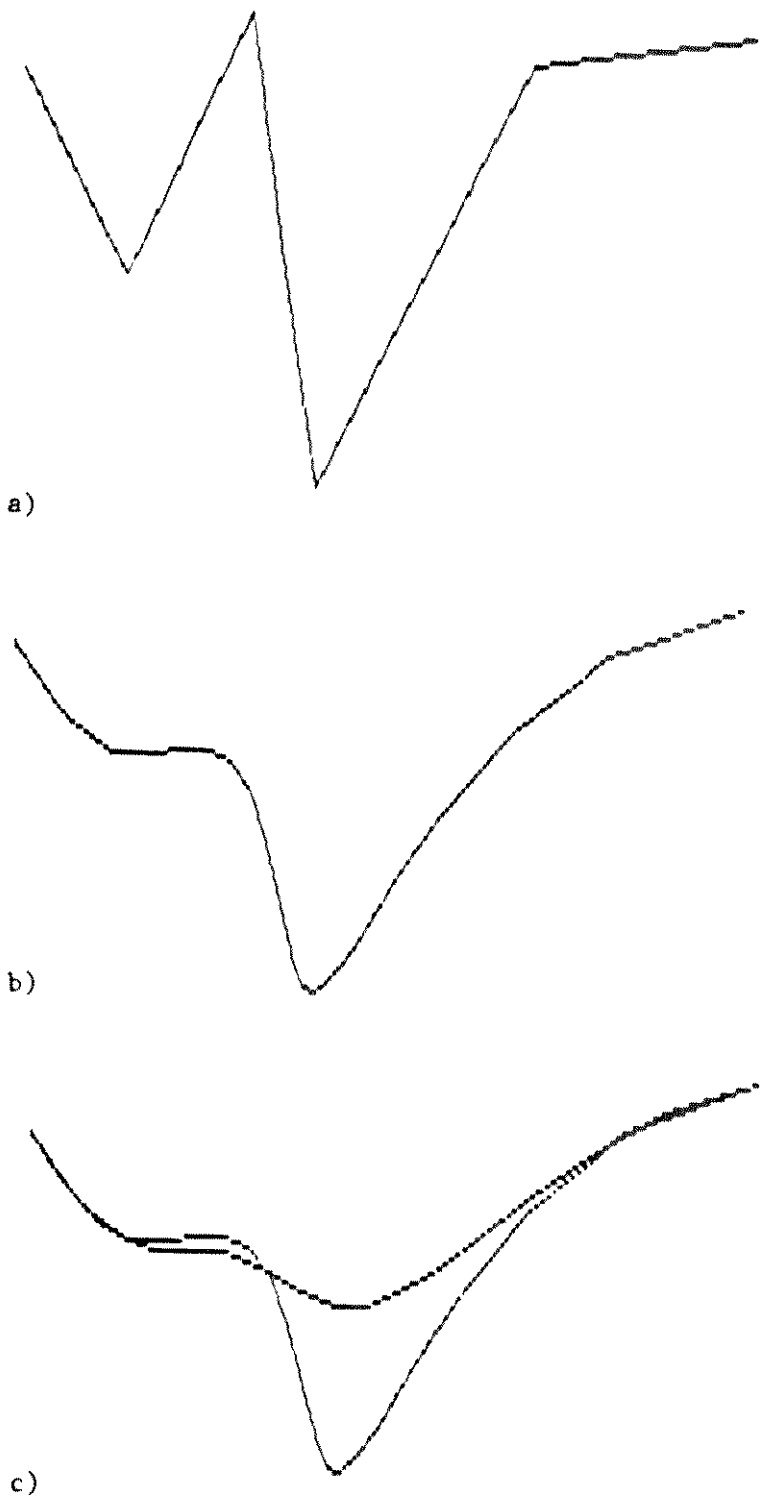


Fig. 3.4a polígono de controle da fig 3.1a, com repetição do vértice de controle.
b curva resultante da repetição do vértice.
c curvas comparadas (3.1b e 3.4b)

Para superfícies, a equação é generalizada pelo produto tensor de duas curvas B-splines, com a formulação:

$$Q_{k,l}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,k}(u)\, N_{j,l}(v)\, V_{i,j}$$

As propriedades das curvas B-spline são perfeitamente extensíveis à formulação para superfícies.

A formulação matemática B-spline possui várias características que fornecem flexibilidade ao projeto da forma de curvas ou superfícies, entre elas a definição da ordem, e, portanto, do grau da curva/superfície (figuras 3.2b e 3.2c), o controle local da forma, como já mencionado, e a definição do vetor de nós.

O vetor de nós possui variações bastante úteis na definição de splines de uso geral. Sua obtenção, bem como suas características, são apresentadas a seguir.

3.2.3 Determinação do vetor de nós B-spline

Além da ordem crescente, a única restrição sobre o vetor de nós da formulação B-spline é que cada nó não pode aparecer mais de k vezes no vetor. Apesar disso, há vários casos especiais de vetor de nós, bastante úteis para a maioria das aplicações.

Um tipo de vetor é o de espaçamento uniforme periódico. Nele, $u_i = i$.

Para que haja k funções base não zero nos valores extremos de u (por razões de consistência), uma versão ligeiramente modificada é comumente usada. O vetor de nós com espaçamento

uniforme é acrescido da repetição do primeiro e último valores do vetor, k vezes. Esse tipo de vetor de nós é denominado espaçamento uniforme não periódico.

Cada vértice do polígono de controle tem uma correspondente função-base, e há n+1 vértices de controle em cada direção de variação do parâmetro. Portanto, há n+1 funções-base combinadas. Percorrendo o vetor de nós, cada função base é não zero sobre um conjunto sucessivo de k+1 nós. Assim, k+n+1 nós definem n+1 funções base que correspondem aos n+1 vértices de controle. Então, para o espaçamento uniforme não periódico, o vetor fica:

$$u_i = \begin{cases} 0 & i = 0, \ldots, k-2 \\ i-k+1 & i = k-1, \ldots, n+1 \\ n-k+2 & i = n+2, \ldots, n+k \end{cases}$$

cujo resultado é:

$$[\underbrace{0\ 0\ \ldots\ 0}_{k-1}\ \underbrace{0\ 1\ \ldots\ r-1\ r}_{n-k+3}\ \underbrace{r\ r\ \ldots\ r}_{k-1}]$$

Nesse caso, o intervalo de variação do parâmetro u para cada direção é dado por: $0 \leq u < n-k+2$.

A manipulação do vetor de nós é mais uma flexibilidade de projeto do método.

Em outros tipos de espaçamento, quando um nó ocorre mais de uma vez, ele é chamado um nó múltiplo. Se um nó tem multiplicidade M então a continuidade da curva neste nó é reduzida de M-1. Esta propriedade facilita a introdução voluntária de descontinuidade da curva, aumentando as opções de projeto de forma.

A avaliação de uma superfície B-spline envolve, portanto, o cálculo do vetor de nós, das funções base e a definição dos pontos de controle.

A versatilidade do sistema, no entanto, pode permitir a avaliação de superfícies definidas pelos mesmos dados por mais de um método de representação. Um opção é o método de Bézier, também amplamente usado em PAC, e que é definido a seguir.

3.2.4 Formulação do método de Bézier para representação de superfícies.

Como B-spline, o método de Bézier para superfícies trabalha com vértices de controle organizados numa malha, como uma extensão de sua formulação para curvas. A diferença, além do vetor de nós que não existe no método de Bezier, é a definição das funções de "blending", cuja formulação é dada por:

$$B_{i,m}(u) = \binom{m}{i} u^i (1-u)^{m-i} \qquad \text{onde:} \quad \binom{m}{i} = \frac{m!}{i!\,(m-i)!}$$

$B_{i,m}(u)$: probabilidade de exatamente i sucessos em m tentativas, cada uma com probabilidade u.

Assim, as funções-base do método de Bézier são as polinomiais de Bernstein, que representam a distribuição binomial de probabilidades, e formam uma base no espaço vetorial dos polinômios de grau menor ou igual a m.

Então, um ponto na superfície é uma média ponderada dos vértices de controle da malha, dada por:

$$Q_{n,m}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_{i,n}(u)\, B_{j,m}(v)\, V_{i,j}$$

Bézier representa uma alternativa de representação, possível de ser incorporada ao sistema desenvolvido sem alteração da estrutura do mesmo. Devido a esse fato, muito embora a escolha para implementação tivesse recaído sobre o método B-spline, Bézier também foi incorporada e, portanto, discussão prossegue com os aspectos de avaliação de superfícies por ambos os métodos.

3.3 Avaliação de uma superfície B-spline

3.3.1 Obtenção dos pontos paramétricos da superfície

Para a obtenção de valores dos pontos de uma superfície B-spline, é necessário avaliar sucessivamente a expressão vetorial que a define, e que é explicitamente dada por:

$$x_{k,3}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,k}(u)\, N_{j,3}(v)\, Vx_{i,j}$$

$$y_{k,3}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,k}(u)\, N_{j,3}(v)\, Vy_{i,j} \qquad \text{(eq. 3.2)}$$

$$z_{k,3}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,j}(u)\, N_{j,3}(v)\, Vz_{i,j} \qquad \text{, onde:}$$

N - função de "blending" B-spline

Vx , Vy , Vz - coordenadas x, y e z do vértice V

Numericamente, para obter coordenadas de pontos da superfície, é necessário aplicar valores consecutivos dos parâmetros u e v nas equações acima. Como é impossível avaliar todos os pontos de uma superfície, isso é feito a partir de alguma discretização, dependente do objetivo para o qual se deseja avaliar a equação.

A avaliação da fórmula envolve a definição prévia de pontos de controle, que devem estar organizados matricialmente. Além disso, envolve também a determinação dos valores do vetor de nós e a opção pelo grau da curva.

A determinação de pontos sobre a superfície implica em fornecer valores dos parâmetros e obter como resultado coordenadas tri-dimensionais (x, y e z) do ponto correspondente a tais parâmetros , de acordo com a aplicação da equação 3.2.

O algoritmo de avaliação para curvas é dado na literatura e é apresentado na figura 3.5. A extensão da fórmula para superfícies é imediata, sendo que o algoritmo de tal extensão foi desenvolvido com relativa facilidade.

```
FUNCTION knot(i:INTEGER) INTEGER;
{Determina o i-esimo componente do vetor de nos B-spline}

begin
   if i<knot1
      then knot:=0
   else
      if i>knotn then
         knot:=knotn-knot1+2
      else
         knot:=i-knot1+1;
end;


FUNCTION Nblend (i,k:INTEGER;u:REAL;n:integer) REAL;
{Determina  o  valor  da i-esima funcao de blending  B-spline  de
ordem k, para o parametro u}

var t:INTEGER;
    v:REAL;

begin
   if k = 1 then
   begin
      v:=0;
      if (knot(i)<=u ) and (u<knot(i+1)) then v:=1
   end
   else
   begin
      v:=0;
      t:=knot(i+k-1)-knot(i);
      if t<>0 then v:=(u-knot(i))*Nblend(i,k-1,u,n)/t;
      t:=knot(i+k)-Knot(i+1);
      if t<>0 then
         v:= v+(knot(i+k)-u)*Nblend(i+1,k-1,u,n)/t
   end;
   Nblend:=v;
end;



PROCEDURE Bspl_curva(VAR x,y,z:REAL;w REAL;n,k INTEGER;p :array),
{Determina um ponto de uma curva definida por B-spline, com o valor
 do parametro w e com os pontos de controle do vetor p}

var
   i:
     INTEGER,
   b:                          {armazena o produto das funcoes de blending}
      REAL;

begin
   if w=0 then
   begin
      x:=p[0,1];y:=p[0,2];z:=p[0,3];
   end
   else
      if trunc(w)=n-k+2 then
      begin
         x:=P[n,1];y:=p[n,2];z:=p[n,3]
      end
      else
      begin
         knot1:=k;knotn:=n;
         x:=0 0;y:=0 0;z:=0 0;
         for i:=0 to n do
         begin
            b:=Nblend(i,k,w,n);
            x:=x+p[i,1]*b;y:=y+p[i,2]*b;z:=z+p[i,3]*b;
         end
      end;
end
```

Fig. 3.5 Trecho de programa Pascal que avalia um ponto de uma curva B-spline. fonte: [Ne 83]

Para avaliar uma curva ou superfície B-spline, é necessário notar que ela não é definida na fronteira de variação do parâmetro (equação 3.1). Para isso, uma decisão deve ser tomada quanto ao que fazer para definir a fronteira de um pedaço de superfície B-spline. Na abordagem adotada neste trabalho, busca-se continuidade no final do pedaço da superfície sendo representado. Para isso, determina-se como fronteira da superfície B-spline, a curva B-spline definida pelo vetor de nós que limita a malha, nos quatro extremos.

A figura 3.6 mostra o algoritmo para a avaliação de uma superfície B-spline, considerando o problema de fronteira.

```
inicio
  obtem limites dos parâmetros (limit_u e limit_v)
  obtem passo de variação dos parâmetros (passo_u e passo_v)
  u <-- 0.0
  enquanto u < limit_u faça
    v <-- 0.0
    Obtém ponto de fronteira na curva P(u)
    Obtém ponto projetado
    Registra ponto
    enquanto v < limit_v faça
      spline_sup (p,u,v,k,l,n,m,x,y,z)
      Obtém ponto projetado
      Registra ponto
      v <-- v + passo_v
    fim enquanto
    Obtém ponto de fronteira final na curva P(u)
    Obtém ponto projetado
    Registra ponto
    u <-- u + passo_u
  fim enquanto
  Obtem curva limitante na direção v - P(v)
fim

Algoritmo spline_sup  para a superfície B-spline
nota: Nblend é a função de "blending" B-spline

inicio
  x <-- 0.0 , y <-- 0.0, z <-- 0.0
  para i <-- 0 ate n faça
    knotk <-- k, knotn <-- n
    b1 <-- Nblend(i,k,u,n)
    para j <-- 0 ate m faça
      knotk <-- l, knotn <-- m
      b2 <-- Nblend (j,l,v,m)
      b <-- b1 * b2
      x <-- x + b * p[i,j,1]
      y <-- y + b * p[i,j,2]
      z <-- z + b * p[i,j,3]
    fim para (j)
  fim para (i)
fim
```

Fig. 3.6 Algoritmo da avaliação de um quadriculado de pontos sobre uma superfície B-spline

3.3.2 Otimização da Avaliação de uma Superfície B-spline.

A construção de superfícies, portanto, envolve avaliação de polinômios, o que é um processo custoso computacionalmente, por empregar operações sucessivas de exponenciação, multiplicação e soma em grande número. Assim, algumas técnicas existem para simplificar a complexidade computacional (medida de eficiência) do cálculo de polinômios.

Durante a modelagem do objeto, o que costuma ocorrer é a alteração sucessiva de um objeto pelo usuário até uma versão final satisfatória, provocando uma re-avaliação da superfície a cada modificação.

Quando uma superfície é avaliada pela primeira vez, é inevitável que necessite ser calculada para todos os pontos paramétricos determinados pela precisão escolhida. Numa posterior edição, entretanto, se o usuário modificar apenas posição de pontos de controle, pode-se proceder a re-avaliação re-calculando a região da superfície afetada, pois parte da superfície permanece inalterada pela modificação (fig. 3.3), dada a característica de localidade do método matemático, exposta no item 3.2.2.

Além da característica que permite re-avaliação parcial de uma superfície B-spline, ela admite o cálculo incremental, ou seja, o novo valor dos pontos afetados pode ser determinado a partir do valor dos próprios pontos antes da alteração, ou seja:

Tendo sido modificada a posição do ponto de índices r,s, tem-se:

$$Q_{novo}(u,v) = Q_{anterior}(u,v) + (Vrs_{novo} - Vrs_{anterior}) * \left(\sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,k}(u) * M_{j,l}(v) \right)$$

Desta forma, pode-se re-avaliar a superfície depois de uma modificação utilizando um menor tempo de processamento, re-calculando a influência, no formato da superfície, provocada pela modificação de um único ponto, agilizando a avaliação. No entanto, o cálculo incremental não é válido se o usuário desejar alterar também o número de pontos de controle, ou a ordem da superfície, ou mesmo o vetor de nós.

Com base nesse mesmo critério de localidade de influência de um ponto de controle, é possível determinar que região da superfície pode ser alterada pela alteração de um único ponto. Com isso, admite-se utilizar apenas parte dos pontos de controle, dependendo da região de cálculo dos parâmetros.

3.4 Avaliação de uma superfície Bézier

Como no método B-spline, a equação que formula o método Bézier é vetorial, isto é, necessita ser avaliada para cada uma das coordenadas (x,y,z) dos pontos da superfície.

Qualquer objetivo da obtenção de pontos sobre a superfície envolve o cálculo de valores da distribuição binomial de probabilidades, que os polinômios de Bernstein empregam, para valores consecutivos dos parâmetros.

O grau da superfície de Bézier é função do número de pontos

da malha de controle.

O processo de obtenção de um ponto sobre uma superfície de Bézier é uma generalização do processo de obtenção de pontos da curva, uma vez que as duas formulações (curva e superfície) envolvem a avaliação da função de "blending", cuja variável é o valor de um parâmetro.

O trecho de programa para o cálculo da função de "blending" de Bézier e para a obtenção de um ponto sobre a curva é dado na figura 3.7.

```
FUNCTION C(n,i: INTEGER): INTEGER;
{Determina a combinação de n, i a i}

var
  j,a: INTEGER;

begin
  a := 1;
  for j := i+1 to n do a:=a*j;
  for j := 1 to n-i do a := a div j;
  c := a;
end;

FUNCTION Bblend (i,n: INTEGER; u: REAL): REAL;
{Determina o valor da i-ésima função de Blending de Bézier de
 ordem n, para o valor do parâmetro u}

var
  j: INTEGER;
  v: REAL;

begin
  v := C(n,i);
  for j:= 1 to i do v := v*u;
  for j := 1 to n-i do v := v*(1-u);
  Bblend := v;
end;

PROCEDURE Bspl_curva(VAR x,y,z: REAL; w: REAL; n,k: INTEGER; p: XYZarray);
{Determina um ponto de uma curva definida por Bézier, com o valor
 do parametro w e com os pontos de controle do vetor p}

var
   i:
     INTEGER;
   b:                    {armazena o produto das funcoes de blending}
     REAL;

begin
  x:=0.0; y:=0.0; z:=0.0;
  for i:=0 to n do
  begin
    b:=Bblend(i,n,w);
    x:=x+p[i,1]*b; y:=y+p[i,2]*b; z:=z+p[i,3]*b;
  end
end;
```

Fig. 3.7 Trecho de programa Pascal que gera uma curva de Bézier
Fonte: [Ne 83]

A obtenção da superfície é uma generalização do processo da fig. 3.7, de acordo com a fórmula, para valores dos parâmetros em duas direções. O algoritmo para obtenção de um quadriculado* de pontos sobre uma superfície de Bézier é dado na figura 3.8.

```
inicio
  Obtém passo de variação dos parâmetros (passo_u e passo_v)
  u <-- 0.0
  enquanto u <= 1.0 faça
    v <-- 0.0
    enquanto v <= 1.0 faça
      spline_sup (p,u,v,k,l,n,m,x,y,z)
      obtem ponto projetado
      registra ponto
      v <-- v + passo_v
    fim enquanto
    u <-- u + passo_u
  fim enquanto
fim


Algoritmo spline_sup  para a superficie de Bézier
nota: Bblend é a função de "blending" de Bézier

inicio
  x <-- 0.0 , y <-- 0.0, z <-- 0.0
  para i <-- 0 ate n faça
    b1 <-- Bblend(i,n,u)
    para j <-- 0 ate m faça
      b2 <-- Bblend (j,m,v)
      b <-- b1 * b2
      x <-- x + b * p[i,j,1]
      y <-- y + b * p[i,j,2]
      z <-- z + b * p[i,j,3]
    fim para (j)
  fim para (i)
fim
```

Fig. 3.8 Algoritmo de geração de uma superfície de Bézier.

3.5 Princípios para elaboração da interface

Um aspecto a ser considerado na implementação de um sistema como o que é proposto, é a típica característica interativa de um sistema de modelagem de formas. O projeto de um objeto, por qualquer método, implica em constante verificação da forma obtida, seguida por alterações nos dados básicos e nova verificação, até que um formato final satisfatório seja obtido. Isso implica na necessidade de rapidez na geração da forma, o que representa cuidado na implementação dos algoritmos de avaliação de polinômios, que são essencialmente lentos. Deve-se portanto, aproveitar características do método que permitam otimizar o tempo de avaliação da superfície. Toda otimização de tempo é útil em aplicações interativas. Uma facilidade disponível é o cálculo incremental, apresentado no item anterior deste capítulo.

O sistema deve:

- ter facilidade de alteração dos pontos de controle, e das escalas utilizadas para definição.

- permitir visualização tri-dimensional dos dados de entrada e mudança dessa visualização para melhor observação.

- ser o mais geral possível, para permitir seu uso com outro método de representação matemática.

No caso de B-splines, deve tornar disponível a versatilidade da técnica, cuidado na escolha dos pontos de controle (alteração da posição, do número e da multiplicidade dos vértices da malha de controle), e escolha da ordem.

3.5.1 Requisitos mínimos para observação de uma figura tri-dimensional.

A apresentação de uma superfície definida matematicamente envolve dois passos básicos: o cálculo das coordenadas tri-dimensionais e alguma técnica de visualização para mostrá-las em dispositivo gráfico bi-dimensional.

Como a quantidade de pontos possível de ser calculada é limitada, busca-se mostrar uma superfície por lofting ou quadriculado.

A apresentação por "lofting" implica na avaliação de duas curvas de fronteira numa direção de variação do parâmetro, seguida de várias "linhas" de pontos paramétricos na outra direção, como uma forma de observação da fronteira e do formato da superfície.

A apresentação por "quadriculado" implica na avaliação de "linhas" da superfície nas duas direções de variação do parâmetro, distantes entre si de um passo paramétrico que define a precisão com que a superfície é observada.

Qualquer dos métodos que se escolha para calcular as "linhas" que mostram a superfície, deve ser seguido de algum tipo de procedimento que permita observar o objeto, que é tri-dimensional, no dispositivo gráfico de saída, bidimensional.

O primeiro procedimento necessário é o de transformação de visualização que transforma as coordenadas do objeto, expressa em sistema de coordenadas próprio (chamado sistema de coordenadas do mundo real), para o sistema de coordenadas em que se localiza o ponto de observação (sistema de coordenadas do observador). Pelas características dos métodos matemáticos de representação, mudar o

sistema de coordenadas de uma superfície, não necessita transformar todos os pontos da superfície; ao invés, pode ser feita pela transformação dos pontos de controle, e re-avaliação da superfície no novo sistema de coordenadas.

Em seguida à avaliação da superfície no novo sistema de coordenadas, é necessário transformar os pontos avaliados por alguma operação de perspectiva, para que possam ser projetados na tela e observados em duas dimensões. Os pontos calculados são então projetados e ligados por segmentos de reta no plano (o que não deixa de ser uma aproximação poligonal da superfície).

As convenções de orientação dos sistemas de coordenadas, e as técnicas de transformação de visualização e perspectiva são fornecidos no apêndice 1.

Ainda com os recursos de visualização e perspectiva, a observação da forma pode se tornar confusa por dois motivos:

- a precisão baixa pode esconder algum detalhe da forma que foi omitido entre várias linhas de apresentação de superfícies;

- o cruzamento de linhas impede a clareza de observação (fig. 3.9).

O primeiro problema só pode ser contornado pelo aumento da precisão no cálculo, ou diminuição do intervalo entre dois valores de parâmetro na mesma direção, o que pode ser permitido ao usuário em caso de necessidade, mas que não é aconselhável em todos os casos, pelo custo disso em termos de tempo de processamento e, de acordo com a implementação escolhida, em termos também de ocupação de memória.

O segundo problema só pode ser evitado pela execução de algum processamento de visualização com retirada de linhas/superfícies escondidas.

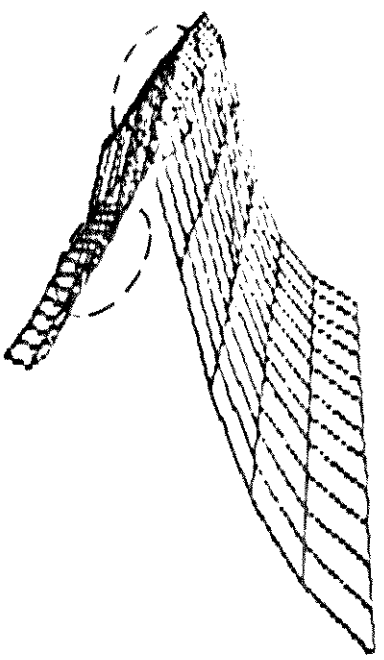


Fig. 3.9 Regiões de cruzamento de linhas que atrapalham a visualização mais clara da superfície

A retirada de linhas escondidas é normalmente um processo trabalhoso. No sistema, foi realizada de duas maneiras que são apresentadas no próximo capítulo. O resultado é uma imagem mais nítida, facilitando o reconhecimento do objeto (figura 3.10) .

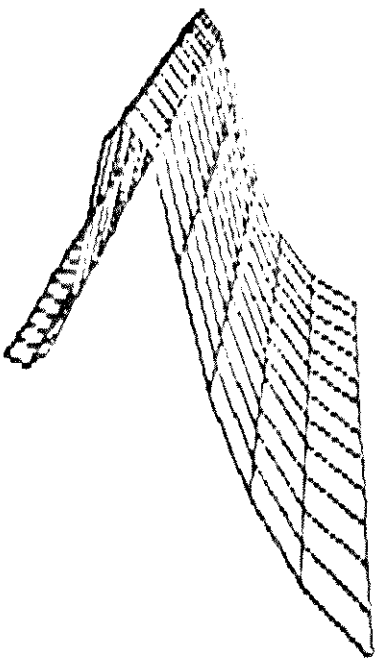


Fig. 3.10 Retirada de linhas escondidas para a superfície da figura 3.9

3.5.2 Interface de Entrada de Dados

O objetivo de uma interface como a do sistema proposto é facilitar a utilização de dados que normalmente são difíceis de manipular, como dados gráficos tri-dimensionais. Assim, deve procurar tornar disponíveis, no contexto do objetivo da aplicação, as características do método de representação utilizado, de uma maneira mais fácil e flexível do que seria fornecer diretamente coordenadas de vértices pelos seus valores reais.

Um dos aspectos desta interface, é que os dados básicos de

entrada de um sistema de modelagem matemática são pontos no espaço. Fornecer tais pontos por valor de coordenadas é um trabalho custoso e difícil, uma vez que a visualização do objeto já torna a interação trabalhosa, quanto mais se essa visualização tiver que se expressar por números. Com base nesses aspectos, a interface de entrada de dados deve ser por edição gráfica, e deve permitir uma visualização em perspectiva dos dados, enquanto estão sendo introduzidos, procurando amenizar a dificuldade de manipulação natural dos dados espaciais.

Por sua vez, a saída do objeto modelado, que é uma superfície matemática, deve permitir reconhecer a forma resultante com segurança, isto é, não deve tornar confusas as linhas de visualização.

3.6 Metodologia de Desenvolvimento

No desenvolvimento de sistemas computacionais, muitos aspectos devem ser considerados, que buscam maior eficiência do software resultante, em termos de aplicabilidade, facilidade de operação, generalidade e principalmente facilidade de manutenção. A área de Engenharia de Software cuida de estudar e propor métodos e técnicas que apoiem o desenvolvimento de sistemas computacionais e que permitam obter as qualidades acima citadas.

A importância de se aceitar e praticar tais técnicas de desenvolvimento reside no fato de que não é possível tratar todos os aspectos de um sistema computacional de tamanho sequer razoável num nível de detalhe muito grande, como é aquele imposto

por uma linguagem algoritmica ou uma linguagem de programação. A divisão do trabalho de construção de sistemas em várias etapas distribui as decisões a serem tomadas em vários níveis, diminuindo a quantidade de tarefas a serem executadas de uma só vez, e por consequência impedindo que erros graves sejam detectados tardiamente.

Em especial, os sistemas gráficos, que, em geral, são de médio/grande porte, justificam a aplicação de tais métodos de Engenharia de Software, embora a literatura específica de algoritmos e sistemas de computação gráfica não cite e não defenda sua aplicação. Ainda assim, o objetivo da área de métodos e técnicas para desenvolvimento de sistemas computacionais propõe que se aplique seus conceitos e ferramentas cada vez mais nas diversas áreas de utilização do computador.

Dentro desse contexto, pela característica e perfil dos algoritmos do software gráfico, são de utilidade para sua construção várias das técnicas da Engenharia de Software, em especial aquelas que apoiam o desenvolvimento descendente ("top-down"), como, por exemplo as técnicas de Análise Estruturada [Ga 83] e Projeto Estruturado [Pa 88] para as primeiras fases da construção de um sistema. A fase de programação pode ser apoiada pelas práticas da programação estruturada.

O desenvolvimento descendente , ou refinamento sucessivo, consiste em subdividir o problema a ser resolvido em funções básicas de alto nível que possam ser tratadas separadamente. Cada uma delas segue sendo subdividida até um nível que permita sua implementação.

Respeitando as diversas fases do chamado ciclo de vida do software, a Análise Estruturada é a primeira técnica empregada. Consiste no uso de ferramentas como o diagrama de fluxo de dados, que permite demonstrar os processos que resolvem o problema, as comunicações entre esses processos, as entidades externas associadas ao ambiente do sistema, e os depósitos de dados necessário à sua manipulação (Ex: figura 4.2).

A próxima fase de desenvolvimento é o projeto do software. Para isso pode ser empregado o Projeto Estruturado, proposto por Yourdon e Constantine que aplica ferramentas como o Diagrama Hierárquico de Funções. Tal diagrama denota a hierarquia (ou estrutura) das funções definidas na análise do software (Ex: figura 4.3), bem como o seu detalhamento até o nível suficiente para permitir a implementação . É determinada a modularização do sistema, e, para os módulos, são definidas as entradas e as saídas. Paralelamente são obtidas as estruturas de dados que apoiam a estrutura do software.

Na fase de implementação, última antes de tornar o software operacional, resta desenvolver algoritmos e testar os módulos de software definidos no projeto. Pode-se utilizar a programação estruturada, como conceito básico de construção dos algoritmos, empregando algum tipo de ferramenta com o diagrama de blocos N-S ou o pseudo-código (português estruturado). Os módulos são implementados e testados de maneira ascendente ("bottom-up"), isto é, primeiro os módulos de mais baixo nível no Diagrama Hierárquico de Funções, e depois os módulos de nível superior, que são apenas controladores de seus subordinados, segundo os conceitos do Projeto Estruturado.

Isso implica em três tipos de testes, nessa fase:

- o teste de unidade: Cada módulo é testado separadamente, tendo as entradas simuladas para as diversas situações possíveis de ocorrer. Dois tipos de teste de unidade são empregados: 1) o teste "caixa preta", onde tais dados de entrada são simulados para um bom conjunto de dados possíveis de ocorrer, e a resposta é verificada quanto ao que se espera como produção do módulo sendo testado; 2) o teste "caixa branca", onde se determina os dados de entrada pela observação do fluxo de controle do algoritmo/programa, isto é, procura-se um conjunto de dados que teste o maior número possível de "caminhos" dentro do código do módulo.

- o teste de integração: após os módulos de mais baixo nível terem sido testados (pois são eles que realmente realizam o trabalho do software), testa-se os níveis superiores na hierarquia. Os módulos de nível superior apenas controlam a chamada dos módulos de mais baixo nível, ou seja, responde pela "interface" entre saída de uns e entrada de outros, bem como controles de execução (seleções e iterações). Erros nesses módulos implicam em maior probabilidade que sejam erros de interface ou controle, já que se "concluiu" que o código interno dos módulos de nível inferior está correto.

- o teste de sistema, que une todos os módulos de software e os torna operacionais no ambiente definitivo de utilização.

A fase posterior e que dura enquanto durar a aplicação do software, é a fase de manutenção, que inclui toda modificação corretiva, adaptativa ou de extensão do software existente. O objetivo da aplicação de técnicas de desenvolvimento é conseguir como resultado um sistema flexível e de fácil manutenção, pois é essa a fase de custo maior do software, via de regra. Apóia a manutenção toda a documentação produzida nas fases anteriores. A figura 3.11 ilustra as fases aqui citadas, do que se chama "ciclo de vida convencional de software".

A documentação do software deve procurar ser abrangente, com a maior clareza possível, e deve ser sempre atualizada para estar de acordo com a última versão em operação do software. Toda ferramenta de desenvolvimento e o próprio código fonte servem como documentação do sistema para posteriores alterações e extensões.

De acordo com o objetivo de flexibilidade do sistema, o software desenvolvido para esta dissertação permite que, embora tenha sido desenvolvido baseado no método de B-spline, sobre a mesma estrutura básica, se possa modelar sólidos por outros métodos, como o de Bézier, por exemplo.

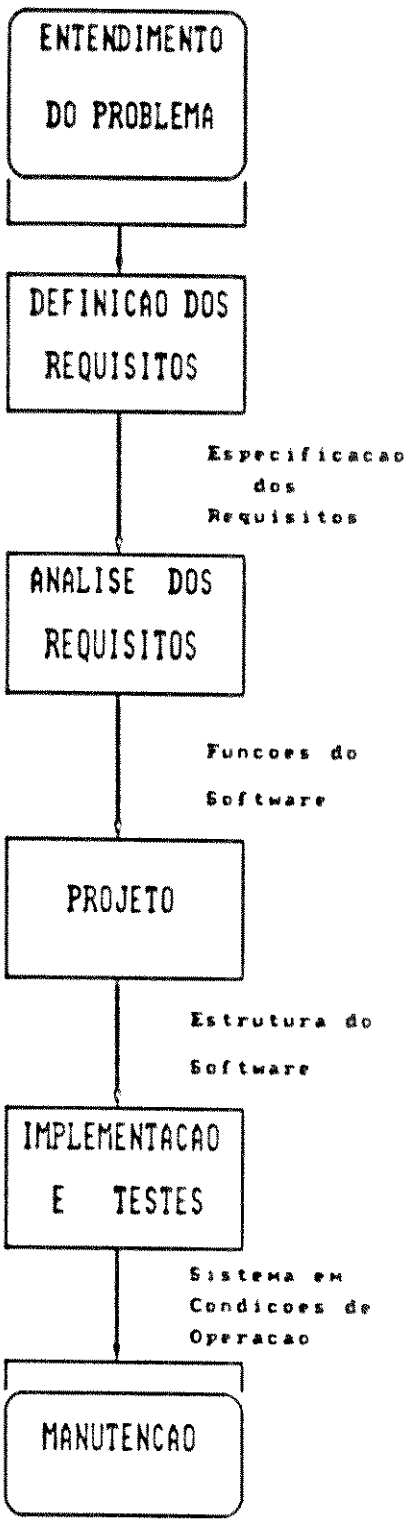


Fig. 3.11 - Processo de Construção de um software.

3.7 Considerações finais

Com o objetivo de desenvolver um sistema de modelagem de uso geral, todos os conceitos apresentados no capítulo foram combinados.

O método de desenvolvimento do referido software foi baseado nas técnicas citadas no item anterior.

Sendo o sistema desenvolvido com o objetivo de implementar a modelagem B-spline, do sistema resultou uma estrutura capaz de processar a modelagem por outros métodos matemáticos, sem alterações de estrutura, com a troca apenas das rotinas de avaliação. Desta forma, também Bézier foi implementado, tornando-se uma opção a mais para o usuário.

O desenvolvimento resultante dos estudos aqui apresentados é descrito no próximo capítulo.

CAPÍTULO IV
SISTEMA DE MODELAGEM POR SUPERFÍCIES MATEMÁTICAS (SMS)

4.1 Considerações Iniciais

O sistema computacional desenvolvido no presente trabalho pretende implementar a modelagem de objetos por superfícies matemáticas, visando produzir um software de propósitos gerais, isto é, um programa que permita modelar objetos tri-dimensionais utilizando métodos numéricos (B-splines e Bézier). Tem por objetivo servir como ferramenta de manipulação de objetos, visando ser incorporado a sistemas gráficos mais potentes posteriormente.

Para atender a esses objetivos deve possuir uma interface com o ambiente e uma base de dados bem definidas, a que possam ter acesso tanto o usuário diretamente quanto programas de aplicação e de cálculos de propriedades (fig. 4.1).

As componentes de interface e modelagem geraram três programas, dois deles, correspondentes à interface, responsáveis pelo diálogo gráfico de introdução de malhas de controle e visualização gráfica dos resultados.

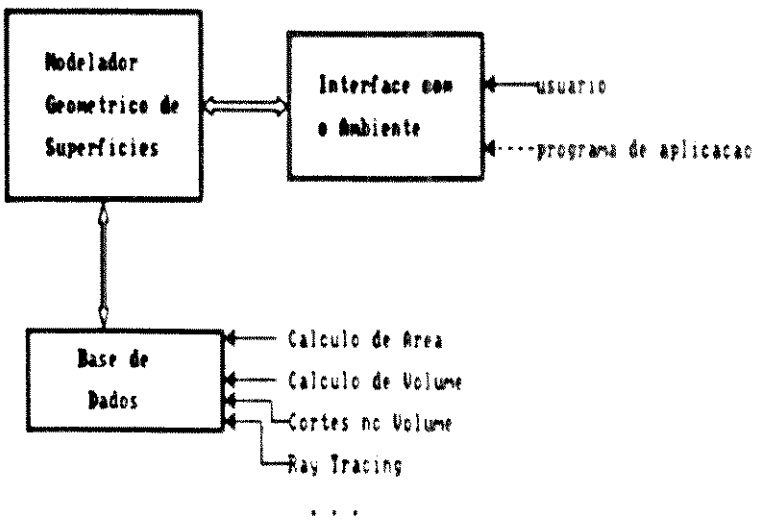


Fig. 4.1 Esquema do SMS (Sistema de Modelagem de Superfícies)

Dentre os métodos estudados foi escolhido o de B-spline, por possuir um conjunto de características mais adequado à maioria das aplicações do esquema de representação. Entretanto, Bézier, que é uma restrição de B-splines, também está disponível como alternativa, uma vez que sobre a mesma estrutura do software de manipulação B-spline, Bézier pode ser perfeitamente incluída.

Por ser um sistema de uso geral, isto é, não dedicado a uma aplicação específica, possui um conjunto de características que necessariamente deverão sofrer alterações na sua adaptação a alguma área em particular. A parte de manipulação direta por programa de aplicação não foi elaborada devido à grande variedade de possíveis acessos, muitas vezes relativos a um conjunto particular de aplicações.

O parágrafo 4.2 apresenta a estrutura geral do sistema desenvolvido e a descrição básica de seus processos.

Os parágrafos 4.3, 4.4 e 4.5, descrevem o desenvolvimento de cada um dos programas básicos que formam o sistema, bem como suas características de implementação.

O parágrafo 4.6 apresenta a técnica utilizada para o desenvolvimento do software.

O parágrafo 4.7 fornece a constituição física do software resultante, bem como detalhes de sua operação.

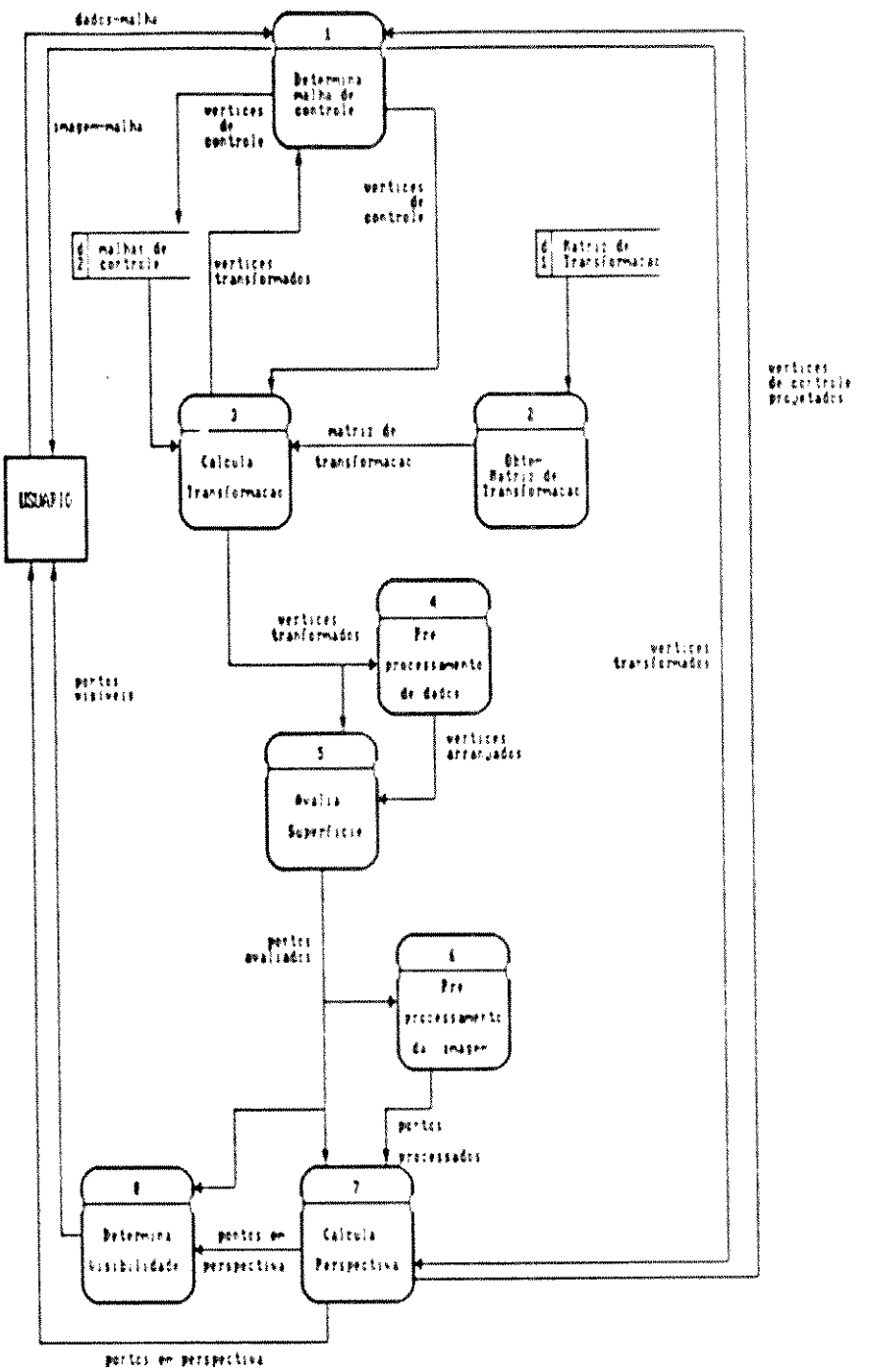
4.2 Estrutura geral do sistema

O desenvolvimento do sistema foi feito através de técnicas estruturadas de análise, projeto e programação.

Como resultado, se obtém um sistema que possui características modulares bastante claras, entre elas a possibilidade de manipular partes do sistema separadamente. Pode-se, portanto, identificar basicamente as três componentes citadas no parágrafo anterior, bem como uma quarta componente de manipulação da imagem, capaz de compor uma cena com vários objetos e processar a retirada de linhas escondidas.

Quanto à base de dados, no caso da modelagem matemática, se reduz a um conjunto de coordenadas espaciais de pontos que definem o formato do objeto, sendo, portanto de estrutura bastante simples.

A versão final do sistema desenvolvido possui um esquema que pode ser observado no diagrama de fluxo de dados (DFD) da figura 4.2a.



a)

Fig. 4.2a. Diagrama de Fluxo de Dados Fundamental do SMS

b)

PROCESSOS

processo	nome	objetivo
1	Determina malha de controle	Corresponde à interface de introdução de dados. O usuário pode introduzir e manipular malhas de controle de objetos, bem como recuperar, destruir ou alterar a visualização. O projeto de objetos deve ser facilitado a partir da criação de um editor de malhas adaptado à introdução dados tridimensionais a partir de dispositivos bidimensionais.
2	Obtem matriz de transformacao	Toma uma malha de controle já editada e recupera a transformação de visualização adequada à gerar a imagem
3	Calcula transformacao	A partir de uma matriz de tranformacao determina as coordenadas da malha de controle no sistema de coordenadas de observador
4	Pre-processamento dos dados	Realiza um ajuste nos vertices de controle para facilitar apresentacao da superficie, conforme o método utilizado de retirada de linhas escondidas
5	Avalia superficie	Calcula valores de pontos sobre a superficie a partir da malha de controle transformada e dos parametros escolhidos pelo usuario de acordo com o método escolhido (Bezier ou B-spline)
6	Pre-processamento da imagem	Responsavel por ajustar os dados da imagem da superficie, para permitir sua utilizacao no algoritmo de determinacao de visibilidade
7	Calcula perspectiva	Calcula a transformacao de perspectiva sobre um ponto em coordenadas do observador para visualizar no tela
8	Determina visibilidade	Corresponde à interface de saida e é responsavel por melhorar a qualidade da imagem da superficie por exemplo por retirada de linhas escondidas

FLUXOS DE DADOS

nome do fluxo	conteudo
dados-malha	Dados que determinam a malha de controle de uma dada superficie. Compreende o numero de pontos de controle, os pontos de controle da malha e os parametros relacionados com funcoes de interface. Juntamente com imagem-malha representa a dialogo de utilizacao da interface de entrada de dados
imagem-malha	Informacao visual da malha resultante dos dados introduzidos
matriz de transformacao	Matriz 4x4 que contem os coeficientes que transformam o sistema de coordenadas mundo no sistema de coordenadas do observador (ver cap. 3)
ponto avaliado	Ponto IR pertencente a superficie, após sua avaliacao pela formula de modelamento matematico escolhido
pontos em perspectiva	Pontos da superficie, em coordenadas de tela
pontos processados	Pontos da superficie, organizados de maneira a facilitar o processo de determinacao de linhas escondidas
pontos visiveis	pontos da superficie, ja testados quanto a visibilidade
vertices arranjados	vertices da malha de controle, arranjados na estrutura de dados para permitir aplicacao do processo de determinacao da malha visivel
vertices de controle	pontos da malha de controle com coordenadas expressas no sistema cartesiano do mundo real
vertices de controle projetados	pontos da malha de controle em perspectiva
vertices transformados	coordenadas x,y e z dos vertices da malha de controle expressos segundo o sistema de coordenadas de observador

DEPOSITOS DE DADOS

nome	funcao
matriz de transformacao	Matriz que transforma coordenadas de mundo real em coordenadas de observador
pontos de controle	Conjunto de malhas de controle das diversas superficies modeladas

ENTIDADES EXTERNAS

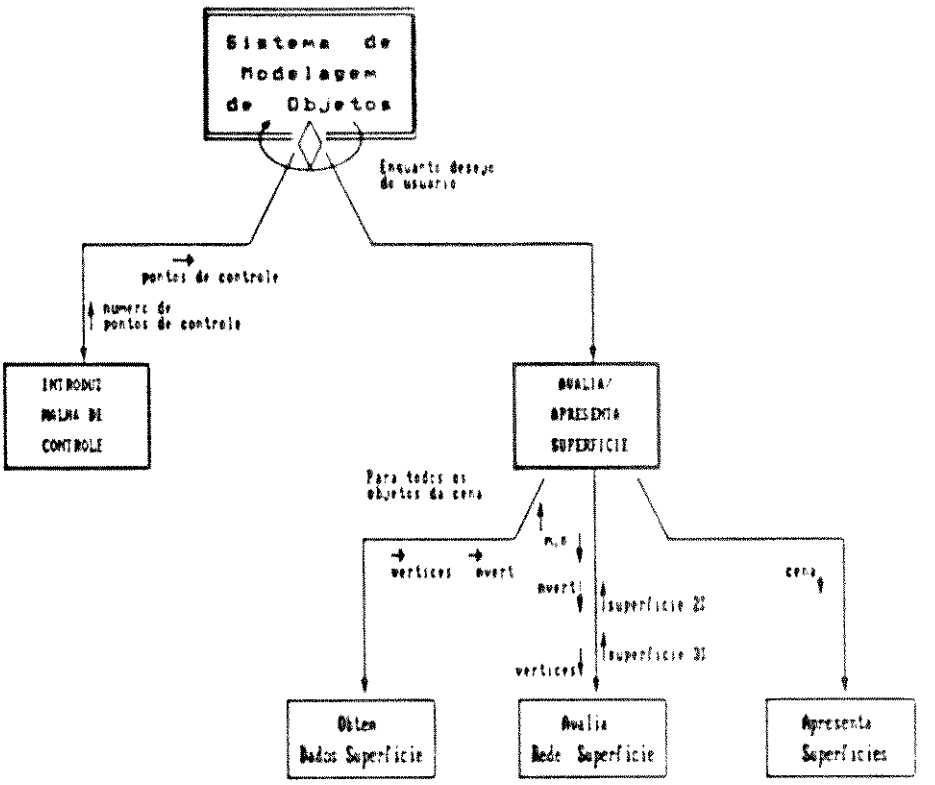
nome	funcao
usuario	Fornece e manipula dados da malha de controle e superficie, e recebe os resultados através de uma tela de video

Fig. 4.2 (continuação)
b. Descrição dos Elementos do DFD

No diagrama da figura 4.2a pode-se identificar os processos responsáveis pela entrada de dados (ou editor gráfico de malhas de controle), pela modelagem em si (ou avaliação) e pela apresentação da imagem, bem como módulos "auxiliares" que realizam processamentos intermediários de objeto e de imagem. Identifica-se também a interação do usuário com tais processos, e a necessidade básica de armazenamento de dados. A descrição dos elementos do DFD é apresentada na fig. 4.2b.

Para o projeto do programa de modelagem, é necessário determinar uma estrutura que comporte os processos e dados do diagrama de fluxo de dados. Considerando as funções de transformação e perspectiva como de uso geral, e o pré-processamento de dados de uso específico do processo de visualização, uma estrutura adequada (embora não única) pode ser a da figura 4.3a. A lógica existente entre os módulos de entrada de dados, modelagem e apresentação da imagem pode variar, levando-se em conta a modularidade do sistema. Basicamente, a introdução, avaliação e apresentação de uma superfície podem ser vistas como três programas separados, que inclusive podem ser desenvolvidos e apresentados separadamente. Nos próximos parágrafos é apresentada a descrição dos programas.

A figura 4.3b descreve os elementos do diagrama de estrutura da figura 4.3a. Neste diagrama, o módulo AVALIA SUPERFICIE corresponde à modelagem em si, e constitui um programa denominado Modelador Geométrico. Devido à modularidade do sistema, tal modelador pode implementar qualquer técnica de modelagem matemática que se utilize de malhas de controle, pois as demais partes do sistema permanecem inalteradas.



a)

MODULOS	
NOME	Funcao
Introduz malha de controle	Corresponde a interface de introducao dos dados de entrada do sistema e armazenamento de dados em arquivo (ver paragrafo 4.4)
Avalia Rede da superficie	Corresponde a funcao de modelamento geometrico em si. E responsavel por fornecer o valor dos pontos no espaco de um trecho de superficie, a partir dos dados da malha de controle e de valores consecutivos dos parametros u e v da superficie. Acumula resultados na estrutura de dados da cena. (ver paragrafo 4.3)
Apresenta superficie	Corresponde a interface de saida do sistema e envolve todos os processos referentes a visualizacao dos objetos de uma cena. Duas possibilidades de implementacao sao consideradas (ver paragrafos 4.5 e 4.6)
Obtem dados de superficie	Obtem de arquivo os dados de um objeto da cena e do usuario os parametros para geracao da superficie modelada Tambem inicia as operacoes de transformacao dos dados.

DADOS DE ENTRADA E SAIDA DOS MODULOS	
NOME	Funcao
Numero de vertices de controle, nvert	Dimensionamento da malha de controle, nas duas direcoes de variacao dos parametros.
Vertices de Controle	Conjunto de vertices de uma malha de controle
m,n	ordem da superficie nas direcoes de variacao dos parametros u e v respectivamente

b)

Fig. 4.3a Diagrama de Estrutura(DE) do SMS
b Descrição dos Elementos do DE

O modelador foi implementado pelos métodos Bézier e B-spline, e tanto um quanto outro podem ser utilizados independentemente do restante do sistema. A seguir o modelador Geométrico é apresentado.

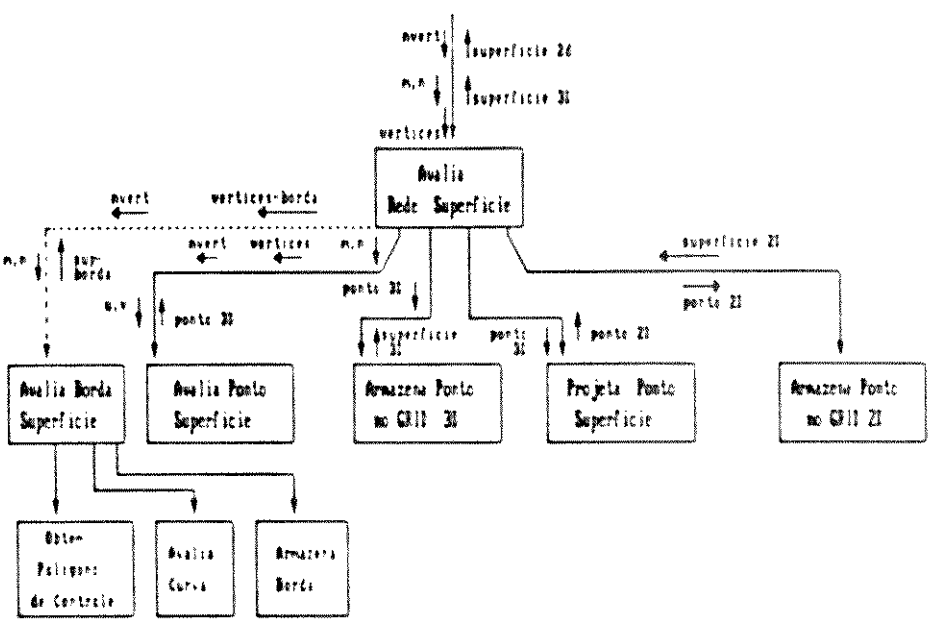
4.3 Modelador geométrico

Na modelagem matemática escolhida, uma superfície é formada pela combinação linear de funções base específicas do método empregado. Os coeficientes dessa combinação são determinados por pontos de controle que formam uma malha. Tal malha define o formato da superfície resultante, que é biparamétrica.

O processo que realiza a modelagem deve avaliar a fórmula matemática do método específico de representação para valores consecutivos de pares de parâmetros u e v da superfície. O resultado é um conjunto de pontos no espaço (um ponto para cada par de parâmetros), pertencentes à superfície resultante. A frequência de cálculo, ou seja, a quantidade de pontos que se deseja amostrar da superfície final, depende unicamente do objetivo da modelagem. Assim a construção do modelador concentra-se em elaborar e implementar algoritmos capazes de avaliar a fórmula do método de modelagem empregado. Para o cálculo são necessários os pontos pertencentes à malha de controle em coordenadas tridimensionais, e, se necessário, os dados sobre a ordem da superfície nas duas direções de variação do parâmetro. O sistema de coordenadas referência no qual são expressos os pontos de controle depende também do objetivo da modelagem e para a

modelagem é indiferente em que sistema os pontos estão expressos. Na estrutura escolhida, foi considerada a necessidade de visualização da superfície enquanto ela é calculada, e portanto, os vértices de controle são expressos já em coordenadas do sistema do observador, e todo ponto avaliado é projetado no plano de tela e armazenado para posterior observação. A fig. 4.4a mostra a estrutura dos módulos do sistema que se encarregam de avaliar todos os pontos de uma superfície. A figura 4.4b descreve os elementos do diagrama de estrutura do modelador.

a)



b)

MODULOS	
NOME	FUNCAO
Avalia Borda Superficie	Obtem pontos da fronteira de uma superficie
Avalia Ponto Superficie	Obtem ponto sobre uma superficie
Armazena Ponto no Grid 3D	Registra o ponto calculado na base de Dados
Projeta Ponto Superficie	Obtem a perspectiva de um ponto tridimensional.
Armazena Ponto no Grid 2D	Registra o ponto projetado na Base de Dados.

DADOS DE ENTRADA E SAIDA DOS MODULOS	
NOME	FUNCAO
Numero de vertices de controle, nvert	Dimensionamento da malha de controle nas duas direcoes de variacao dos parametros.
vertices	Conjunto de vertices de uma malha de controle.
m,n	ordens da superficie nas direcoes de variacao dos parametros s e t respectivamente
sup.borda	pontos de fronteira de uma superficie
superficie 3D	conjunto de pontos no espaco sobre a superficie calculada
superficie 2D	conjunto de pontos da imagem da superficie

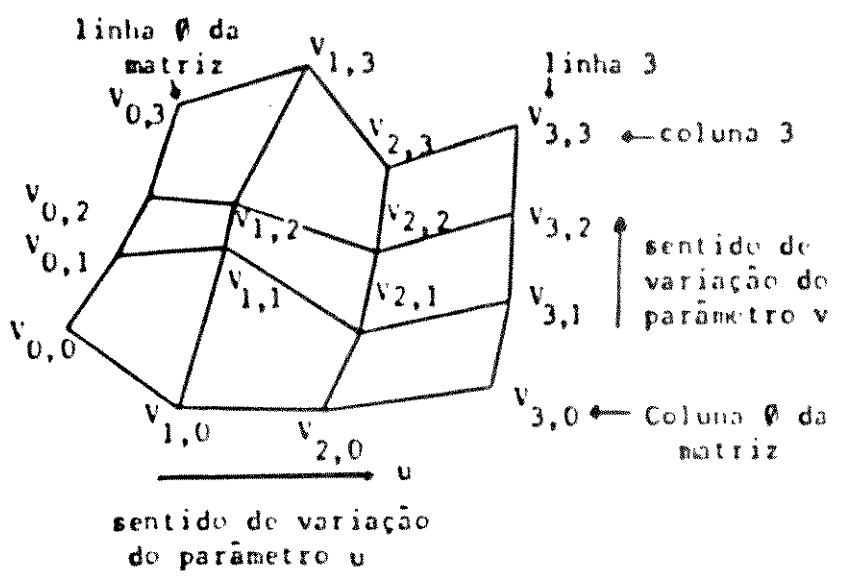
Fig. 4.4a Diagrama de estrutura da avaliação de uma superfície matemática.
b Descrição dos elementos do Diagrama

A implementação do modelador depende da definição das estruturas de dados para representação interna da malha de controle, da superfície no espaço e da superfície projetada, que são apresentadas a seguir.

4.3.1 Estruturas de Dados

A primeira estrutura necessária é a malha de controle, que define o conjunto de pontos que, de acordo com a modelagem matemática, determinam o formato do objeto resultante. Um exemplo de malha de controle é fornecido pela figura 4.5.

a)

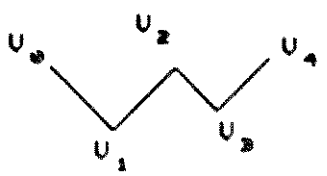


b)

$$M = \begin{bmatrix} V_{00} & V_{01} & V_{02} & V_{03} \\ V_{10} & V_{11} & V_{12} & V_{13} \\ V_{20} & V_{21} & V_{22} & V_{23} \\ V_{30} & V_{31} & V_{32} & V_{33} \end{bmatrix} \quad \begin{matrix} \rightarrow v \\ \downarrow u \end{matrix}$$

Fig. 4.5a Ilustração de uma malha de controle com:
número de vértices na direção u: 4
número de vértices na direção v: 4.
b Matriz correspondente à malha

No caso de uma curva, ou mesmo no caso da determinação da fronteira de uma superfície B-spline, é necessária uma estrutura que armazene um polígono de controle, que é a versão da malha de controle para curvas. A figura 4.6 fornece um exemplo de polígono de controle.



a)

$$P = \left[V_0 \quad V_1 \quad V_2 \quad V_3 \quad V_4 \right] \xrightarrow{u}$$

b)

Fig. 4.6a Ilustração de um polígono de controle com 5 vértices.
b Vetor resultante do armazenamento dos vértices

A avaliação da superfície é feita para um conjunto de pontos ao longo da região definida pelo intervalo de variação dos parâmetros u e v. Normalmente, se o objetivo é a visualização, esse conjunto de pontos forma uma "rede" que "cobre" a superfície, com variação constante entre os valores distintos do parâmetro u e do parâmetro v. Uma estrutura de dados é necessária para armazenar os valores dos pontos avaliados da superfície, para posteriores processamentos (por exemplo, visualização). O dimensionamento dessa estrutura depende da precisão com que se deseja gerar a superfície.

Por fim, cada ponto gerado em coordenadas tridimensionais

deve ser projetado no plano de tela (apêndice 1) para posterior visualização. Usa-se também uma estrutura de dados que armazena todos os pontos projetados, isto é, coordenadas bi-dimensionais dos vértices da "rede" em perspectiva.

A descrição e definição das estruturas de dados mencionadas são apresentadas na figura 4.7.

Estruturas de dados - Avaliacao de Superficie

entidade	sinonimos	objetivo	organizacao
malha de controle	malha_mundo, malha_observador, m_m, m_o	Armazena os vertices, em coordenadas 3D, da malha que define a forma da superficie.	Matricial, onde cada elemento da matriz corresponde a um vertice da malha, contendo suas tres coordenadas reais.
poligono de controle	pp	Armazenar um conjunto de vertices do poligono que defina a forma de uma curva.	Vetorial, onde cada elemento do vetor corresponde a um vertice do poligono de controle, contendo suas tres coordenadas.
Superficie 3D	Grid 3D Rede 3D	Armazenar os pontos avaliados da superficie, em coordenadas 3D.	Matricial, com numero de elementos dependente da precisao da superficie. Cada elemento da matriz contem um ponto 3D da superficie.
Superficie 2D	Grid 2D Rede 2D	Armazenar os pontos da superficie projetados em tela, isto e, a imagem da superficie.	Matricial, com numero de elementos dependente da precisao da superficie. Cada elemento da matriz contem um ponto 2D da imagem.

Implementacao Utilizada

entidade	Implementacao Pascal correspondente
malha de controle	malha_3D = ARRAY[0..maxlin,0..maxcol,1..3] OF REAL;
poligono de controle	XY2array = ARRAY[0..dmax,1..3] OF REAL; *
Superficie 3D	tipo_grid_3D = ARRAY[0..maxdiv,0.maxdiv] OF RECORD coord:COORDENADA_3D; END;
Superficie 2D	tipo_grid_2D = ARRAY[0..maxdiv,0..maxdiv] OF RECORD visivel:BOOLEAN; coord:COORDENADA_2D; END;

* fora dos padroes de nomenclatura do sistema, pois se mantem aqui a notacao do algoritmo para curvas, de (He 83).

constantes utilizadas

maxlin,maxcol	- dimensoes maximas da malha de controle
dmax	- dimensao maxima do poligono de controle (deve ser o max(maxlin,maxcol))
maxdiv	maxima divisao da superficie, numa das direcoes de variacao dos parametros

Fig 4.7 Descrição das Estruturas de dados do Modelador

4.3.2 Estrutura do Processo

O processo de avaliar uma superfície a partir de seus pontos de controle implica em criar funções que implementem a fórmula:

$$P(u,v) = \sum_{j=0}^{m} \sum_{i=0}^{n} B(u)\ B(v)\ V_{i,j} \qquad \text{(eq. 4.1)}$$

onde:

B(t) - função de "blending" do método utilizado

$V_{i,j}$ - ponto de controle da malha que define o formato da superfície.

A diferença entre os vários métodos está na função de "blending" B(t), que possui uma formulação distinta para cada método, como apresentado em capítulos anteriores. Utilizando a malha de controle completa e um par de valores para os parâmetros u e v, consegue-se o valor de um ponto da superfície objetivo pela equação 4.1. A modelagem consiste em executar esse procedimento consecutivamente para vários pontos, a partir de uma dada precisão. A avaliação de um ponto da superfície pode ser feita, no sistema construído, por qualquer dos dois métodos apresentados nos próximos parágrafos.

A geração de uma superfície a ser visualizada em um sistema CAD, é usualmente feita de duas formas distintas: por "lofting" ou por "quadriculado". Por qualquer dos métodos o resultado é uma rede de pontos sobre a superfície, que deve ser gerada conservando o valor de um dos parâmetros e variando o outro do

valor mínimo ao máximo, a intervalos constantes. Em seguida varia-se novamente o primeiro parâmetro e o processo se repete.

Assim, o procedimento de avaliação é executado um número de vezes dependente da precisão desejada.

4.3.2.1 Avaliação de uma Superfície B-spline

O método de B-spline para representação superficial de objetos foi, no estudo desenvolvido, considerado o que, dentre os mais amplamente aplicados, possui maior flexibilidade de projeto pelas características matemáticas de sua formulação.

A obtenção de um ponto de uma superfície B-spline é feita com base na malha de controle e na definição da ordem da superfície nas duas direções de variação dos parâmetros.

A figura 4.8 fornece o trecho de programa Pascal que avalia um ponto sobre uma superfície B-spline. Trata-se de uma extensão do algoritmo de [Ne 83] para curvas B-spline.

No cálculo do quadriculado que representa o objeto, a avaliação é chamada várias vezes, para toda a região (pacth) sendo modelada. No entanto, o método B-spline possui a característica de que sua formulação matemática não é definida para valores extremos dos parâmetros. Essa característica, chamada "problema de fronteira" pode ser manipulado com objetivos mais diversos.

```
PROCEDURE Spline_sup(p:MALHA_3D;u,v:REAL;k,l,n,m:INTEGER;VAR xs,ys,zs:REAL;
                     limit_u,limit_v:REAL);
{Determina o ponto da superficie b-spline correspondente aos parametros u e v
 a superficie e' modelada pelos pontos de controle dados por p.}

var
   i,j
               :INTEGER;
   b1,b2,b             {contem os valores das funcoes de blending}
               :REAL;

begin
   xs:=0.0;ys:=0.0;zs:=0.0;
   for i:=0 to n do
   begin
      knotk:=k;knotn:=n;
      b1:=nblend(i,k,u,n);
      if b1<>0 then (* nada se altera se b1 =0 *)
         for j:=0 to m do
         begin
            knotk:=l;knotn:=m;
            if (j=m) and (v=limit_v) then
               b2:=1
            else
               b2:=nblend(j,l,v,m);
            b:=b1*b2;
            xs:=xs+b*p[i,j,x];
            ys:=ys+b*p[i,j,y];
            zs:=zs+b*p[i,j,z];
         end;
   end;
end;
```

Fig. 4.8 Programa Pascal de Avaliação de uma superfície B-spline.

O contorno do "problema de fronteira" é feito, no SMS, por um critério bastante simples. Cada uma das quatro fronteiras do pedaço é representada por uma curva B-spline. Como dados para a geração da curva, tem-se que a ordem é a mesma da superfície, na direção de variação do parâmetro que a curva limita. O poligono de controle para a curva é a própria fronteira da malha de controle do pedaço. Esse tipo de controle do problema de fronteira obriga que os quatro vértices extremos da malha de controle pertençam à superfície gerada. Além disso, proporciona continuidade entre pedaços vizinhos de um mesmo objeto, no caso

da necessidade de conexão entre superfícies. Pedaços vizinhos necessitam ter a mesma fronteira da malha de controle para ter continuidade de função.

A figura 4.9 mostra o algoritmo de geração da rede B-spline, adotando esse tipo de contorno para o problema de fronteira.

```
PROCEDURE avalia_registra(VAR p_MALHA_3D;n,m,i,k:INTEGER;npatch:INTEGER);
{obtem o grid de um patch avaliado, a partir de sua malha observador, e
 parametros da funcao spline}

TYPE
  VETOR_3D = ARRAY[0..maxdiv] of record
                                   coord :COORD_3D
                                 end;
  VETOR_2D = ARRAY[0..maxdiv] of record
                                   coord : COORDENADA_PLANO
                                 end;
var
   limit_u,limit_v,
   passo_u,passo_v
                          :REAL;
   pi,pf
                          :COORD_3D;


PROCEDURE curva(pont:XYZarray; passo,limite:REAL; m,i:INTEGER; VAR cont:INTEGER;
                VAR p_esp :VETOR_3D; VAR p_proj :VETOR_2D);

{ Avalia e apresenta uma curva B-spline definida pelo poligono de controle
  pont, pela ordem i, e pelo numero de pontos m+1 }

var
   v                      {parametro de definicao da curva}
         :REAL;
   j,j,
   xs,ys,zs               {ponto pertencente a superficie}
         :REAL;
   xaux,yaux

                          :REAL;

begin
  cont := 0;
  {* Ponto inicial eh um "canto" da malha *}
  p_esp[cont].coord.x := pont[0,1];
  p_esp[cont].coord.y := pont[0,2];
  p_esp[cont].coord.z := pont [0,3];
  perspec(pont[0,1],pont[0,2],pont[0,3],
          p_proj[cont].coord.x,p_proj[cont].coord.y);
  v:=passo;
  while v<=limite do
  begin
    cont := cont +1;
    Bspl_curva(xs,ys,zs,v,m,i,pont);
    p_esp[cont].coord.x := xs;
    p_esp[cont].coord.y := ys;
    p_esp[cont].coord.z := zs;
    perspec(xs,ys,zs,p_proj[cont].coord.x,p_proj[cont].coord.y);
    v:=v+passo;
  end;
  cont := cont +1;
  {* Ponto final igual a outro "canto" da malha *}
  p_esp[cont].coord.x := pont[m,1];
  p_esp[cont].coord.y := pont[m,2];
  p_esp[cont].coord.z := pont [m,3];
  perspec(pont[m,1],pont[m,2],pont[m,3],p_proj[cont].coord.x,p_proj[cont].co
end;
```

Fig. 4.9 -Trecho do programa de geração da rede B-spline. (continua)

```
PROCEDURE gera_supIp MULMA_3D,k,l,n,m: INTEGER;VAR pi,pf: COORD_3D);
{Apresenta curvas na direcao do parametro v para valores successivos do
 parametro u}

var
    xs,ys,zs,                   {ponto no espaco pertencente a superficie}
    u,v                         {ponto parametrico da superficie}
            :REAL;
    p1                          {poligono de controle da curva limitante}
            :XYZarray;
    i,j,cont
            :INTEGER;
    xant,yant,
    xtel,ytel
                                :REAL;
    p_esp
                    :VETOR_3D;
    p_proj
                    :VETOR_2D;

begin
    pi.x := p[0,0,x]; pf.x:=p[0,0,x];
    pi.y := p[0,0,y]; pf.y:=p[0,0,y];
    pi.z := p[0,0,z]; pf.z:=p[0,0,z];
    cont_l :=-1;
    u:=0.0;
    while u < limit_u do
    begin
        cont_l := cont_l + 1;
        contcol := 0;
        v:=0;
        (* det. do ponto inicial da linha da malha *)
        for i:=0 to n do
            for j:=1 to 3 do
                p1[i,j]:=p[i,0,j];
        Bspl_curva(xs,ys,zs,u,n,k,p1);
        (* teste de limitacao da regiao da superficie *)
        if xs < pi.x then
          pi.x := xs;
        if ys < pi.y then
          pi.y := ys;
        if zs < pi.z then
          pi.z := zs;
        if xs  > pf.x then
          pf.x := xs;
        if ys > pf.y then
          pf.y := ys;
        if zs > pf.z then
          pf.z := zs;
        perspec(xs,ys,zs,xtel,ytel);
        patch_3d[npatch]^[cont_l,contcol].coord x:=xs;
        patch_3d[npatch]^[cont_l,contcol].coord y:=ys;
        patch_3d[npatch]^[cont_l,contcol].coord z:=zs;
        patch_2d[npatch]^[cont_l,contcol].coord x:=xtel;
        patch_2d[npatch]^[cont_l,contcol].coord y:=ytel;
        xant:=xtel;yant:=ytel;
        v:=passo_v;
        while v < limit_v do
        begin
          spline_supIp(p,u,v,k,l,n,m,xs,ys,zs,limit_u,limit_v);
          (* teste de limitacao da regiao da superficie *)
          if xs < pi.x then
            pi.x := xs;
          if ys < pi.y then
            pi.y := ys;
          if zs < pi.z then
            pi.z := zs;
          if xs  > pf.x then
            pf.x := xs;
          if ys > pf.y then
            pf.y := ys;
          if zs > pf.z then
            pf.z := zs;
```

Fig. 4.9 -Trecho do programa de geração da rede B-spline. (continuação)

```
        perspec(xs,ys,zs,xtel,ytel);
        contcol := contcol + 1;
        patch_3d[npatch]^[cont_l,contcol].coord x:=xs;
        patch_3d[npatch]^[cont_l,contcol].coord y:=ys;
        patch_3d[npatch]^[cont_l,contcol].coord z:=zs;
        patch_2d[npatch]^[cont_l,contcol].coord x:=xtel;
        patch_2d[npatch]^[cont_l,contcol].coord y:=ytel;
        reta(round(xant),round(yant),round(xtel),round(ytel));
        xant:=xtel;yant:=ytel;
        v:=v+passo_v;
      end;
      (* det. do ponto final da linha da malha *)
      for i:=0 to n do
        for j:=1 to 3 do
          p1[i,j]:=pl[i,m,j];
      Bspl_curva(xs,ys,zs,u,n,k,p1);
      (* teste de limitacao da regiao da superficie *)
      if xs < pi.x then
        pi.x := xs;
      if ys < pi.y then
        pi.y := ys;
      if zs < pi.z then
        pi.z := zs;
      if xs  > pf.x then
        pf.x := xs;
      if ys > pf.y then
        pf.y := ys;
      if zs > pf.z then
        pf.z := zs;
      perspec(xs,ys,zs,xtel,ytel);
      reta(round(xant),round(yant),round(xtel),round(ytel));
      contcol := contcol + 1;
      patch_3d[npatch]^[cont_l,contcol].coord x:=xs;
      patch_3d[npatch]^[cont_l,contcol].coord.y:=ys;
      patch_3d[npatch]^[cont_l,contcol].coord.z:=zs;
      patch_2d[npatch]^[cont_l,contcol].coord x:=xtel;
      patch_2d[npatch]^[cont_l,contcol].coord.y:=ytel;
      u:=u+passo_u;
    end;
    (* Quando u = limit_u, a fronteira vira a curva limitante *)
    for i := 0 to m do
      for j := 1 to 3 do
        p1[i,j] := pl[n,i,j];
    curva(p1,passo_v,limit_v,m,l,cont,p_esp,p_proj);
    traca_vetor (p_proj,cont);
    cont_l := cont_l + 1;
    for j := 0 to cont do
    begin
      patch_3d[npatch]^[cont_l,j].coord x := p_esp[j].coord x;
      patch_3d[npatch]^[cont_l,j].coord y := p_esp[j].coord y;
      patch_3d[npatch]^[cont_l,j].coord z := p_esp[j].coord z;
      patch_2d[npatch]^[cont_l,j].coord x := p_proj[j].coord x;
      patch_2d[npatch]^[cont_l,j].coord y := p_proj[j].coord y;
    end;
  end;


begin
    (* det. de passos para criacao de curvas *)
    limit_u:=n-k+2;limit_v:=m-l+2;
    prec_u := 4;
    prec_v := 4;
    passo_u :=limit_u/(k*prec_u);
    passo_v:=limit_v/(l*prec_v);
    cena_sup[npatch].nv := round(limit_v/passo_v);
    cena_sup[npatch].nu := round(limit_u/passo_u);
    gera_sup(p,k,l,n,m,pi,pf);
    boxes[npatch].pi := pi;
    boxes[npatch].pf := pf;
    mostra_patch(npatch)
end;
```

Fig. 4.9 -Trecho do programa de geração da rede B-spline. (continuação)

O método B-spline de representação de superfícies conta com a vantagem de, entre uma alteração e outra do objeto, desde que só se altere a posição de pontos de controle, poder re-avaliar apenas parte da superfície, conservando os valores da função de "blending" para todas as regiões não afetadas pela modificação do projetista. Isso é consequência da característica de localidade do próprio método matemático.

Em outras palavras, como o projeto da forma da superfície deve ser interativo, com avaliações consecutivas da mesma superfície, o tempo em computar novamente a superfície pode ser reduzido aproveitando-se a característica de localidade da fórmula.

4.3.2.2 Avaliação de uma Superfície Bézier.

O método de Bézier para representação de superfícies matemáticas não possui problemas de fronteira e possui avaliação direta para todos os valores dos parâmetros. Assim, a avaliação de um ponto de uma superfície Bézier ocorre pela aplicação dos valores dos pontos de controle na fórmula. Ao contrário de B-spline, a ordem (e, portanto, o grau) da superfície resultante não é opção na formulação de Bézier. O grau da superfície depende da quantidade de pontos de controle em cada direção de variação do parâmetro. O algoritmo para avaliação de um ponto sobre uma superfície Bézier é generalização do algoritmo para curvas, e o programa resultante é apresentado na figura 4.10.

```
PROCEDURE gera_sup(p:MALHA_3D;k,l,n,m:INTEGER;VAR pi,pf:COORD_3D);
{Apresenta curvas na direcao do parametro v para valores successivos do
 parametro u}

var
   xs,ys,zs,                  {ponto no espaco pertencente a superficie}
   u,v                        {ponto parametrico da superficie}
            :REAL;
   p1                         {poligono de controle da curva limitante}
            :XYZarray;
   i,j,cont
            :INTEGER;
   xant,yant,
   xtel,ytel
                              :REAL;
   p_esp
                  :VETOR_3D;
   p_proj
                  :VETOR_2D;

begin
   pi.x := p[0,0,x]; pf.x:=p[0,0,x];
   pi.y := p[0,0,y]; pf.y:=p[0,0,y];
   pi.z := p[0,0,z]; pf.z:=p[0,0,z];
   cont_l :=-1;
   u:=0.0;
   perspec(pi.x,pi.y,pi.z,xant,yant);
   while u <= limit_u do
   begin
      cont_l := cont_l + 1;
      contcol := -1;
      v:=0;
      while v <= limit_v do
      begin
        spline_sup(p,u,v,k,l,n,m,xs,ys,zs);
        (* teste de limitacao da regiao da superficie *)
        if xs < pi.x then
          pi.x := xs;
        if ys < pi.y then
          pi.y := ys;
        if zs < pi.z then
          pi.z := zs;
        if xs  > pf.x then
          pf.x := xs;
        if ys > pf.y then
          pf.y := ys;
        if zs > pf.z then
          pf.z := zs;
        perspec(xs,ys,zs,xtel,ytel);
        contcol := contcol + 1;
        patch_3d[npatch]^[cont_l,contcol].coord.x:=xs;
        patch_3d[npatch]^[cont_l,contcol].coord.y:=ys;
        patch_3d[npatch]^[cont_l,contcol].coord.z:=zs;
        patch_2d[npatch]^[cont_l,contcol].coord.x:=xtel;
```

Fig. 4.10 Programa Pascal de Geração de uma superfície de Bézier.
(continua)

```
            patch_2d[npatch]^[cont_l,contcol].coord.y:=ytel;
            reta(round(xant),round(yant),round(xtel),round(ytel));
            xant:=xtel;yant:=ytel;
            v:=v+passo_v;
          end;
          u:=u+passo_u;
      end;
  end;


  begin
      (* det. de passos para criacao de curvas *)
      limit_u:=1;limit_v:=1;
      prec_u := 4;
      prec_v := 4;
      passo_u:=limit_u/(k*prec_u);
      passo_v:=limit_v/(l*prec_v);
      cena.sup[npatch].nv := round(limit_v/passo_v);
      cena.sup[npatch].nu := round(limit_u/passo_u);
      gera_sup(p,k,l,n,m,pi,pf);
      boxes[npatch].pi := pi;
      boxes[npatch].pf := pf;
      mostra_patch(npatch)
  end;
```

Fig. 4.10 Programa Pascal de Geração de uma superfície de Bézier.
(continuação)

4.3.3 Conceitos Básicos envolvidos

Os conceitos a respeito da modelagem geométrica foram apresentados nos capítulos II e III deste trabalho, no que se refere às formulações e às características dos métodos envolvidos. Destaca-se, desses conceitos, a vantagem existente do método de B-splines sobre o de Bézier, devido às características matemáticas, em especial localidade e determinação da ordem da superfície.

4.3.4 Características de Implementação

O programa de modelagem do objeto é responsável por gerar pontos sobre a superfície projetada.

As entradas do programa de modelagem incluem o número de pontos de controle (nas duas direções de variação dos parametros), os pontos da malha, e a ordem (quando necessária).

Os dados podem vir de arquivo (cujo nome o usuário fornece) ou da própria estrutura de dados, ambos produzidos pelo programa de interface de entrada de dados, apresentado no próximo parágrafo.

A base de dados é, portanto, muito simples, consistindo de um arquivo de números reais representando as coordenadas dos vários pontos da malha. Há um arquivo por superfície projetada pelo usuário. Tanto a modelagem por B-splines como Bézier utiliza-se da mesma base de dados.

A opção pela inclusão também do método de Bézier é devida à sua grande divulgação entre os métodos de geração de splines. A estrutura do programa e a facilidade de implementação do método em si levaram à possibilidade de inclusão dessa opção sem grandes dificuldades, em especial sem nenhuma alteração das estruturas do programa de visualização, ou mesmo do editor de malhas de controle.

A precisão com que a superfície é gerada é calculada inicialmente com base na ordem e no número de pontos de controle da malha, mas pode ser aumentada a desejo do usuário.

Independente da opção de método de visualização (feita após os cálculos da modelagem) a superfície é apresentada enquanto é

gerada, sem retirada de linhas escondidas.

A opção pelo método de geração da forma (Bézier ou B-spline) é feita incluindo-se uma ou outra biblioteca de rotinas no programa de geração de superfície.

A biblioteca física de nome "aval-reg.pro" armazena as rotinas da avaliação B-spline e a biblioteca "aval-bez.pro", as rotinas de Bézier.

Além destas, utiliza-se de uma biblioteca de rotinas geral do sistema, para manipulação de matrizes.

A introdução das malhas de controle e a montagem dos arquivos, são feitas através de um programa que permite edição de dados tridimensionais com visão interativa do resultado. O próximo parágrafo apresenta esse programa.

4.4 Interface de Entrada de Dados

O procedimento de introdução de dados para geração de um objeto por superfícies matemáticas envolve uma série de questões como, por exemplo, o tipo de usuário do sistema.

Como o sistema desenvolvido é de uso geral, e, portanto, não considera aplicações específicas, é de interesse da interface facilitar a manipulação de pontos tridimensionais para um usuário que tem apenas uma idéia da forma que deseja obter. O usuário deve ter noção intuitiva de qual o comportamento da superfície resultante para um conjunto de pontos de controle, ou seja, deve ter idéia do comportamento de uma superfície B-spline ou Bézier, conforme o modelo escolhido. Essa é a própria base da modelagem

matemática: o projetista "sabe" qual vai ser o resultado aproximado da forma para um dado conjunto de pontos.

Com base nesse aspecto, foi desenvolvido um editor gráfico que permite ao usuário introduzir e posteriormente modificar dados de uma malha de controle.

4.4.1 Requisitos de Interface

Uma vez que a aplicação não é específica, o objetivo da interface deve ser auxiliar a introdução de dados tridimensionais por coordenadas reais, na formação de uma malha de controle.

A interface possui um editor gráfico que permite ao usuário introduzir um ponto por cursor, através de dois planos de edição: o plano xy e o plano xz do sistema de coordenadas do mundo real, manipulável diretamente pelo usuário.

A cada ponto introduzido, a interface apresenta a situação atual da malha de controle, pela visualização e projeção da malha na própria tela.

Os pontos da malha de controle são introduzidos na ordem de variação dos parâmetros, isto é, para cada índice da malha na direção u, obtém-se todos os pontos da "linha" da malha para índices da direção v.

O usuário deve, então, elaborar seus dados com visão dos vértices da malha em projeçãao ortogonal sobre os planos xy e xz, e a visão da malha no espaço é fornecida pelo próprio programa.

A visualização pode ser alterada para melhor observação do usuário, através de rotações, translações e escalas da malha. A

superfície vai ser gerada sob um ponto de vista idêntico à visualização da malha que for do gosto do usuário.

O usuário tem liberdade de mudar a escala com que está introduzindo os dados, alterar pontos introduzidos, gravar ou recuperar resultados. A geração da superfície só pode ser executada com o término da introdução dos dados da malha, ou seja, só com a malha completa.

4.4.2 Estruturas de Dados

A estrutura de dados da interface é responsável por guardar as informações do "estado" da tela a um dado instante. A tela é originalmente dividida em quatro regiões, uma para a introdução de dados do plano xy, outra para introdução dos dados do plano xz, uma para apresentação interativa da malha de controle e uma região de "menus", onde são fornecidos dados a respeito do objeto sendo inserido, as opções do usuário e os resultados numéricos das operações.

O estado da tela é representado por uma estrutura que nada mais é do que um registro que reflete o estado da tela de inserção. A figura 4.11 apresenta o estado inicial de uma tela de inserção. A figura 4.12 apresenta a estrutura de dados de controle da tela. Além desta estrutura a interface manipula as estruturas de malha de controle tridimensional e projetada, apresentadas no parágrafo do modelador, uma vez que são comuns aos dois programas (fig. 4.7).

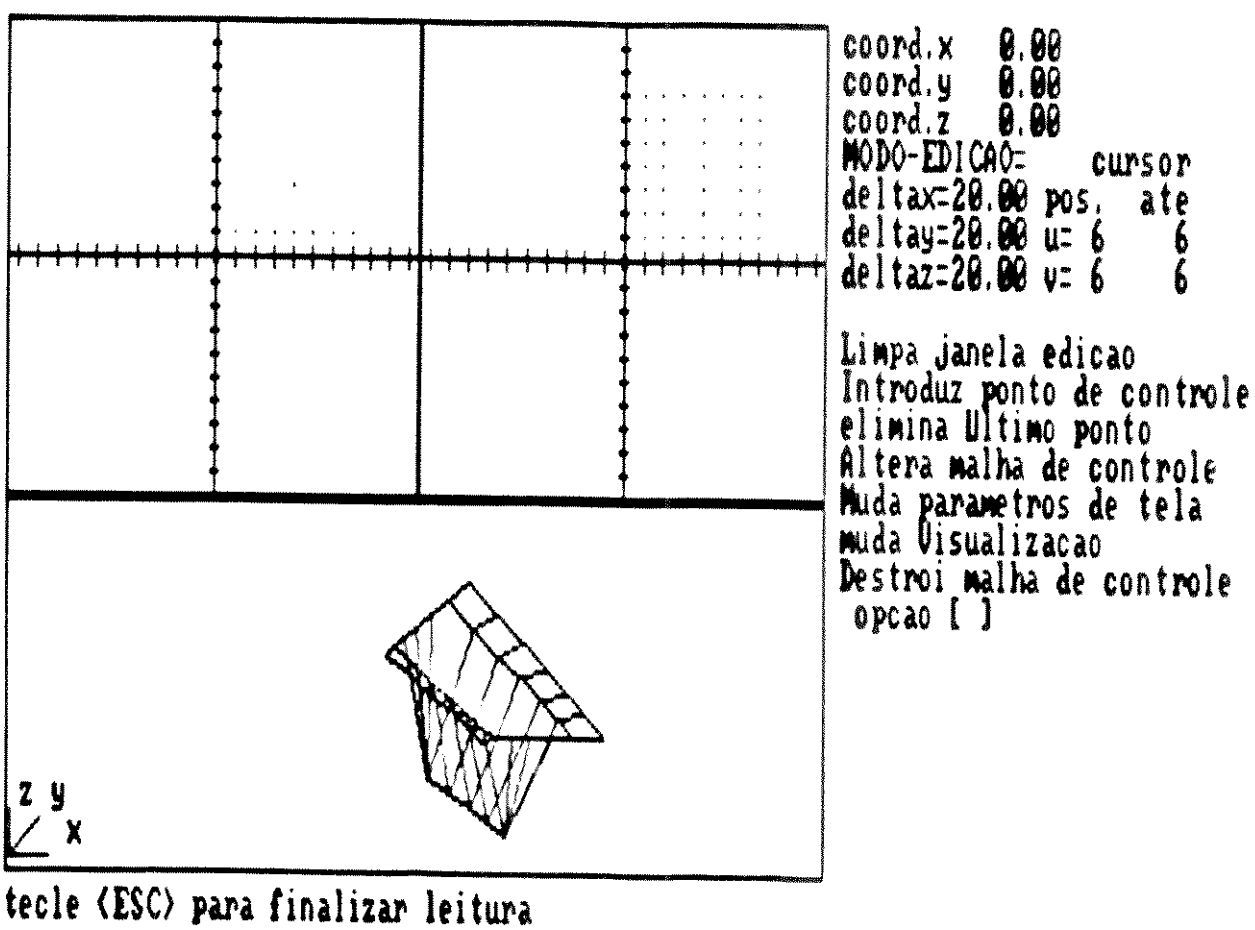


Fig. 4.11 Tela inicial da interface de inserçao da malha de controle

Estruturas de dados - Interface de Entrada			
entidade	**sinonimos**	**objetivo**	**organizacao**
registro de situacao corrente de edicao	situacao corrente, sc	Armazena os dados relativos a situacao da tela de edicao a cada momento.	Registro, onde cada cada campo reflete a situacao de uma regiao de tela do programa de interface.
registro de escalas da malha	malha desenho, md	Armazenar um conjunto de dados que determinam as escalas com que a malha e manipulada, na direcao x, y e z.	Registro, onde cada campo corresponde a um dado capaz de auxiliar a localizacao e medida da malha de controle.

Implementacao Utilizada	
entidade	**Implementacao Pascal correspondente**

registro de situacao corrente de edicao	situacao_corrente = RECORD
	  ortela_xy,ortela_xz,ortela_proj,
	  sup_jar_xy,inf_jar_xy,
	  sup_jar_xz,inf_jar_xz,
	  sup_jar_proj,inf_jar_proj,
	  or_mundo_proj,pos_cursor_xy,pos_cursor_xz
	  pos_cursor_proj,
	  tamanho_jar_xy,tamanho_jar_xz,
	  tamanho_jar_proj
	  : COORDENADA_TELA,
	  orint_xy,orint_xz,ccursor_xy,ccursor_xz
	  :COORDENADA_PLANO,
	  modo_leitura
	  :LEITURA,
	  dx,dy,dz
	  :REAL,
	  inter_x,inter_y,inter_z
	  :INTEGER,
	  ponto_cursor_xy,ponto_cursor_xz,
	  ponto_cursor_proj
	  :FLAGS,
	  ha_eixo
	  :BOOLEAN,
	  inicio_menu,nlin_menu,ncol_menu
	  :INTEGER,
	END;

registro de escalas da malha	MALHA_DESENHO = RECORD
	  nome_malha : NOMES
	  deltax,deltay,deltaz :REAL;
	  intx,inty,intz :INTEGER,
	  indice_linha,indice_coluna :INTEGER,
	  fator_escala:integer,
	END;

tipos utilizados:

COORDENADA_TELA - registro de coordenadas x, y , inteiras

COORDENADA_PLANO - registro de coordenadas x, y, reais

LEITURA = (coordenadas, cursor)

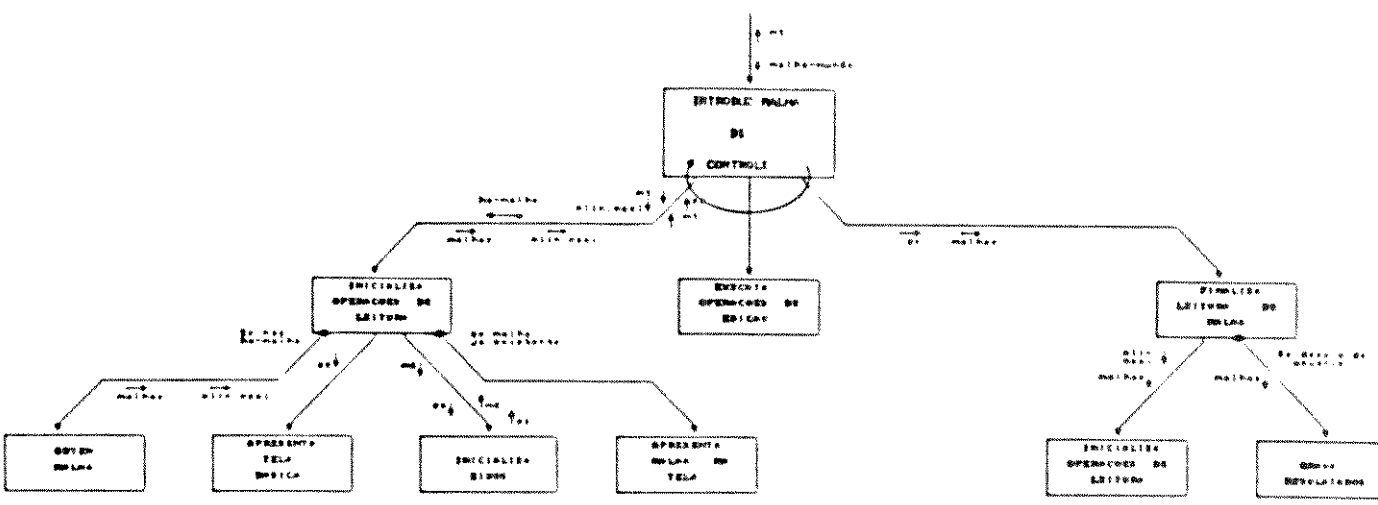
FLAGS - vetor de valores booleanos que indicam presenca de pontos em cada "dot" ocupado pelo cursor

Fig. 4.12 Estruturas de dados do programa da interface de introdução das malhas de controle.

4.4.3 Estrutura do Processo

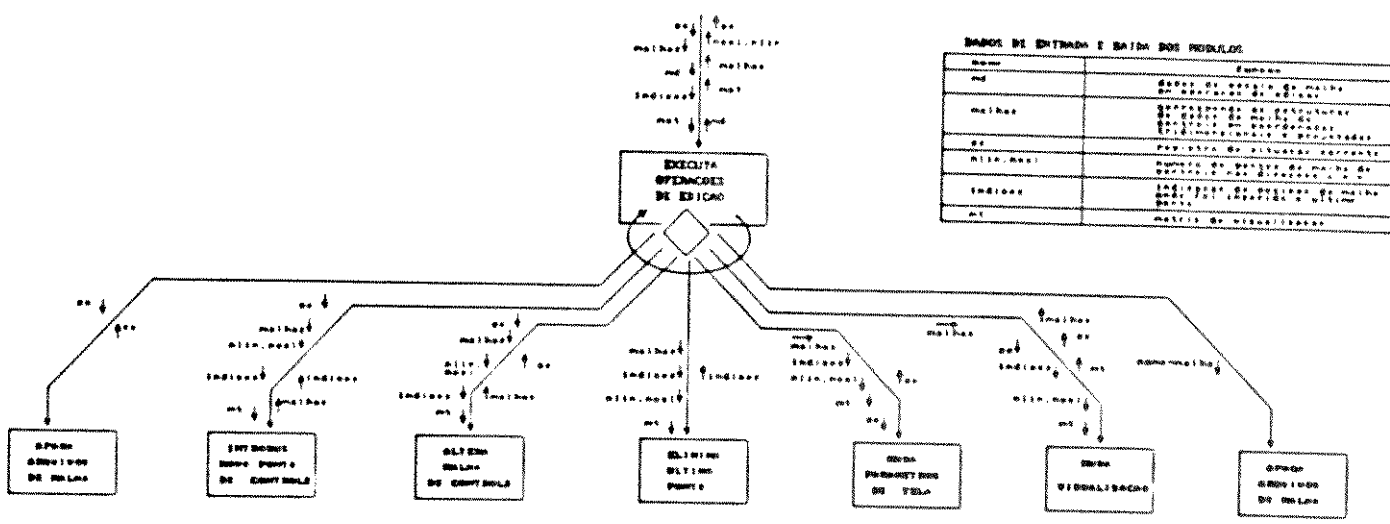
Basicamente, a interface permite ao usuário introduzir os valores dos pontos de controle da malha de duas maneiras: por cursor ou por coordenadas. Por coordenadas, o usuário entra com o valor real das coordenadas x, y e z de cada vértice de controle. Por cursor, pode utilizar o editor gráfico, que permite caminhar com o cursor sobre duas janelas que representam os planos xy e xz do sistema de coordenadas mundo (fig. 4.11). Assim, o usuário tem a possibilidade de elaborar os dados pensando em coordenadas x e y, seguida da altura do ponto com relação ao plano xy (coordenada z). Automaticamente , a imagem da malha de controle sendo introduzida é mostrada em perspectiva. A introdução de pontos se dá na ordem do mapeamento para o plano uv , isto é, para cada valor de índice na direção u da malha são introduzidos todos os vértices de controle na direção v. O processo é repetido para todos os valores de índices na direção u, em ordem crescente.

A figura 4.13 apresenta o diagrama básico da interface. O usuário pode, a qualquer instante, alterar os pontos já introduzidos da maha de controle, introduzir novos pontos, ou mudar a visualização da malha de controle. No final, a última visualização pode ser armazenada para geração da superfície sob aquele ponto de vista. Durante a edição pode-se mudar a escala do objeto ou a escala de tela. O apêndice 2 fornece o conjunto completo de operações possíveis de se manipular através da interface de entrada.



a)

Fig. 4.13a Diagrama de Estrutura Básico da Interface de entrada.



b)

Fig. 4.13 (continuação)
b Detalhamento primário da função Executa Operações de Edição

4.4.4 Conceitos Básicos envolvidos

A elaboração da Interface empregou basicamente um tratamento simples de janela, e as rotinas básicas de visualização e perspectiva para apresentação da malha projetada durante a introdução de dados.

4.4.5 Características de Implementação

Implementada em Turbo Pascal 3.0, possui uma organização de janelas fixas bastante simples. Traduções do programa para versões posteriores da linguagem (4.0 ou 5.0) permitiriam maior flexibilidade. A estrutura do programa admite modificações para manipulação de janelas sobrepostas, apoiadas por funções das versões mais novas da linguagem.

Os planos de edição do editor da interface permitem caminhar com o cursor enquanto se projeta a malha de controle. O "campo" do plano de edição na tela pode ser alterado, caso o cursor saia da região delimitada pelos contornos da janela. O programa gerencia tais "excursões".

Previamente às operações de edição o usuário deve selecionar se deseja trabalhar com uma nova malha ou se vai manipular uma malha de controle já armazenada. Para uma nova malha de controle necessita dizer com que nome deseja armazená-la.

A malha de controle é armazenada (se desejo do usuário) com o nome fornecido e extensão ".PDN", no disco corrente. Os dados de escala para o editor (malha-desenho) são armazenados em arquivo com o mesmo nome, e extensão ".MD". A matriz de

transformação inicial (para malhas novas) deve estar em disco, no arquivo *DEFAULT.MT*, e no caso do usuário desejar mudar a visualização, o resultado é armazenado num arquivo em disco com o mesmo nome da malha, e extensão *.MT*.

Se o usuário optar por uma malha nova e der o nome de uma malha já existente no disco corrente, ele deverá confirmar sua destruição antes de prosseguir. Se optar por malha já existente e der o nome de uma malha que não consta do disco corrente, deverá optar novamente e re-introduzir um nome de malha correto.

O arquivo de malha produzido pela interface e utilizado pelo modelador, tem como conteúdo os números de pontos da malha nas direções u e v de variação dos parâmetros, e os próprios pontos da malha.

O programa da interface possui aproximadamente 3000 linhas de código, e trabalha, se desejo do usuário, separadamente do programa do modelador e do programa de retirada de linhas escondidas.

No processo de projeto de um objeto, além de poder modificar interativamente o formato do objeto, o usuário necessita observar a superfície para poder identificar possíveis alterações. Nesse aspecto, a técnica de visualização deve permitir ao usuário ter uma visão clara do formato. Como a apresentação de todos os pontos gerados provoca um congestionamento de linhas que pode deturpar a visualização correta, é sempre desejável algum tratamento da imagem. Essa questão foi tratada de duas formas, resultando os programas que são apresentados no próximo parágrafo.

4.5 Interface de saída - Visualização com retirada de linhas escondidas.

Toda superfície gerada pelo modelador pode ser observada empregando dois diferentes métodos, apresentados a seguir.

4.5.1 Visualização com teste de visibilidade por coordenadas de tela

4.5.1.1 Descrição do Método

O primeiro programa para mostrar a superfície implementado, utiliza um método bastante simples de retirada de superfícies escondidas. A técnica de apresentação é por "lofting" e a imagem é um conjunto de segmentos de reta, e a verificação da visibilidade dos pontos ocorre no traçado de cada reta.

O método de verificação da visibilidade exige que a primeira "linha" da superfície esteja totalmente visível. A partir daí, suprime-se todo ponto da região de tela entre duas linhas já impressas. Esse é o critério de visibilidade para todo ponto avaliado da superfície (figura 4.14).

O objetivo do procedimento é descongestionar o cruzamento de linhas, o que permite maior facilidade, por parte do usuário, de identificar a forma final do objeto.

Assim, a primeira das técnicas empregadas apresenta a forma por lofting e é feita pela simples eliminação do cruzamento de linhas na figura formada.

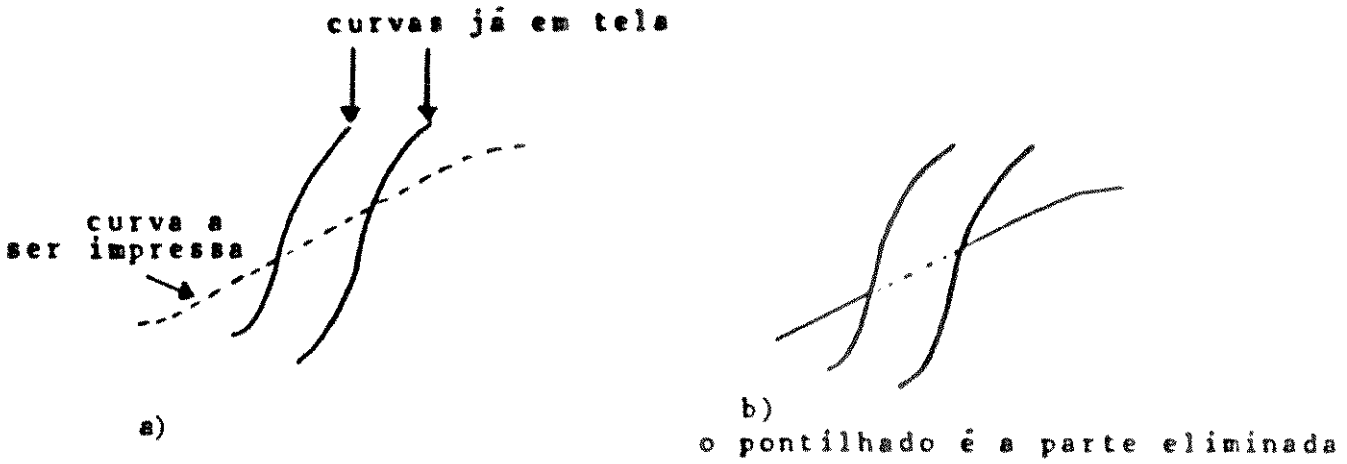


Fig. 4.14a situação de invisibilidade a ser tratada pelo algoritmo
b resultado do algoritmo

Para garantir que sejam eliminadas realmente as linhas da parte de trás do objeto, é necessário que a primeira "linha" da visualização esteja totalmente visível. Para determinar essa linha, a malha de controle é reorganizada pelos valores das coordenadas z, para que a superfície comece a ser gerada por sua parte visível.

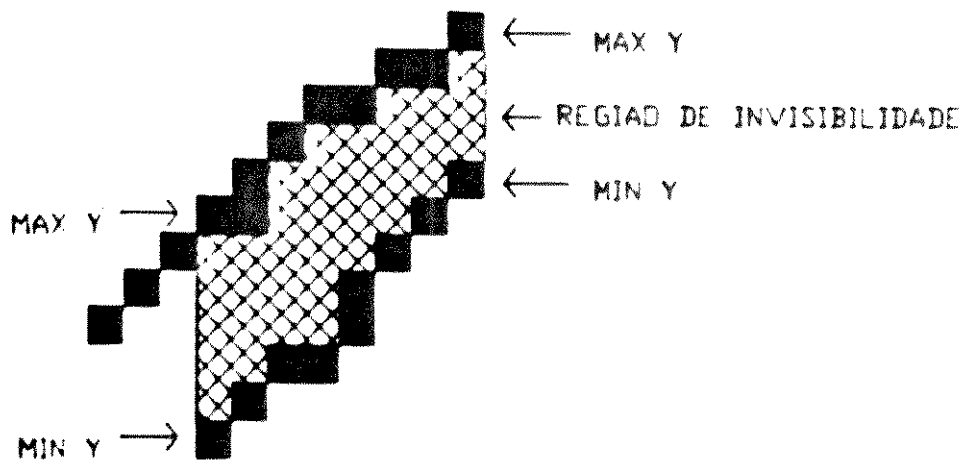
O processo consiste em:

1. Gravar o menor e maior valores de y apresentados em tela até o momento, para cada valor de x do campo de apresentação (tela). O objetivo é impedir que sejam apresentados os pontos com determinada coordenada x e coordenada y de valor no intervalo entre $y_{mínimo}$ e $y_{máximo}$ para aquele x. (figura 4.15a), ou

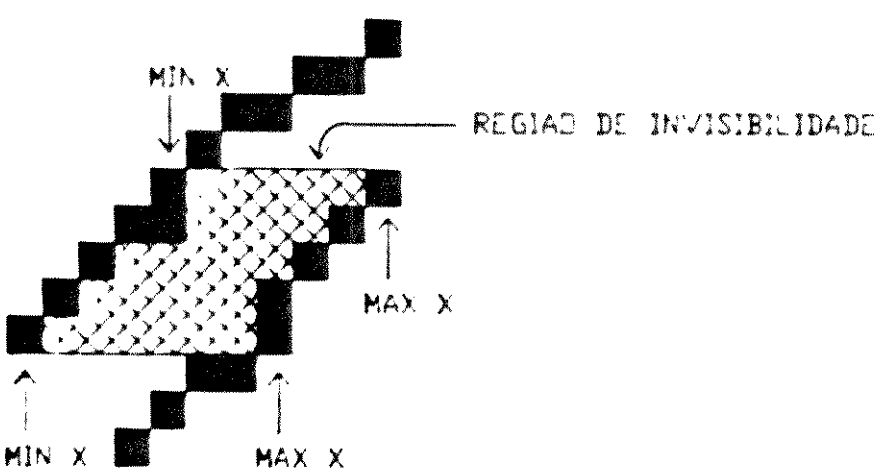
2. Executar o mesmo processo, só que para valores $x_{máximo}$ e $x_{mínimo}$ para todo y (figura 4.15b).

A escolha entre os procedimentos 1. e 2. deve ser feita de acordo com a característica de visualização da figura, isto é, em

algumas circunstâncias, como a da figura 4.16a, o método 2 causaria invisibilidade da região demarcada, o que não representa a realidade. Da mesma forma, o método 1 causaria invisibilidade da região demarcada da figura 4.16b, incorrendo num erro.

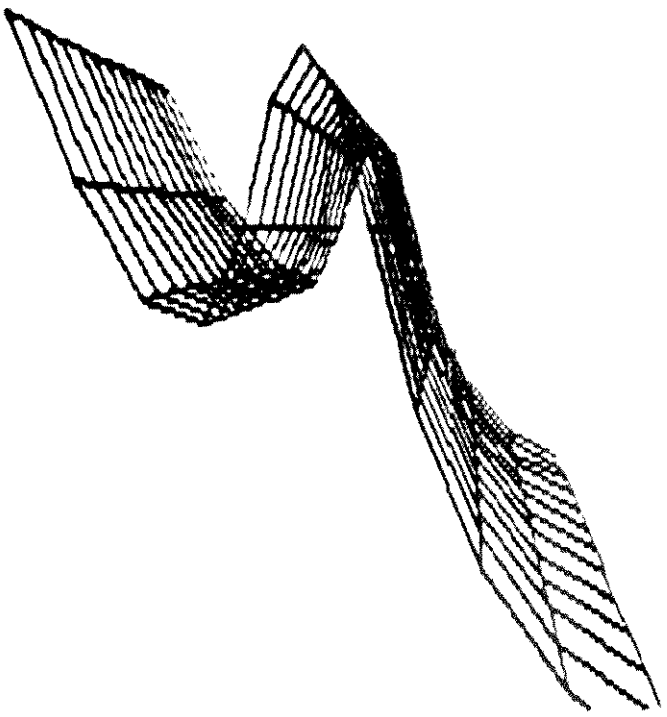


a)



b)

Fig. 4.15a Ilustração da determinação de invisibilidade por valores máximo e mínimo de coordenadas y, para um dado x.
b Ilustração da determinação de invisibilidade por valores máximo e mínimo de coordenadas x, para um dado y.



a)

b)

Fig. 4.16a Figura que necessita do processo 1 de determinação de linhas escondidas
b Figura que necessita do processo 2 de determinação de linhas escondidas.

A escolha entre os dois métodos é apoiada por um procedimento que verifica se houve inversão em y (assim foi chamada a situação da superfície da figura 4.16a), ou se houve inversão em x (situação da figura 4.16b), na superfície projetada. O resultado da aplicação do método pode ser observado na figura 4.14b.

Análise do Método: Esse método é rápido, relativamente a todos os demais que eliminam superfícies escondidas, e só trabalha com as coordenadas de tela da figura, mas necessita de um tipo de consistência incompatível com alguns dos objetivos do sistema em construção. Ele exige que ao menos uma das "linhas" da superfície seja totalmente visível, o que nem sempre é possível (fig. 4.17). Além disso, apresenta o pedaço de superfície por "lofting", e o quadriculado na maioria das circunstâncias é uma ferramenta bem melhor de compreensão da forma. Este método não prevê e não é extensível para cenas, ou seja, apresentação de mais de um objeto de uma única vez.

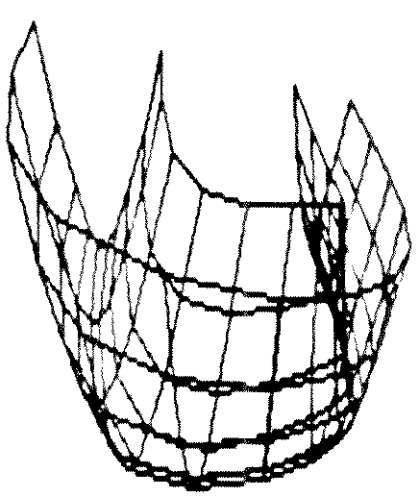


Fig. 4.17 Objeto que não possui a consistência necessária para visualização pelo primeiro método de apresentação

O processo para determinar a visibilidade implica num conjunto de operações, que incluem o pré-processamento da malha de controle e a obtenção da visibilidade enquanto a superfície é gerada. Tais operações correspondem às seguintes tarefas:

- determinação da ordem de geração da superfície: é necessário garantir que a primeira curva seja totalmente visível; para isso um procedimento re-ordena a malha de controle para a geração da superfície na sequência correta.

- determinação da inversão: Aqui convém fornecer uma idéia geral de como funciona o processo de visualização. Para se controlar os pontos a serem impressos ou não é necessário manter um buffer interno que armazena, para cada abscissa, as duas ordenadas que limitam o "intervalo de visibilidade", uma vez que a tela gráfica é discreta. Por exemplo, dado o trecho da imagem da figura 4.18,a impressão posterior de pontos com x = 10 e y entre 16 e 20 resulta numa situação de invisibilidade. Assim, para cada x, há um limite superior e um inferior para y. Fora desses limites, um ponto é visível. Dentro desses limites, não. Em algumas condições é mais adequado executar o procedimento considerando os limites dos valores de x para cada y, ou seja, por processo idêntico ao demonstrado acima, apenas trocando o teste para valores de x entre máximo e mínimo.

Assim, o programa verifica se houve inversão, determinando se a malha possui mudança de direção das arestas de controle, para cada linha da malha. Esse procedimento fornece a informação para escolha entre formar o buffer das coordenadas x ou y. Se ocorre inversão em y, a visibilidade é controlada em função dos valores de x, para cada y.

- <u>verificação da visibilidade</u>: De acordo com o procedimento descrito acima, a determinação da visibilidade implica em duas atividades: a primeira, determinar se um ponto a ser impresso está ou não no "intervalo de invisibilidade". A segunda é a atualização do buffer toda vez que um ponto, decidido como visível, é impresso. As duas operações devem ser controladas a partir do procedimento de traçado de reta. As retas vão sendo traçadas ponto a ponto, cada um deles sendo aceso se for visível. A atualização do buffer é feita quando o ponto é incluído na imagem.

Fig. 4.18 Trecho de imagem refletindo possível situação de tela.

4.5.1.2 Estruturas de Dados

Para controlar o "trecho" visível da imagem é necessário armazenar num vetor, os valores das coordenadas já impressas para cada coluna (linha) da tela. Tal vetor deve armazenar ora coordenadas x, ora coordenadas y, conforme o tipo de imagem a ser produzida. A figura 4.19 mostra a organização da estrutura que desempenha essa tarefa.

Estrutura de dados - Visualizacao por Coordenadas de Tela			
entidade	**sinonimos**	**objetivo**	**organizacao**
vetores de maximos e minimos	max_x e min_x ou max_y e min_y	Armazenar coordenadas maximas e minimas dos pontos ja impressos da imagem da superficie. Armazena maximo x e maximo x para cada y de tela ou armazena maximo y e minimo y para cada x de tela que ja tenha pontos acessos. Uma ou outra abordagem depende da aparencia da superficie	Vetorial , onde cada elemento e um valor de coordenada de tela, indexado pela outra coordenada do mesmo ponto. Assim , se o ponto (10, 20 e ponto de maximo valor de x, para o y=20, tem-se max_x(20) = 10.

Implementacao Utilizada	
entidade	**Implementacao Pascal da Estrutura correspondente**
vetores de maximos e minimos	inv = (vps,xis,vz,nenhuma) para_cada_x = ARRAY [-zerox..zerox+1] OF INTEGER para_cada_y = ARRAY [-zeroy..zeroy+1] OF INTEGER min_max = RECORD case inversao:INV OF vps :(max_y :PARA_CADA_X; min_y :PARA_CADA_X) xis :(max_x :PARA_CADA_Y; min_x :PARA_CADA_Y) nenhuma :(max_y :PARA_CADA_X; min_y :PARA_CADA_Y) END;

constantes utilizadas:

zerox :ponto da tela correspondente a origem em x

zeroy :ponto da tela correspondente a origem em y

Fig. 4.19 Estruturas de Dados - primeiro algoritmo de visualização

4.5.1.3 Estrutura do Processo

A figura 4.20 apresenta a organização estrutural do programa de visualização por teste de coordenadas de tela.

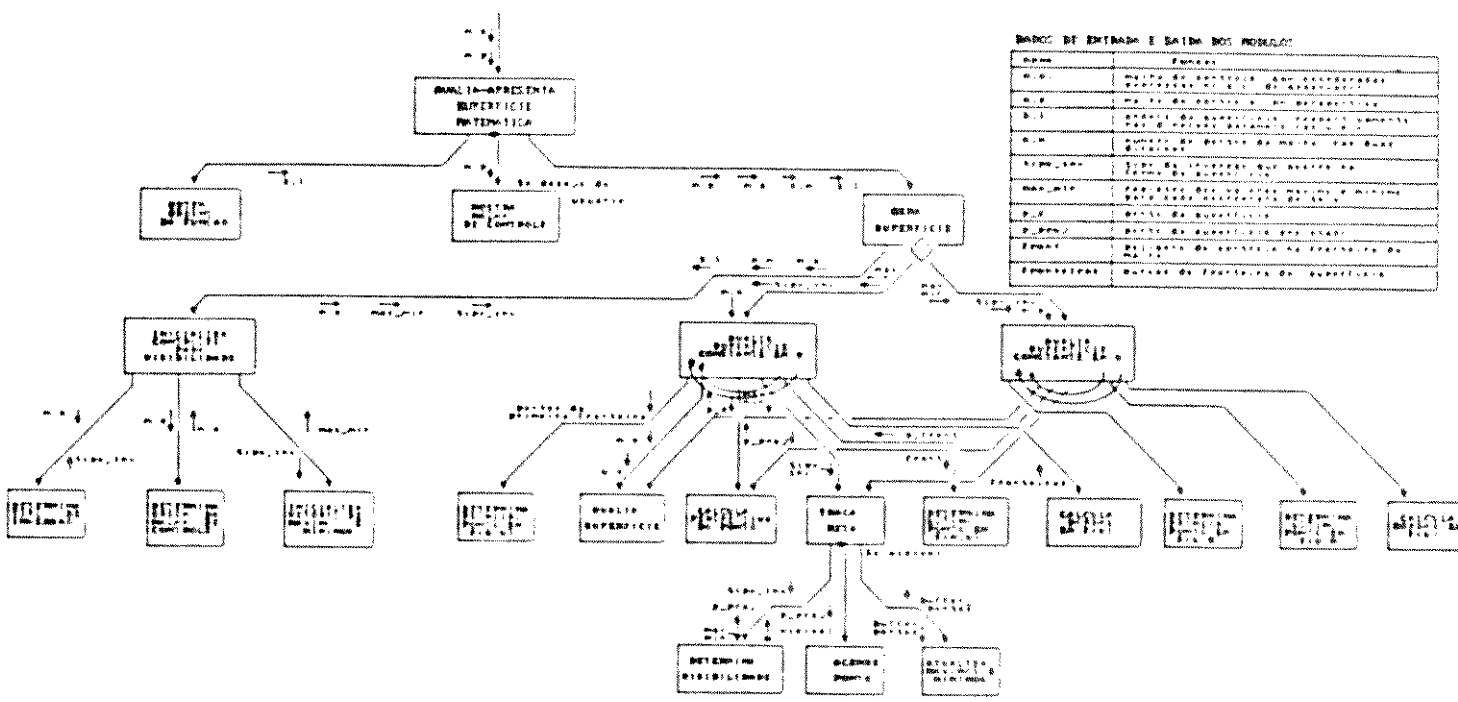


Fig. 4.20 Diagrama de Estrutura do primeiro algoritmo de visualização

O teste de visualização é executado durante a geração da superfície. Cada ponto gerado do objeto forma um segmento de reta com o ponto gerado anteriormente (aproximação poligonal da

superfície). Tal reta é gerada ponto a ponto, e cada um deles é testado quanto à visibilidade. Pontos visíveis são armazenados no vetor de máximos e mínimos.

4.5.1.4 Conceitos Básicos envolvidos

O método de retirada de linhas escondidas trata o vídeo (saída) pela geração de cada um dos pontos de tela das retas que aproximam a superfície modelada. O aspecto de implementação implica na utilização de um algoritmo incremental para traçado de reta, ao invés do uso do recurso gráfico da própria linguagem de programação onde o sistema foi desenvolvido.

4.5.1.5 Características de Implementação

Com a "aproximação" da superfície por segmentos de reta, obtém-se um ajuste para visualização, que reduz cada "linha" da superfície a uma poligonal de precisão controlável. Como cada segmento de reta resultante é discretizado para apresentação em tela, pode ocorrer de um mesmo segmento de reta possuir duas abscissas diferentes para uma mesma ordenada ou vice-versa. Nesse caso, um ponto, se aplicado no algoritmo de visualização, poderia "esconder" outro ponto da mesma reta, o que geraria uma inconsistência na imagem. A solução é atualizar o buffer de imagem apenas no final do traçado de cada reta da superfície. Por esse motivo, um buffer interno à rotina do traçado de retas é responsável por guardar os valores de pontos da mesma reta e só

atualizar a estrutura de dados após o traçado da reta toda, na região ocupada pelo segmento traçado. A presença do buffer interno pode ser notada no diagrama da figura 4.20.

Dado que apenas um dos dois tipos de verificação é empregado a cada vez, isto é, uma superfície possui um ou outro tipo de inversão, não há necessidade de reservar espaço de memória para os dois buffers (o de extremos de x e o de extremos de y). Um só par de vetores resolve os aspecto de armazenamento. Assim, em termos de implementação física, o coerente é optar por uma estrutura que reserve espaço de memória para apenas um dos buffers, a cada execução do programa, conforme a característica da superfície a ser apresentada. No Pascal, a estrutura de registro com variante (cláusula CASE) implementa essa característica (ver figura 4.19).

Uma limitação inerente deste método de retirada de linhas escondidas é o fato de que a apresentação é necessariamente por "lofting", isto é, não se pode usar o recurso do "quadriculado" para observar a figura. Uma outra limitação vem da coerência de imagem exigida pelo método, isso é, da necessidade de que as duas primeiras "linhas" da imagem sejam necessariamente visíveis. Tais ocorrências apontam para a necessidade, neste sistema gráfico, de métodos de maior flexibilidade no auxílio à identificação correta da forma obtida, durante o projeto da superfície.

Assim, uma outra técnica para retirada de superfícies escondidas foi efetivada, com o objetivo de melhorar a visualização do objeto, ou seja, foi elaborada uma alternativa para a implementação do módulo de visualização de superfície do

diagrama da figura 4.3. Essa alternativa gerou um programa que é apresentado a seguir.

4.5.2 Interface de saída - Visualização com teste de visibilidade pelo método de Ohno

4.5.2.1 Descrição do Método

A segunda alternativa de visualização com retirade de linhas e superfícies escondidas foi incorporada ao sistema pela implementação de um método divulgado na literatura [Oh 83].

O algoritmo consiste em considerar o "quadriculado" da avaliação da superfície como um conjunto de quadriláteros, cada um deles composto de dois triângulos, os quais podem ou não esconder qualquer ponto da superfície.

O objetivo do algoritmo é definir quais pontos da rede formada pelo quadriculado são escondidos ou não pelos quadriláteros. Cada reta formada por dois pontos do quadriculado é então verificada quanto à visibilidade de seus extremos. Se ambos forem visíveis, ela é totalmente mostrada. Se ambos forem invisíveis, é totalmente escondida. Um extremo de uma aresta sendo visível e outro invisível exige um procedimento de bissecção para encontrar o trecho visível do segmento.

Primeiramente, os dois triângulos de cada quadrilátero são classificados segundo as tabelas que podem ser vistas no apêndice 3. Os vértices do quadrilátero são aplicados ao determinante, cujo sinal indica se sua disposição é horária ou anti-horária. Sendo 1, 2, 3 e 4 os vértices projetados do quadrilátero, a

classificação se dá verificando se são horários os triângulos 123, 134, 234 e 124. Para cada triângulo ijk o determinante é dado por:

$$A_{ijk} = (1/2) \begin{vmatrix} i_x & i_w & 1 \\ j_x & j_w & 1 \\ k_x & k_w & 1 \end{vmatrix}$$

Se $A_{ijk} < 0$ então a disposição dos vértices é horária. Se $A_{ijk} > 0$ então a disposição dos vértices é anti-horária.

A segunda figura do apêndice 3 apresenta as classificações de quadriláteros de acordo com todos os possíveis padrões (51 ao todo, apresentados na primeira figura do mesmo apêndice). Quando ocorre um cruzamento de arestas de um quadrilátero (é possível na projeção da superfície), então há a necessidade da determinação do quinto ponto, intersecção das duas arestas, para cálculos posteriores.

O processo de verificar se todos os triângulos escondem todos os vértices é de tempo de processamento muito alto. Para minimizar esse tempo, o algoritmo prevê um pré-processamento da superfície já avaliada e projetada, com valores de pontos armazenados, que procura minimizar os quadriláteros a serem comparados com todos os pontos, e também minimizar os pontos a serem verificados no momento da apresentação.

Para isso, todo o espaço da superfície é subdividido em regiões chamadas "boxes", que são paralelepípedos no espaço do objeto, nos quais são registrados todos os quadriláteros que pertencem àquela região.

No momento da verificação da visibilidade, basta determinar

os "boxes" que estão no caminho retilíneo entre o ponto sendo verificado e o observador (raio que projeta o ponto na tela). Todos os quadriláteros que podem esconder o ponto tem que estar nesses paralelepípedos. Os demais não precisam ser testados. Isso reduz bastante o número de quadriláteros a ser verificado a cada teste de visibilidade. Para otimizar ainda mais a verificação de visibilidade para cada box, os quadriláteros são ordenados pelo valor máximo da coordenada y projetada, em ordem decrescente. Assim, se um quadrilátero de um box não esconde o ponto porque seu y máximo é menor que o y do ponto, nenhum outro quadrilátero do box poderá fazê-lo, já que possuem coordenadas y ainda menores, e já se pode passar para verificação do próximo box.

No passo seguinte, o algoritmo prevê a determinação da visibilidade de cada ponto da rede de quadriláteros ("grid"). Posteriormente , realiza o traçado das arestas e trechos de arestas visíveis.

Para determinar a visibilidade de um ponto com relação a um quadrilátero, faz-se a comparação com os dois triângulos classificados do quadrilátero. Para cada triângulo, verifica-se:

1. se ele pertence à região projetada do triângulo. Isso é fácil de determinar, já que os vértices do triângulo estão classificados no sentido anti-horário.
2. Se a verificação 1 for positiva, é necessário determinar se o plano a que o triângulo pertence, no espaço, esconde o ponto, isto é, se está entre o observador e o ponto.

Somente se a verificação 2 também é positiva é que se conclui que o ponto é invisível.

O próprio autor do algoritmo propõe uma melhoria de velocidade que implica em reunir no mesmo passo de algoritmo: - a determinação da visibilidade de cada ponto da rede ("grid") e - a apresentação das arestas e trechos de arestas visíveis.

Análise do algoritmo: Não possui restrições quanto ao formato da superfície, ao número de objetos de uma cena, nem ao método de representação da superfície. Ainda assim, conserva problemas de ocupação de memória e tempo de processamento, muito comuns a esse tipo de algoritmo, e que acabam refletindo em limitações com relação ao tipo de cena permitida, de acordo com a máquina sendo utilizada. Cobre também aqueles objetos que não podem ser mostrados pelo primeiro algoritmo, no que se refere à apresentação de objetos sem "linhas" completamente visíveis (ver figura 4.21).

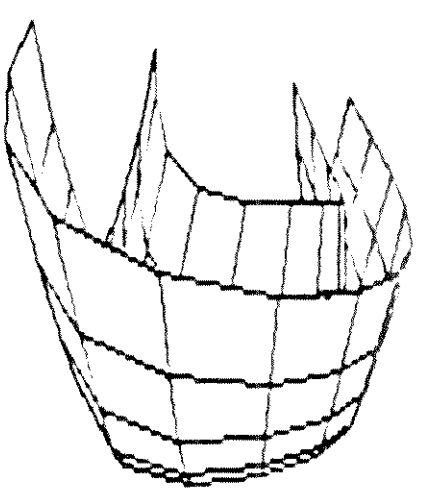


Fig. 4.21 Efeito da aplicação do segundo método de eliminação para afigura 4.17.

Para o método de Ohno para vizualização de superfícies

paramétricas, a superfície deve ter sido avaliada por quadriculado, ou seja, são obtidos pontos da rede B-spline, por incrementos sucessivos dos valores dos parâmetros u e v, a partir de uma determinada precisão, como apresentado no parágrafo 4.3.

Com base nesse quadriculado, o método para visualização de superfícies consiste, em termos conceituais, de dois passos básicos:

1. Classificar a projeção de cada quadrilátero da rede, de acordo com os dois triângulos que tal quadrilátero forma, de acordo com uma tabela básica de 51 padrões, que é apresentada no apêndice 3.

2. Obter a visibilidade de cada ponto a ser mostrado pela verificação da possibilidade de cada quadrilátero da rede escondê-lo. Um ponto é invisível se qualquer dos quadriláteros da superfície puder escondê-lo. O processo de determinação da visibilidade é aquele descrito acima.

A figura 4.22 mostra a organização do módulo APRESENTA SUPERFÍCIES de acordo com as tarefas explicadas acima.

O módulo MOSTRA QUADRILATEROS é responsável por realizar as seguintes tarefas:

1. Verificar a visibilidade dos pontos da rede (ou "grid")

2. Traçar as arestas da rede.

O autor do algoritmo recomenda (embora ele não implemente assim), que as tarefas 1 e 2 sejam agrupadas, isto é, assim que um ponto é verificado quanto à visibilidade, as arestas das quais faz parte devem ser traçadas. O modo como o autor implementa, e o modo como ele recomenda são refletidos nos diagramas do apêndice

4. O SMS opta pelo recomendado, uma vez que a justificativa do autor se verifica, isto é, o tempo total do programa não diminui significativamente, mas a imagem do objeto começa a ser vista mais rapidamente, o que para o usuário é mais adequado.

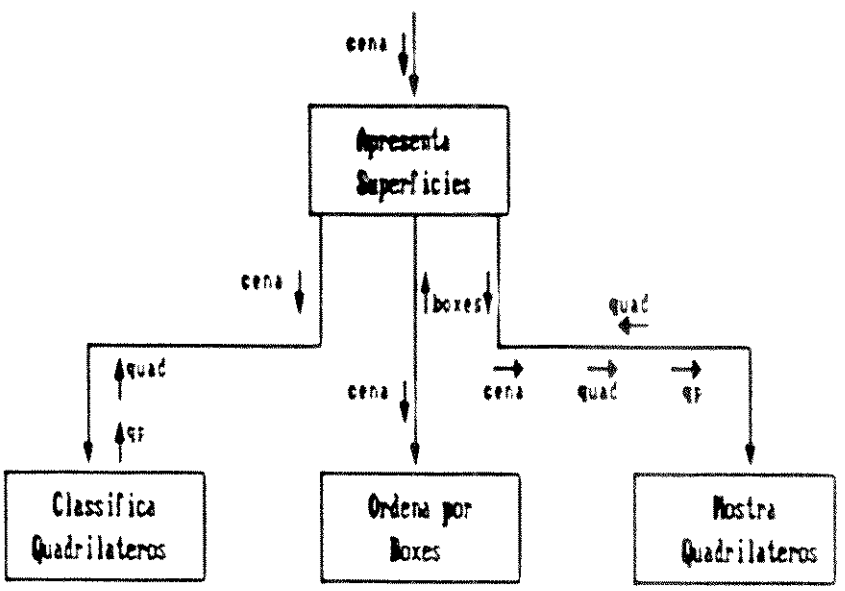


Fig. 4.22 Diagrama de estrutura básico da apresentação de superfícies pelo método de Ohno

Para traçar cada aresta da rede , três condições devem ser consideradas:

C1. Se os dois extremos da aresta são invisíveis.

C2. Se os dois extremos da aresta são visíveis.

C3. Se um extremo da aresta é visível e outro invisível.

Admite-se que na condição C1 não há o que traçar. Na condição C3, é necessária uma bissecção, tanto no espaço como no plano de tela, para obter qual o trecho da aresta é visível, para poder traçá-lo. Na condição C2, o autor escreve que a aresta deve ser inteiramente traçada. Entretanto, nem sempre isso é verdade.

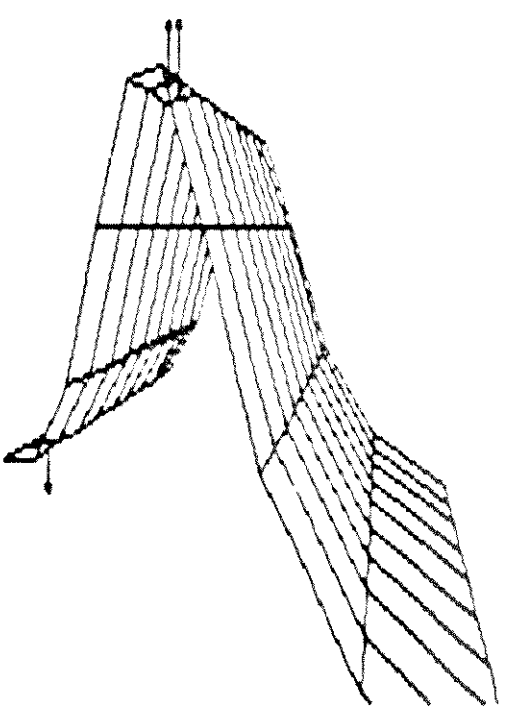
Como mostra a figura 4.23a, conforme a precisão com que a superfície é calculada, um ponto visível pode ser origem de um trecho de aresta invisível. Dessa forma, uma aresta com os dois extremos visíveis nem sempre é totalmente visível para poder ser traçada diretamente.

Uma maneira de solucionar o problema é o aumento da precisão no cálculo da superfície, que suaviza mais a forma da imagem, evitando a condição do erro do algoritmo. Entretanto, para aplicações em microcomputador, o incremento de memória necessário, e a lentidão da apresentação provocados pelo aumento da precisão devem ser evitados.

Uma solução alternativa, e adotada pelo SMS, é verificar a visibilidade real do extremo da aresta, isto é, testa-se a visibilidade de um ponto "vizinho" dos extremos das arestas que tem inicialmente os dois extremos visíveis. Se um "vizinho" for invisível, a aresta é considerada na condição C3, se ambos forem invisíveis, a aresta é considerada na condição C1. O erro é então satisfatóriamente corrigido, como mostra a figura 4.23b, embora acarrete um aumento do tempo de execução do programa de visualização. Os diagramas do Apêndice 4 refletem o módulo que contorna o "erro" do algoritmo.

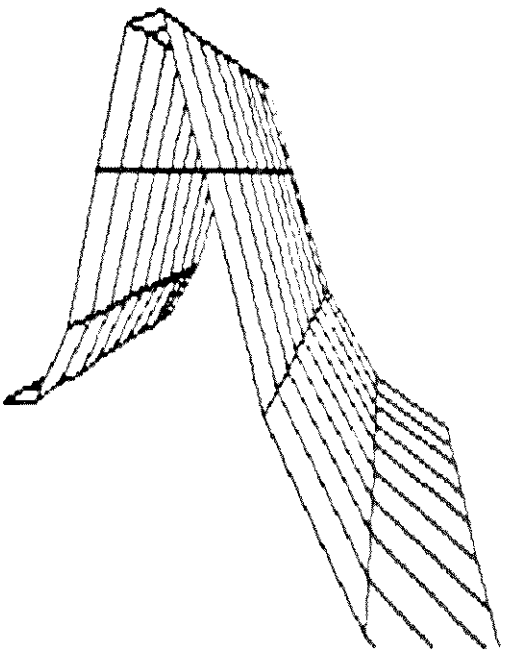
Uma outra condição não considerada pelo algoritmo e nem pelo

SMS é a possibilidade de uma mesma aresta possuir vários trechos intercalados de visibilidade e invisibilidade. Dessa forma, o algoritmo de retirada de linhas e superfícies escondidas tem uma única coerência de imagem exigida: uma aresta só pode possuir um trecho invisível e um trecho visível, no máximo. O método é extensível para incluir a flexibilidade de não exigir tal coerência, porém seria necessário gerar cada aresta ponto a ponto, verificando a visibilidade de cada ponto. Como o procedimento de verificação de visibilidade é o que imprime maior lentidão ao programa, isso acarretaria um alto custo computacional.



a)

Fig. 4.23a Efeito do erro no algoritmo de Ohno



b)

Fig. 4.23 (continuação)
b Efeito da correção adotada sobre a fig. 4.23a

O sucesso da aplicação do algoritmo, que originalmente foi executado num computador de grande porte, em microcomputador está centrado numa boa estrutura de software e num conjunto de estruturas de dados que reflita a estrutura da cena. Os próximos parágrafos apresentam a versão de ambos os aspectos do desenvolvimento.

4.5.2.2 Estruturas de Dados

O programa que cuida da visualização com retirada de superfícies escondidas deve se preocupar em estruturar a cena a ser montada. Isto é, considerar a possibilidade de uma cena complexa, com vários objetos.

A estrutura de dados deve ser capaz de registrar os quadriláteros pertencentes a cada "box", as classificações de cada quadrilátero, e os pontos de cada superfície da cena, tanto tridimensionais, quanto projetados.

Para cada superfície, é necessário armazenar os pontos inicial e final da região do espaço por ela ocupada (pi e pf), bem como o comprimento dos "boxes" em cada direção (deltax, deltay, deltaz). Também é preciso armazenar os boxes que contém quadrilláteros, e o registro destes para cada box.

Para cada quadrilátero é necessário armazenar a classificação (duas ao todo), e o valor da coordenada y máxima da projeção de seus vértices.

Por questões de ocupação de memória que posteriormente serão discutidas no presente texto, a implementação das estruturas de dados se dá por alocação dinâmica de variáveis.

O esquema geral das estruturas de dados é apresentado na figura 4.24 e a definição de cada estrutura é dada na figura 4.25.

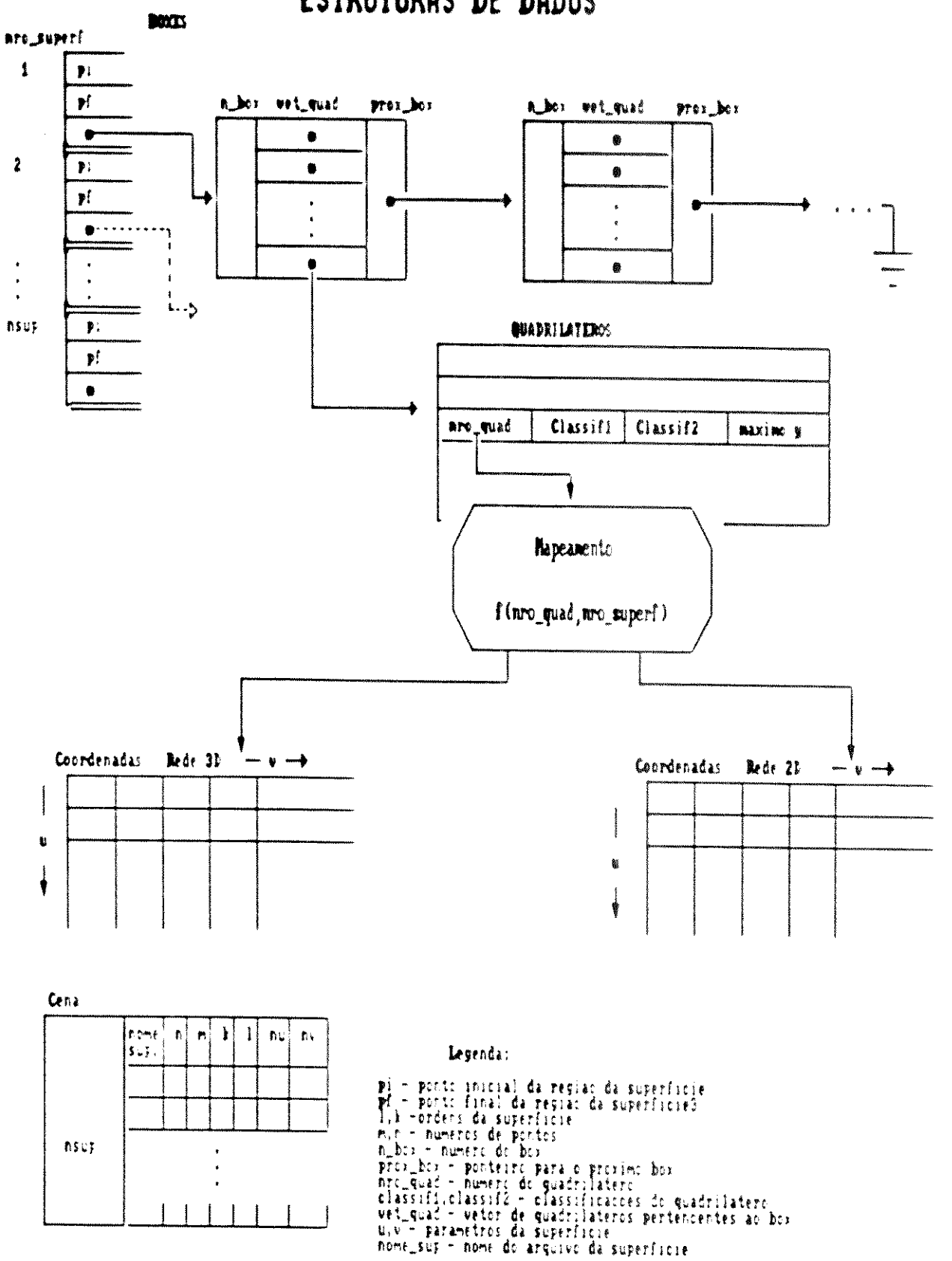


Fig. 4.24 Esquema das estruturas de dados do programa de retirada de linhas e superfícies escondidas pelo método de Ohno

Estruturas de dados - Método de Ohno			
entidade	sinonimos	objetivo	organizacao
quadrilateros	quadri, quad	Armazenar, para cada quadrilatero, suas classes e o valor da maior coordenada y dos seus vertices projetados	Vetorial, onde cada elemento e um registro contendo o numero do quadrilatero, suas classificacoes e o y maximo
"quinto ponto" de quadrilatero	qp	Armazenar os pontos de cruzamento de linhas dos projetores de quadrilatero quando ocorre.	Vetorial, onde cada elemento do vetor contem o numero do quadrilatero onde ocorre cruzamento e as coordenadas x e y do projetar do ponto.
Cena	reg_cena	Registrar as superficies que pertencem a cena sendo manipulada	Registro, contendo o numero de superficies da cena e um vetor que armazena dados de cada cena: seu nome, o numero de pontos da malha de controle, as ordens da superficie nas duas direcoes de variacao dos parametros e a precisao da superficie
Catalogo de redes bi-dimensionais e tri-dimensionais	patch_2i patch_3i	Armazenar os enderecos de todas as redes avaliadas pertencentes a uma cena e suas imagens.	Vetorial, onde cada elemento aponta para as estruturas de dados de rede, projetada ou espacial (ver estruturas de superficie 2i e 3i).
Tabela de classificacoes de quadrilateros	tab_classe	Armazenar as classes possiveis para todos os tipos de quadrilateros.	Conjunto (4 dimensoes), indexado pelo sinal do determinante da area do quadrilatero. Cada elemento possui duas classificacoes.

Implementacao Utilizada	
entidade	Implementacao Pascal da Estrutura correspondente

quadrilateros	tipo_classe = (_123, _124, _134, _234, _12, _13, _14, _23, _24, _34, _1234, _1324, _1423, _2314, _2413, _3412, _0);
	vet_quad = ARRAY[1..max(quad)] of
	RECORD
	class:set of tipo_classe;
	ymax:real;
	END;
	ponto_quad = ^tipo_ponto;
	tipo_quad = ARRAY[1..max(quad)] OF ponto_quad;

"quinto ponto" de quadrilatero	vet_qp = ARRAY[1..max(qp)] of
	RECORD
	num_quad:integer;
	class_p: tipo_classe;
	END;
	ponto_qp = ^tipo_qp;
	qp : ARRAY[1..max(qp)] OF ponto_qp;

Cena	reg_sup = RECORD nome :STRING[8]; n_m_k :integer; nu_nv:integer; END; tipo_cena=RECORD nsup :integer; sup :ARRAY[1..nsup] OF reg_sup; END;

Catalogo de redes bi-dimensionais e tri-dimensionais	ponto_grid_2d = ^tipo_grid_2d;
	cena_2d = ARRAY[1..nsup] OF ponto_grid_2d;
	ponto_grid_3d = ^tipo_grid_3d;
	cena_3d = ARRAY[1..nsup] OF ponto_grid_3d;

Tabela de classificacoes de quadrilateros	tipo_tab = ARRAY[-1..1,-1..1,-1..1,-1..1] OF RECORD class:f1,class:f2:TIPO_CLASSE; END;

constantes utilizadas:

max1..2	numero maximo de divisoes da rede
max_sup	numero maximo de quadrilateros
max_qp	numero maximo de quadrilateros com cruzamento de linhas
nsup	numero maximo de superficies na cena

Fig. 4.25 Definição das Estruturas de Dados do método de Ohno para retirada de linhas escondidas

4.5.2.3 Estrutura do Processo

A figura 4.22 apresentou o diagrama básico do programa de retirada de superfícies escondidas pelo método de Ohno. Cada um dos módulos foi desenvolvido separadamente, gerando uma estrutura própria que reflete a manipulação das estruturas de dados elaboradas.

O módulo CLASSIFICA QUADRILATEROS monta a estrutura de dados Quadriláteros. Para isso, define que os pontos $P_{i,j}$, $P_{i+1,j}$, $P_{i+1,j+1}$, $P_{i,j+1}$ formam um quadrilátero de número i. Utilizando os quatro vértices de cada quadrilátero, aplica a fórmula para obtenção da classificação, registrando-as. Também determina, para cada quadrilátero, o "y máximo", que é a máxima coordenada y do quadrilátero projetado. Faz isso para todos os quadriláteros de todas as superfícies. Posteriormente o módulo MOSTRA QUADRILATEROS pode alterar a estrutura de dados, substituindo a classificação obtida neste processo pela característica INVISIVEL, se a invisibilidade do quadrilátero for confirmada.

O módulo ORDENA POR BOXES monta a estrutura de dados Boxes. Para toda superfície da cena, subdivide o espaço ocupado e registra todos os quadriláteros que "pertencem" a cada subdivisão ("box"). A pertinência de um quadrilátero a um "box" implica na inclusão de determinado quadrilátero, ou parte dele, na região delimitada pelo "box". O vetor de quadriláteros formado para cada "box" é ordenado em função de y máximo.

O módulo MOSTRA QUADRILATEROS deve verificar a visibilidade de cada ponto da rede da superfície e traçar os trechos de arestas visíveis. O esforço maior está centrado na determinação

da visibilidade de um ponto. Em princípio, basta verificar cada ponto contra os quadriláteros dos "boxes" que estão no "caminho" do raio que liga o ponto a ser verificado ao observador. Verificar se um quadrilátero esconde um ponto implica em vários passos:

1. Se a coordenada y do ponto projetado for maior que o y máximo do quadrilátero, não só o quadrilátero não pode escondê-lo, como nenhum outro quadrilátero do mesmo "box" pode.

2. Caso a condição do passo 1 não se verifique, é importante testar se o ponto projetado pertence ao retângulo que envolve o quadrilátero. Se não pertencer, o quadrilátero não pode escondê-lo. Se pertencer, é necessário testar a posição do ponto com relação aos dois triângulos que formam o quadrilátero, pois o quadrilátero não é planar (passos 3 e 4). A classificação dos quadriláteros fornece os vértices de cada triângulo que o forma, organizados em sentido anti-horário.

3. Para testar um ponto contra um triângulo, primeiramente é necessário verificar se o ponto projetado está dentro do triângulo projetado, cujos vértices estão armazenados nas redes ("grids"). Isso é simples, uma vez que se conhece a organização anti-horária dos vértices de cada triângulo, imposta pela classificação.

4. Apenas se a condição do passo 3 for verificada deve-se testar se o ponto sendo verificado e o observador estão em posições opostas, no espaço, com relação ao plano que os vértices do triângulo formam. O ponto só pode ser considerado escondido pelo quadrilátero se, nesse passo, o resultado fornecer posição oposta dos dois pontos com relação ao plano do triângulo.

Os diagramas de estrutura dos módulos CLASSIFICA QUADRILATEROS, ORDENA POR BOXES e VERIFICAÇÃO DA VISIBILIDADE são apresentados no apêndice 5.

4.5.2.4 Conceitos Básicos envolvidos

Todos os procedimentos básicos envolvidos na determinação espacial da visibilidade das arestas das superfícies envolvem conceitos puramente relacionados à geometria analítica e descritiva, como a determinação de intersecção entre quadriláteros no espaço, intersecção de um segmento numa região do espaço, e outros de mesma natureza.

Do ponto de vista computacional, um conceito de estruturas de dados e seu relacionamento com a eficiência e flexibilidade do software resultante foi necessário para a decisão de qual estrutura utilizar no desenvolvimento. Os demais conceitos envolvidos estão refletidos da descrição do método de visualização.

4.5.2.4.1 Indexação de quadriláteros

Cada quadrilátero da rede superficial possui quatro pontos que são compartilhados com outros quadriláteros da mesma rede. Assim, para registrar os quadriláteros numa outra estrutura de dados, que não a estrutura do "Grid", implicaria num gasto muito grande de memória se isso fosse feito pelo armazenamento de todos os seus vértices espaciais e projetados. Dessa forma, é feito um

mapeamento dos quadriláteros para uma estrutura de dados linear, mapeamento este que permite recuperar os vértices de cada quadrilátero diretamente de estrutura de dados do Grid.

A figura 4.26a mostra a indexação, isto é, o mapeamento do número do quadrilátero de acordo com sua posição do grid da superfície.

A cada vez, os vértices:

armazenados nessa posição no "grid", formam um quadrilátero.

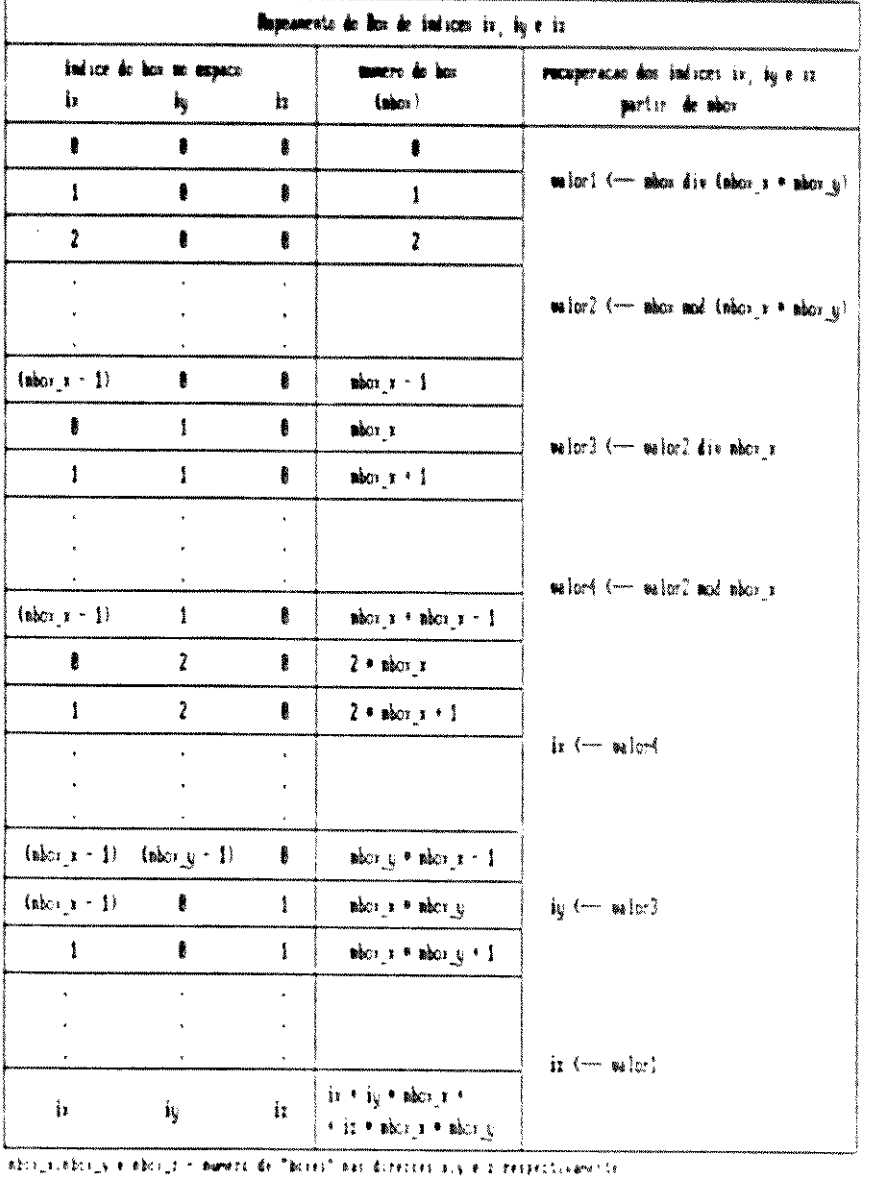
Cada quadrilátero possui um número, através do qual se acessa sua classificação, e valor de y máximo. Através do mesmo número, pode-se obter os índices dos vértices na estrutura de dados do "Grid", onde os valores das coordenadas estao armazenados.

Mapeamento do Quadrilatero de vertice i,j			
indice na estrutura da rede	numero do vertice	numero do quadrilatero (equal)	recuperacao do vertice i,j a partir de equal
i,j	1	(i*ny) + j	i <— equal div ny
i+1,j	2		
i+1,j+1	3		j <— equal mod ny
i,j+1	4		

ny - numero de divisoes da rede na direcao do parametro v

a)

Fig. 4.26 a Indexação da Estrutura de Dados Quadriláteros



Mapeamento de box de índices ix, iy e iz				
índice do box no espaço			número do box	recuperação dos índices ix, iy e iz
ix	iy	iz	(nbox)	partir de nbox
0	0	0	0	
1	0	0	1	valor1 (— nbox div (nbox_x * nbox_y)
2	0	0	2	
.	.	.		valor2 (— nbox mod (nbox_x * nbox_y)
.	.	.		
.	.	.		
(nbox_x - 1)	0	0	nbox_x - 1	
0	1	0	nbox_x	valor3 (— valor2 div nbox_x
1	1	0	nbox_x + 1	
.	.	.		
.	.	.		valor4 (— valor2 mod nbox_x
.	.	.		
(nbox_x - 1)	1	0	nbox_x * nbox_x - 1	
0	2	0	2 * nbox_x	
1	2	0	2 * nbox_x + 1	ix (— valor4
.	.	.		
.	.	.		
.	.	.		
(nbox_x - 1)	(nbox_y - 1)	0	nbox_y * nbox_x - 1	
(nbox_x - 1)	0	1	nbox_x * nbox_y	iy (— valor3
1	0	1	nbox_x * nbox_y + 1	
.	.	.		
.	.	.		
.	.	.		iz (— valor1
ix	iy	iz	ix * iy * nbox_x * iz * nbox_x * nbox_y	

nbox_x,nbox_y e nbox_z - numero de "boxes" nas direcoes x,y e z respectivamente

b)

Fig. 4.26 (continuação)
b Indexação da Estrutura de Dados Boxes

4.5.2.4.2 Indexação de "boxes"

Para cada subdivisão do espaço da superfície ("box"), tem-se que registrar todos os (números de) quadriláteros que pertencem a ele. Da mesma forma que para quadriláteros, os pontos extremos de um "box" são compartilhados com seus vizinhos. Além disso, nem todo "box" possui necessariamente algum quadrilátero. É necessário e possível, portanto, mapear cada "box" para uma

estrutura de dados linear. O ponto inicial da superfície é obtido ainda durante a avaliação da superfície, e é o ponto inicial do "box" 0. A partir daí, cada "box" é obtido somando-se a esse valor inicial, os valores de deltax, deltay e deltaz, que dimensionam o tamanho da aresta do "box" nas direções x, y e z do sistema de coordenadas do observador. Assim, todas as arestas dos "boxes" são paralelas aos eixos y e z do sistema de coordenadas do observador. Assim, todas as arestas dos "boxes" são paralelas aos eixos do sistema de coordenadas.

O vértice 1 de cada box é o ponto esquerdo inferior da região que ele delimita, expressos por três índices: ix, iy e iz. A figura 4.26 b fornece a indexação, isto é, o mapeamento dos índices de "boxes" para uma estrutura linear que permita identificar cada "box" da região ocupada pela superfície por um único número.

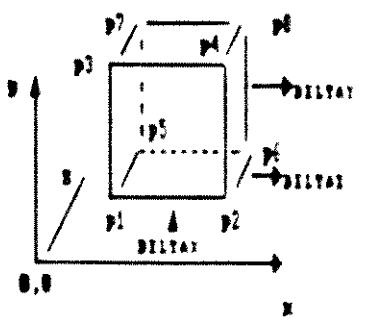
4.5.2.4.3 Obtenção dos valores extremos dos "boxes"

Para se obter os vértices do paralelepípedo no espaço que representa a região do "box", são necessários: seus índices ix, iy e iz, os comprimentos das arestas na direção x, na direção y e na direção z (deltax, deltay e deltaz) e o ponto inicial da região da superfície (pi).

O vértice 1 do paralelepípedo é dado por:

```
p1.x <-- pi.x + ix * deltax
p1.y <-- pi.y + iy * deltay
p1.z <-- pi.z + ix * deltaz
```

A figura 4.27a mostra a convenção do sistema para os números dos vértices. A partir de tal convenção e do valor de p1, as coordenadas dos demais vértices são facilmente obtidas. A figura 4.27b mostra a obtenção das coordenadas dos vértices do "box", a partir do valor de p1.



Obtencao das coordenadas dos vertices do "box" com indices ix, iy e iz					
numero do vertice (j)		1	. . .	6	8
coordenadas	pj.x	pi.x + ix * deltax	. . .	p1.x + deltax	p1.x + deltax
	pj.y	pi.y + iy * deltay	. . .	p1.y	p1.y + deltay
	pj.z	pi.z + iz * deltaz	. . .	p1.z + deltaz	p1.z + deltaz

pi - ponto inicial da regiao da superficie
deltax, deltay, deltaz - tamanho das arestas nas direcoes x,y e z respectivamente

Fig. 4.27a Convenção para os números dos vértices de um "box".
b Obtenção dos vértices do "box"

4.5.2.4.4 Verificação da pertinência de um ponto à região de um triângulo

Uma das operações básicas necessárias do SMS é a determinação da pertinência de um ponto projetado sobre o plano de tela à região de um triângulo também projetado.

A figura 4.28 mostra o triângulo de vértices A,B,C e as possíveis posições de um ponto P na região.

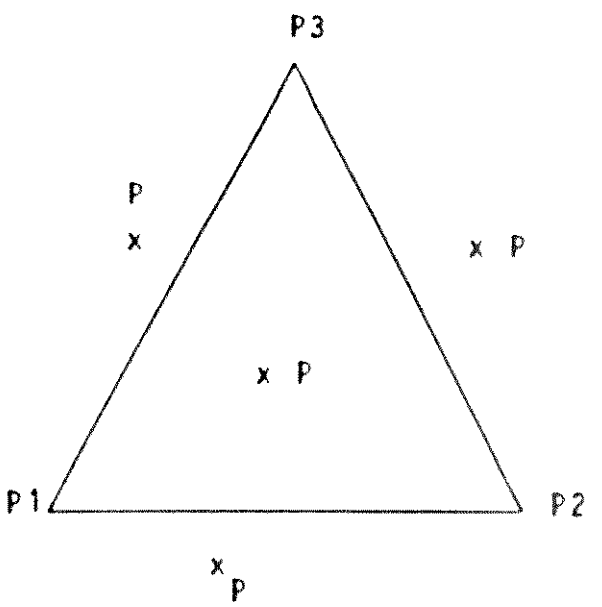


Fig. 4.28 Posições relativas de um ponto e um triângulo no plano.

Estando os vértices A, B e C assim organizados no sentido anti-horário, o seguinte algoritmo é válido:

Se PAB, PBC e PCA possuem sentido anti-horario então

P pertence à região do triângulo

senão

P não pertence à região do triângulo.

Para se determinar em que sentido do relógio estão organizados três pontos p1, p2 e p3 no plano, usa-se o determinante:

$$det = \begin{vmatrix} p1.x & p1.y & 1 \\ p2.x & p2.y & 1 \\ p3.x & p3.y & 1 \end{vmatrix}$$

Se det < 0 então p1 p2 p3 estão em sentido horário. Se det > 0 então p1 p2 p3 estão em sentido anti-horário. Na expressão, .x e .y significam, respectivamente, coordenadas x e y dos pontos.

4.5.2.4.5 Pertinência de um quadrilátero a um "box"

O critério de pertinência de um quadrilátero no espaço a uma região determinada ("box") é a seguinte:

Um quadrilátero pertence a um "box" se alguma de suas arestas intercepta alguma das faces do "box" ou todos os seus vértices estão inseridos na região do "box".

A inserção é fácil de verificar, bastando comparar as coordenadas dos quatro vértices do quadriláteros com as coordenadas dos vértice extremos do "box" (p1 e p8).

Quanto à intersecção, um possivel procedimento consiste em projetar ortogonalmente as faces do cubo sobre os planos xy, xz e zy do sistema de coordenadas do observador. Da mesma forma, projeta-se os quatro vértices do quadrilátero que está sendo verificado. Se, em qualquer das projeções, a face do cubo e o quadrilátero não se interceptarem, então quadrilátero e "box" não se interceptam no espaço (fig. 4.29). Como os "boxes" possuem

arestas paralelas aos eixos de coordenadas, as projeções ortogonais das faces do "box" sobre os planos coincidem duas a duas, sendo necessário verificar apenas três projeções de faces, e não seis.

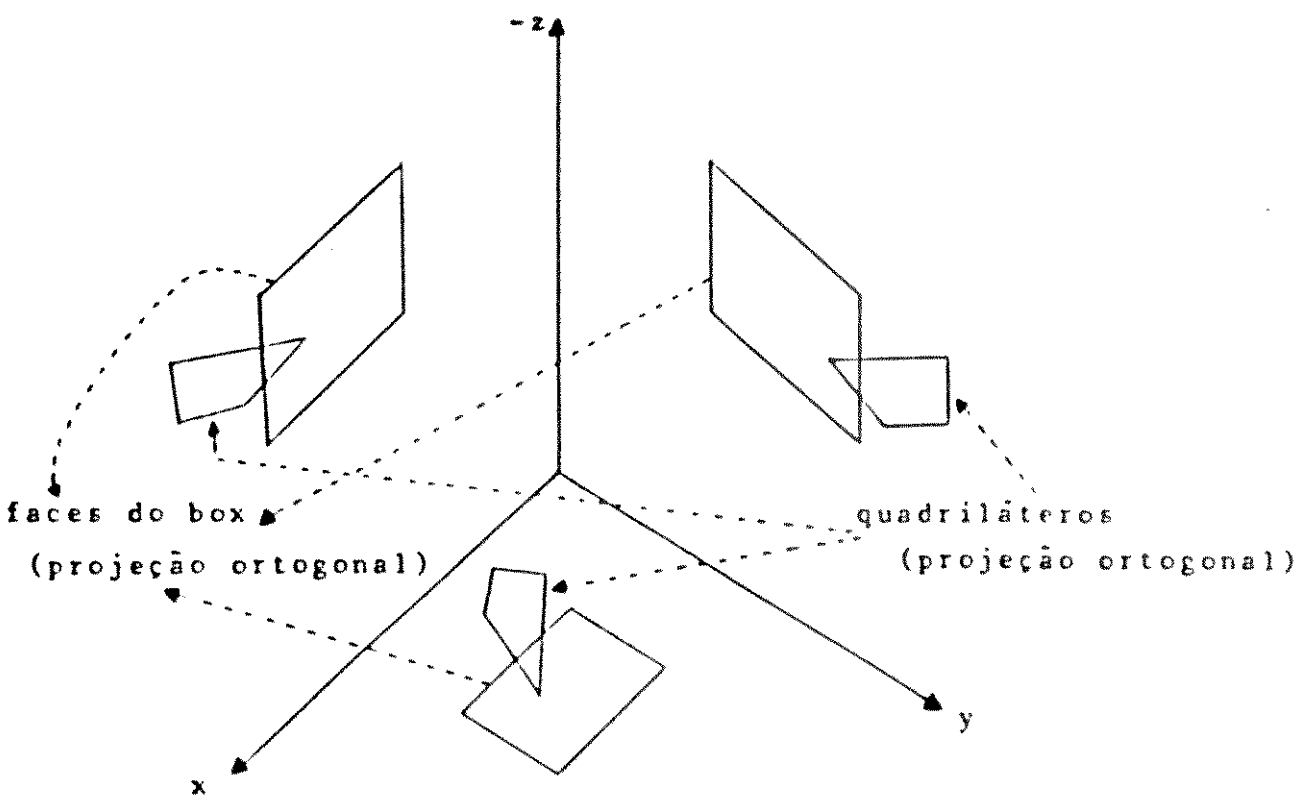


Fig 4.29 Ilustração da verificação de pertinência entre quadrilátero e "box"

Para verificar a intersecção entre face e quadrilátero projetados, é necessário verificar intersecção entre todas as arestas do quadrilátero e todas as arestas da face. Se ocorrer uma interseção na região dos segmentos, a intersecção dos dois entes (quadrilátero e face) ocorre (fig 4.30).

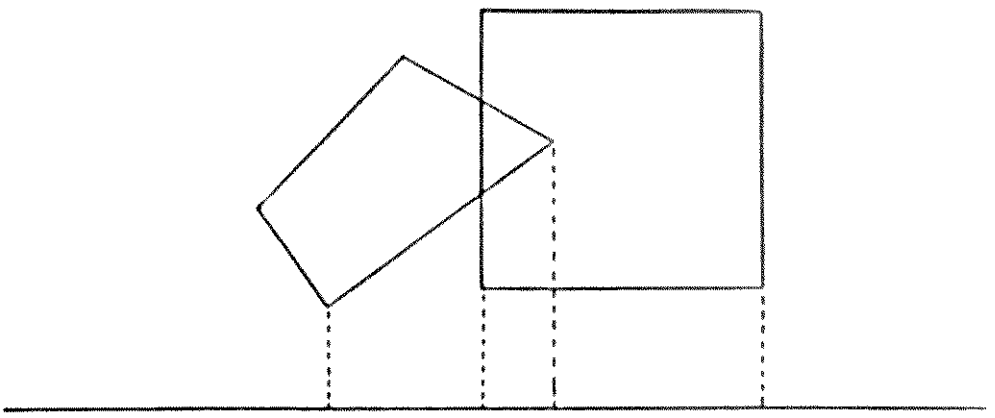


Fig. 4.30 Ilustração da interseção entre quadrilátero e face projetados.

Outros conceitos foram utilizados na elaboração dos algoritmos básicos de manipulação, como, por exemplo, equação paramétrica da reta no plano e no espaço, equação do plano, intersecção entre duas retas no plano, intersecção entre retas no espaço, intersecção entre reta e plano, posição relativa entre ponto e plano. Tais procedimentos não serão apresentados aqui por serem amplamente divulgados e conhecidos.

4.5.2.5 Características de Implementação

Os procedimentos para retirada de linhas e superfícies escondidas são essencialmente lentos. Muito embora a avaliação tenha sido otimizada e o método de Ohno, pela própria estrutura, tenha características de diminuição do tempo de processamento, ainda assim o algoritmo é lento para aplicação em

microcomputador. A formação da imagem de uma superfície a partir dos pontos da rede já avaliados é lenta.

Entretanto, o método é bastante flexível, permitindo apresentar objetos diversos, que serão mais perfeitos conforme respeitarem a coerência de imagem exigida pelo método e apresentada na secção 4.5.2.3. Permite cenas complexas, limitadas apenas pela quantidade de memória de dados disponível no computador utilizado.

Em virtude da quantidade de dados manipulada, todas as estruturas de dados com razoável necessidade de memória foram implementadas utilizando alocação dinâmica de variáveis, com os tipos definidos na secção 4.5.2.2. Assim, a declaração de variáveis para utilização no programa, de acordo com os tipos anteriormente definidos, fica:

```
VAR

  boxes: TIPO_BOX;
  quad:TIPO_QUAD;
  cena: TIPO_CENA;
  patch_2d: CENA_2D;
  patch_3d: CENA_3D;
```

O acesso para a leitura ou escrita em uma estrutura de dados (todas implementadas como variáveis globais) é apresentado na figura 4.31.

Exemplos de acesso a elementos das estruturas de dados do algoritmo de retirada de superfícies escondidas	
tipo de acesso	**comando**
CABEÇA DA LISTA DE ARCOS DO i-ÉSIMO OBJETO DA CENA	boxes[i].cabeca
PONTO INICIAL DA ARESTA DE ESPAÇO OCUPADA PELO i-ÉSIMO OBJETO	boxes[i].p; boxes[i].p.x (coord. x do ponto boxes[i].p.y (coord. y do ponto boxes[i].p.z (coord. z do ponto
NÚMERO DE i-ÉSIMO QUADRILÁTERO DE PARTIDA DOS DO i-ÉSIMO OBJETO	boxes[i].cabeca^.vetquad[j]
CLASSIFICADORES DO QUADRILÁTERO ATIVO	boxes[i].cabeca^.vetquad[j].classif1 boxes[i].cabeca^.vetquad[j].classif2
PONTO MULTIPLO DE ÍNDICES i,j DO k-ÉSIMO OBJETO DA CENA	patch_26[k]^.[i,j].coord patch_26[k]^.[i,j].coord.x (x do ponto patch_26[k]^.[i,j].coord.y (y do ponto patch_26[k]^.[i,j].coord.z (z do ponto
PONTO PROJETADO DE ÍNDICES i,j DO k-ÉSIMO OBJETO DA CENA	patch_2d[k]^.[i,j].coord patch_2d[k]^.[i,j].coord.x (x do ponto patch_2d[k]^.[i,j].coord.y (y do ponto
VISIBILIDADE DO PONTO DE ÍNDICES i,j DO k-ÉSIMO OBJETO DA CENA	patch_2d[k]^.[i,j].visivel
"QUINTO PONTO" DO j-ÉSIMO QUADRILÁTERO DO k-ÉSIMO OBJETO DA CENA	qp[k]^[j].coord_p qp[k]^[j].coord_p.x (x do ponto qp[k]^[j].coord_p.y (y do ponto

Fig. 4.31 Acesso às estruturas de dados do método de Ohno

4.6 Uso de Técnica no Desenvolvimento de Software Gráfico

A Engenharia de Software tem defendido que a necessidade da aplicação de técnicas de desenvolvimento de software é urgente em todas as áreas em que o computador atua.

No desenvolvimento do sistema aqui descrito, empregou-se os métodos de Análise Estruturada [Ga 83], Projeto Estruturado [Pa 88] e Programação Estruturada, além de um cuidado com a interface com o usuário, recomendado como fundamental na produção do software de manipulação direta.

A Programação Estruturada é um conceito de desenvolvimento

de programas amplamente divulgado por muitos anos, e consiste da elaboração de programas utilizando basicamente as estruturas de sequência, seleção e iteração, que podem ser combinadas apenas por inclusão.

A Análise e o Projeto Estruturados acompanham as duas fases iniciais do desenvolvimento. Ambos aplicam a idéia de refinamento sucessivo (desenvolvimento "top-down" ou descendente), isto é, o problema é sucessivamente dividido em "partes" que podem ser tratadas separadamente. A análise é responsável por gerar o conjunto de funções básicas que resolvem o problema e o fluxo de informações entre elas, modelando o ambiente. O projeto é responsável por determinar uma estrutura e um detalhamento maior para essas funções, segundo alguns critérios básicos e visando mapear o problema numa solução implementável por computador, numa versão modular do sistema.

4.7 Considerações Finais

Basicamente o SMS possui três módulos básicos: o primeiro, de auxílio à introdução de malhas de controle, que funciona igualmente para qualquer dos métodos de modelagem utilizados; o segundo, de modelagem em si, que gera pontos sobre a superfície matemática e que implementa opcionalmente B-spline ou Bézier; o terceiro, que apresenta a superfície resultante com retirada de linhas escondidas, por dois diferentes métodos. Um deles, não pode ser utilizado para qualquer superfície possível de ser

modelada, sendo entretanto mais rápido para aquelas que pode apresentar. O outro, mais geral, permite um conjunto muito maior de casos de superfícies e mesmo cenas com vários objetos, sendo, porém, mais lento. Em todo o desenvolvimento houve uma preocupação em avaliar as opções quanto às possibilidades de melhora de eficiência de tempo de processamento, ocupação de memória e flexibilidade do sistema. Várias limitações aparecem, como consequência do objetivo geral do software e dos recursos computacionais, ou mesmo da própria característica do método matemático. O próximo capítulo detalha as especificações técnicas e de utilização do software, que totalizou, com todas as opções, por volta de 10000 linhas de código. Observa-se, entretanto que grande parte do esforço de desenvolvimento e do código produzido se concentra na manipulação dos dados e no tratamento de entrada e saída, como característica geral de um sistema de modelagem. A modelagem em si, ou geração dos pontos sobre a superfície, toma um esforço apenas parcial no contexto do sistema de modelagem.

CAPÍTULO V
RESULTADOS

5.1 Considerações Iniciais

O sistema desenvolvido permite modelar um conjunto de objetos tri-dimensionais, servindo por isso como uma ferramenta da análise do uso e da eficiência dos dois métodos empregados no contexto da modelagem geométrica.

Com o objetivo de ser aprimorado e adaptado a outro sistemas, dois dos três componentes do SMS podem ser integralmente aproveitados: o modelador em si, isto é, as rotinas que avaliam pontos sobre a superfície para os dois métodos empregados, e o programa de para determinação de linhas/superfícies escondidas, que permite visualizar a forma do objeto com maior nitidez.

Quanto à interface de entrada de dados, suas funções básicas cumprem os critérios mínimos de manipulação gráfica tri-dimensional; entretanto, sua generalidade compromete as necessidades de um software aplicativo. Com objetivos mais específicos, deverá ser adaptada, aproveitando-se, entretanto, a maior parte das rotinas que a integram, especialmente na produção dos arquivos que são utilizados pelos demais módulos. Os próximos parágrafos apresentam alguns resultados, em termos de potencialidades dos métodos e do sistema desenvolvido.

5.2 Características da modelagem

5.2.1 Projeto de formas pela malha de controle.

Alguns aspectos especiais envolvem a modelagem de sólidos utilizando malha de controle, que é a base de toda superfície spline.

Um desses aspectos é que o transporte de uma superfície de um sistema para outro se dá pelo transporte dos pontos de controle e re-avaliação no sistema destino. Em outras palavras, a base de dados que envolve um objeto projetado por superfícies matemáticas é bastante simples e pequena. Uma vez obtida a forma desejada, basta guardar os pontos da malha, que os algoritmos de avaliação gerarão pontos sobre a superfície, de acordo com o critério desejado de discretização. O espaço ocupado pelo armazenamento dos dados necessários para recuperar uma superfície não é problema num sistema deste tipo.

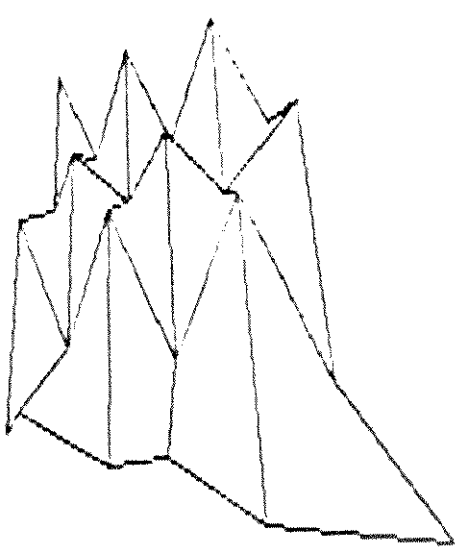
A modelagem por malha de controle (base do modelo implementado) exige que o o projetista do tenha uma noção intuitiva da forma final da superficie segundo o método desejado (Bézier ou B-spline) através da posição espacial dos pontos da malha. Tal noção nem sempre é trivial, como é o caso da figura 5.4. Dessa forma, o usuário do sistema necessita desenvolver a habilidade de reconhecer o comportamento da modelagem para efetivamente aproveitar as características deste tipo de modelo.

Entretanto, um aspecto importante da modelagem através da malha de controle é a grande flexibilidade que ela fornece ao processo de elaboração da forma. Com ela, pode-se promover

alterações do objeto com modificação de apenas alguns pontos, sem necessidade de redefinir todos os dados do sólido.

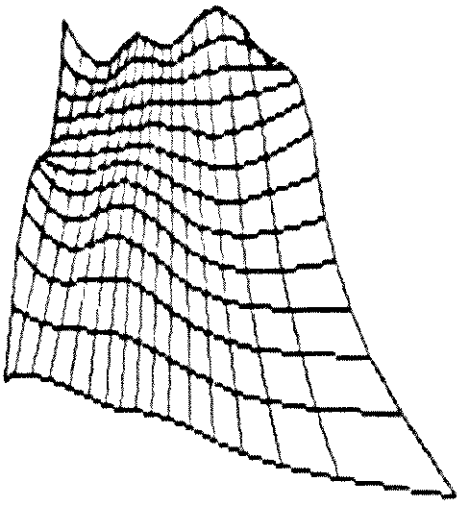
5.2.2 Comportamento da modelagem

As figuras 5.1 a 5.4 fornecem modelos de objetos construídos através do SMS. Devido às características dos métodos utilizados, o domínio dos objetos possíveis de serem representados é bastante amplo. Superfícies fechadas podem ser facilmente modeladas, igualando-se os dois extremos das malhas de controle na direção desejada.

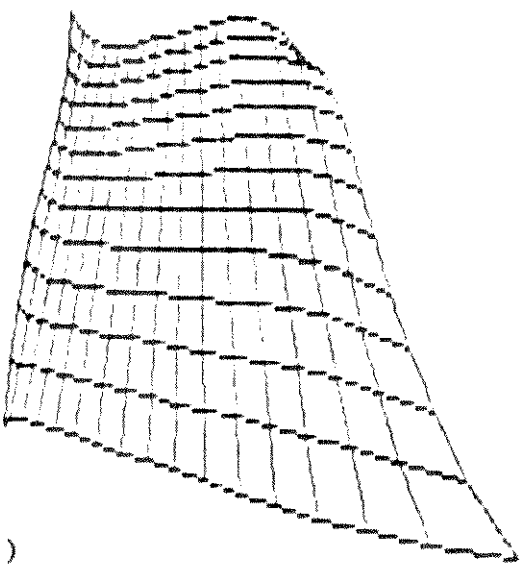


a)

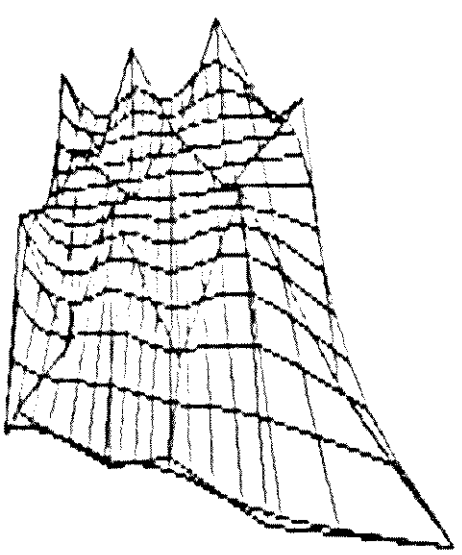
Fig. 5.1 Cena 1
a malha de controle



b)

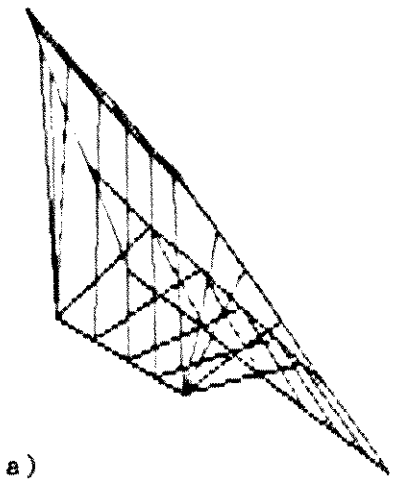


c)

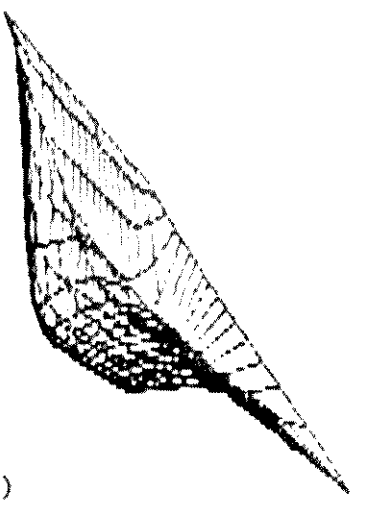


d)

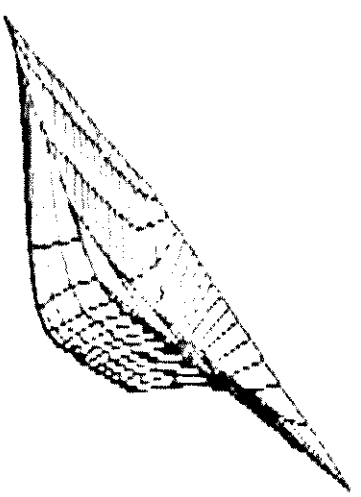
Fig. 5.1 (continuação)
b superfície B-spline correspondente a 5.1a
c superfície Bézier correspondente a 5.1a
d malha e superfície



a)

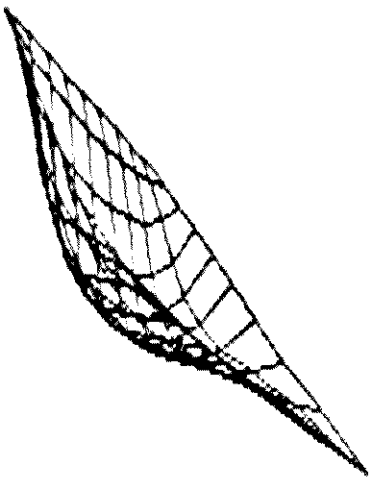


b)

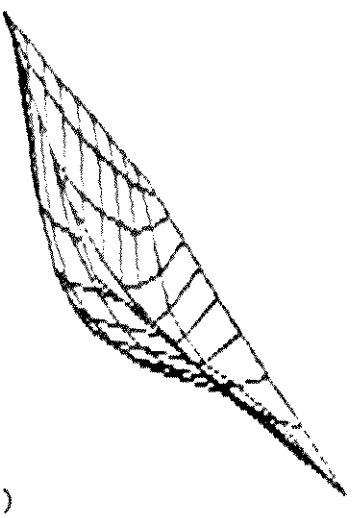


c)

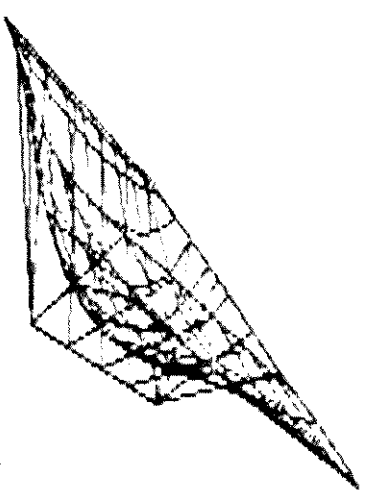
Fig. 5.2 Cena 2
a malha de controle
b superfície B-spline correspondente a 5.2a
c superfície de 5.2b sem linhas escondidas



d)

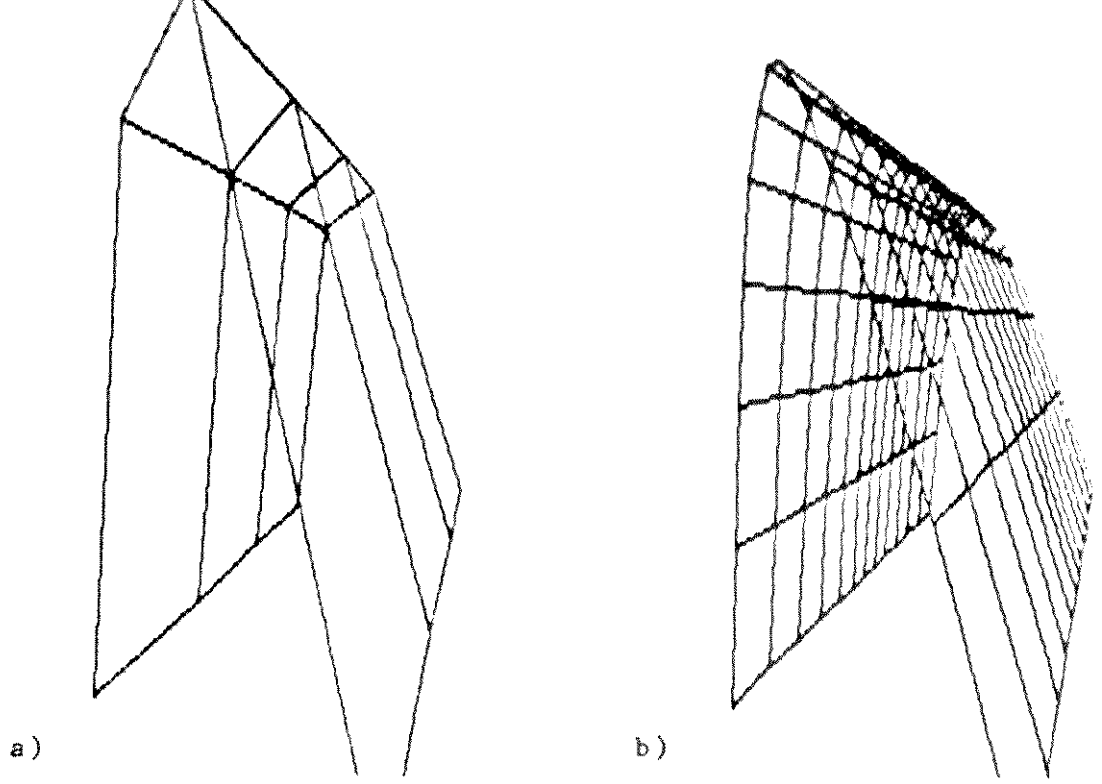


e)



f)

Fig. 5.2 (continuação)
d superfície Bézier correspondente a 5.2a
e superfície de 5.2d sem linhas escondidas
f malha e superfície



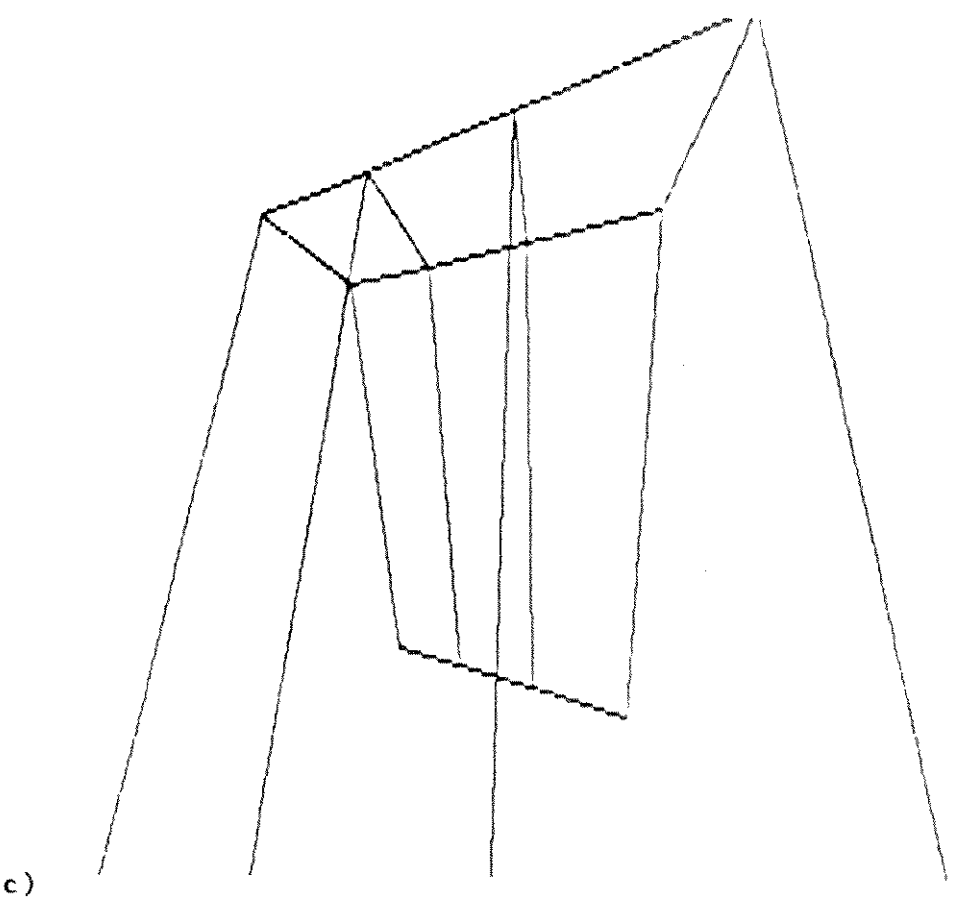
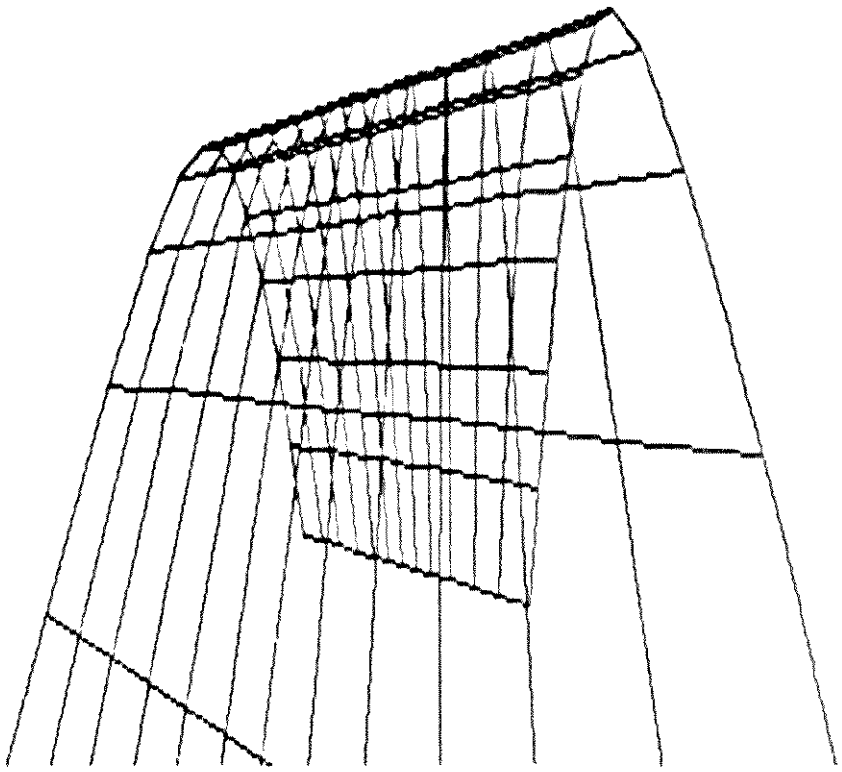
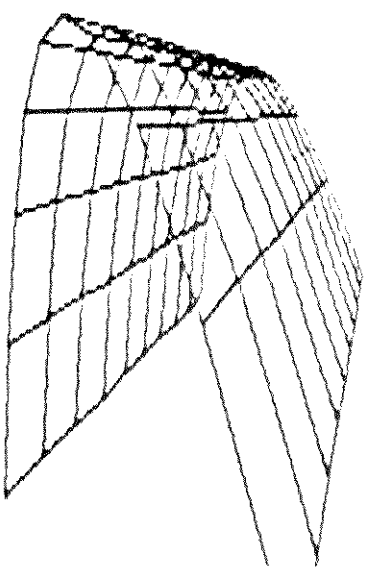


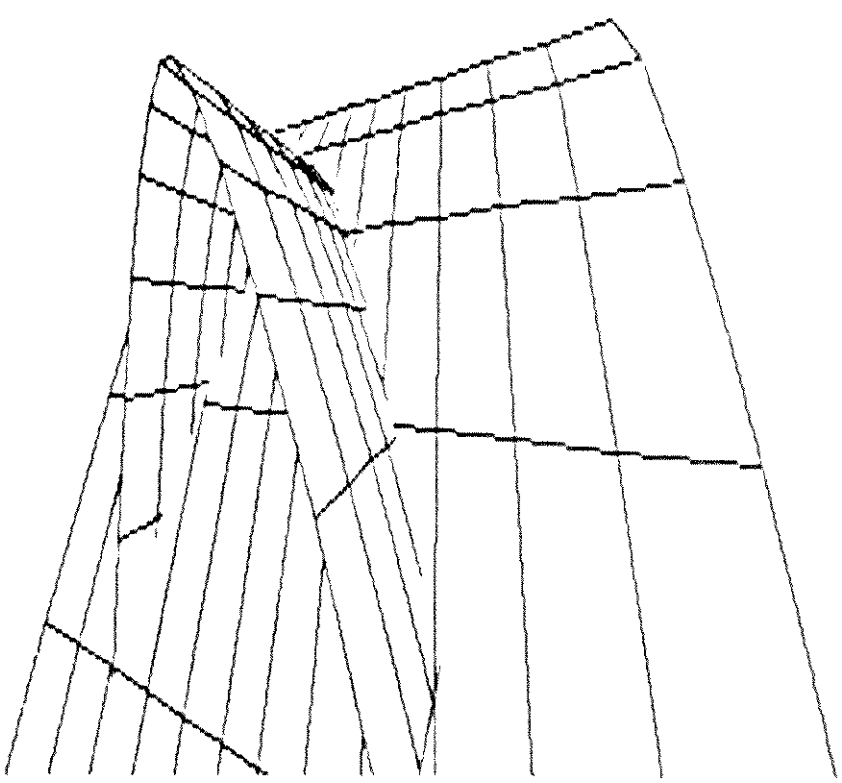
Fig. 5.3 Cena 3
a malha de controle da primeira superfície
b superfície B-spline correspondente a 5.3a
c malha de controle da segunda superfície



d)

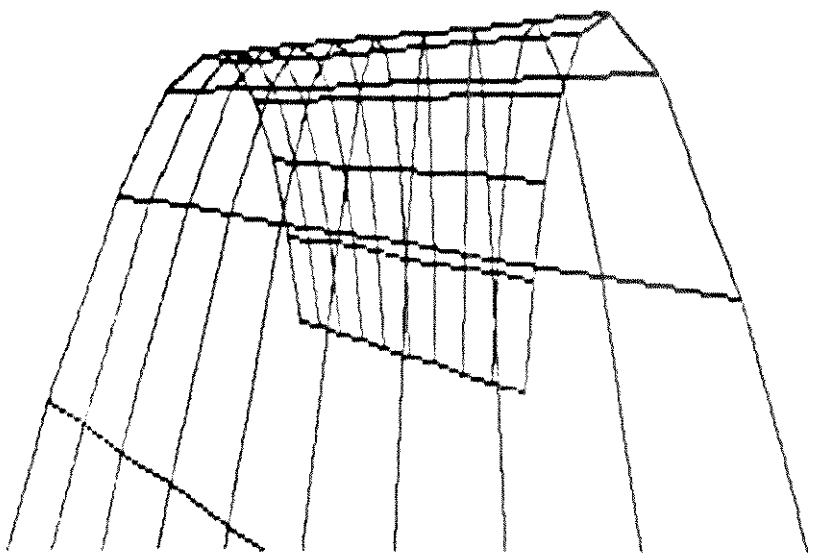


f)

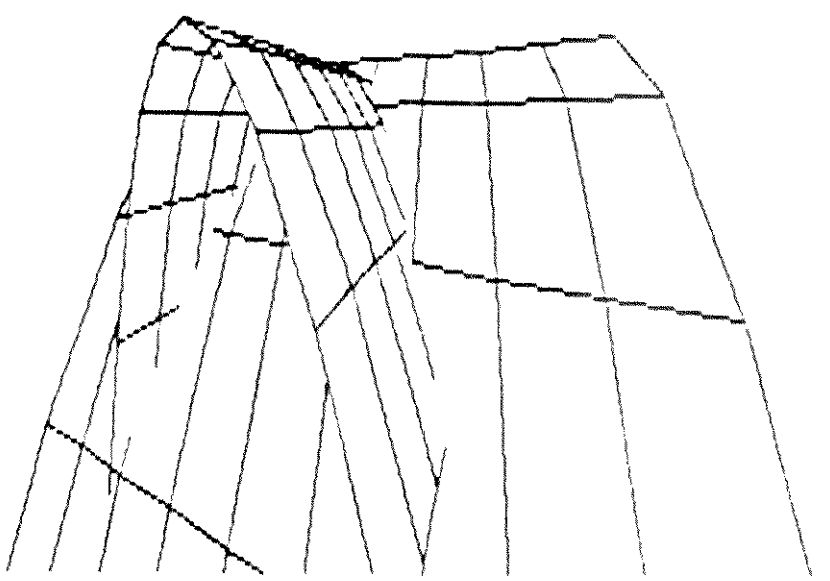


e)

Fig. 5.3 (continuação)
d superfície B-spline correspondente a 5.3c
e cena B-spline incluindo as duas superfícies
f superfície de Bézier correspondente a 5.3a

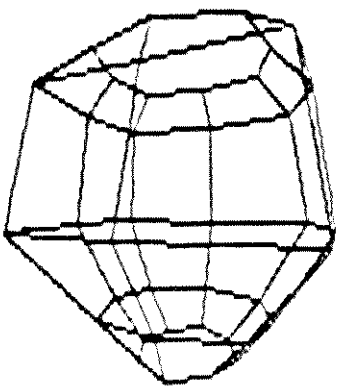


g)

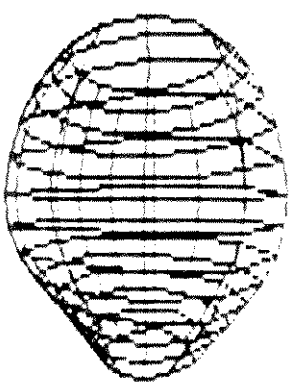


h)

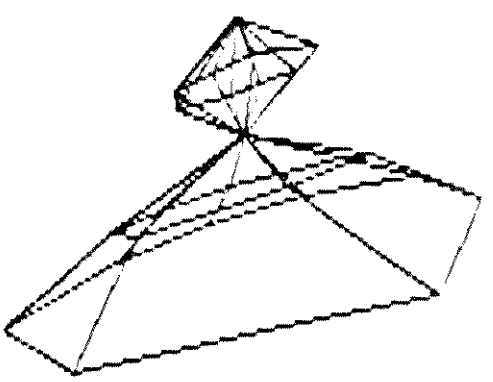
Fig. 5.3 (continuação)
g superfície B-spline correspondente a 5.3c
h cena de Bézier incluindo as duas superfícies



a)

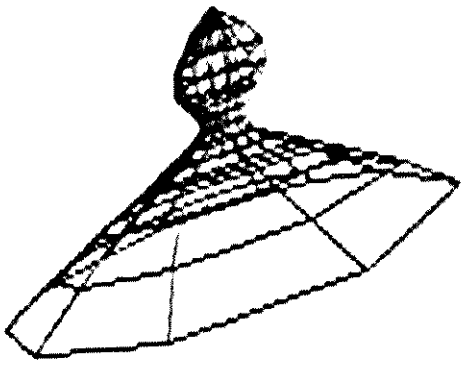


b)

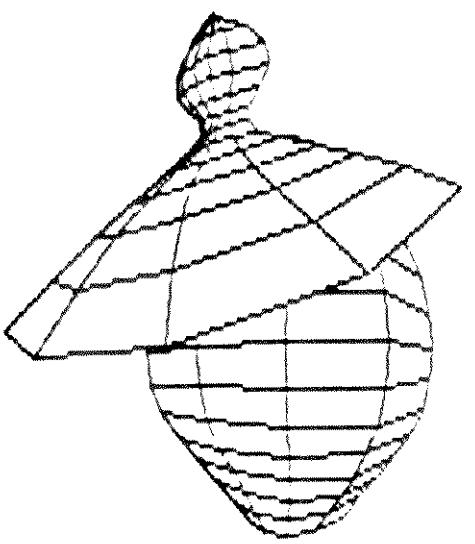


c)

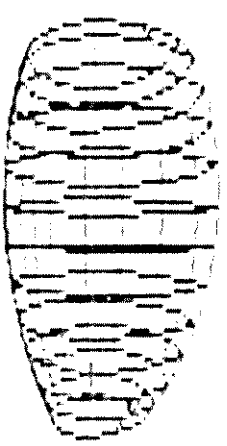
Fig. 5.4 Cena 4
a malha de controle da primeira superfície
b superfície B-spline correspondente a 5.4a
c malha de controle da segunda superfície



d)

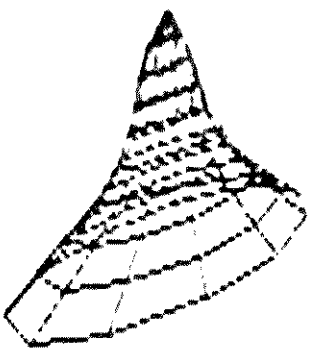


e)

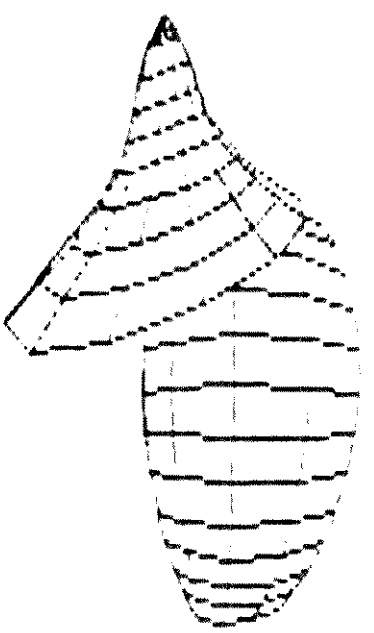


f)

Fig. 5.4 (continuação)
d superfície B-spline correspondente a 5.4c
e cena B-spline incluindo as duas superfícies
f superfície de Bézier correspondente a 5.4a



g)



h)

Fig. 5.4 (continuação)
g superfície de Bézier correspondente a 5.4c
h cena de Bézier incluindo as duas superfícies

A modelagem por B-spline dá a possibilidade de definição do grau da superfície nas duas direções de variação dos parâmetros utilizados, e a modificação de pontos da malha do objeto é localmente controlada, fazendo com que as alterações possam ser feitas com maior domínio dos efeitos pelo usuário. É sem dúvida um método mais flexível de projeto de formas, por suas características matemáticas específicas.

Possui entretanto um problema de fronteira. A borda do trecho de superfície projetado não é definida pela equação matemática que formula o método. O tratamento adotado para especificar a fronteira da superfície resolve plenamente o problema de continuidade de função entre duas superfícies adjacentes, mas não resolve continuidade de mais alta ordem, isto é, a junção não é "suave".

Já Bézier possui o grau da superfície dependente do número de pontos da malha, o que gera dois problemas a serem contornados:

1. o grau da superfície alto gera dificuldade de manipulação e tempo de processamento maior e

2. um erro de programação pode ser inserido, uma vez que as funções de "blending" de Bézier necessitam do cálculo de fatorial, que em processamento de valores inteiros possui limitação de precisão bastante grande (overflow para valores relativamente baixos). Além disso, o método possui controle global da forma, o que não permite grande domínio, pelo usuário, de qual vai ser a mudança da forma com a alteração de posição de determinado ponto.

O primeiro problema citado costuma ser contornado pela manipulação de uma superfície de Bézier por "patches", isto é, pela conexão de diversos pedaços de superfície para modelar a superfície toda, cada um deles bicúbico. A mesma medida também permite fornecer certa "localidade" de manipulação da forma, já que reduz a região a ser manipulada a cada vez, e que corresponde ao "patch". Como contrapartida, este método gera o problema de fronteira entre um e outro pedaço da superfície. Tal tratamento não foi adotado neste trabalho, uma vez que a modelagem B-spline, tendo sido o objetivo principal, cobre os objetivos de modelagem previstos para o sistema.

O segundo problema citado foi contornado pela alteração da rotina original de geração das funções de "blending", trocando por valores reais, mais precisos portanto, os resultados da função de "blending". Essa providência, entretanto, pode gerar um erro de precisão, advinda da manipulação de variáveis reais para resultados tipicamente inteiros. Além disso, com relação ao tempo de processamento, observa-se que aritmética real normalmente é mais lenta, diminuindo a performance do sistema.

Mesmo considerando a abordagem acima, observa-se que a modelagem de objetos por Bézier é mais rápida computacionalmente do que a B-spline. A tabela abaixo mostra os tempos de processamento da avaliação das superfícies apresentadas nas figuras 5.1 a 5.4, para um computador baseado no microprocessador 80386.

TABELA DOS TEMPOS DE PROCESSAMENTO DA AVALIAÇÃO DE SUPERFÍCIES

figura	fig 5.1	fig 5.2	fig 5.3		fig 5.4	
método	4x3	3x3	3x3	3x3	4x4	3x4
B-SPLINE	14 s	10 s	7 s	7 s	40 s	22 s
BEZIER	10 s	15 s	6 s	6 s	25 s	21 s

Na tabela, as dimensões anotadas na parte inferior da especificação da figura representam, para B-spline, a ordem da superfície, e para Bézier, o controle da precisão.

5.3 Interface de Entrada de Dados

A interface de entrada de dados tem por objetivo permitir uma maneira amigável de projetar malhas de controle, com auxílio de procedimentos gráficos, já que a introdução de dados numéricos para modelagem matemática é quase impossível para um conjunto de dados grande, o que se obtem de uma forma apenas relativamente complexa. Tal interface deve, portanto, gerar dados de malha de controle, obtendo esses dados por um editor gráfico, com funções usuais de manutenção da base de dados.

O apêndice 2 resume o manual de utilização da interface com o usuário, que mas ela permite a edição em doi modos: por coordenadas ou por cursor. O modo de edição por coordenadas

permite ao usuário entrar valores das coordenadas x, y e z dos pontos. O modo de edição por cursor permite ao usuário introduzir dados por dois campos de trabalho: um que identifica o plano xy e outro que itentifica as posições do plano xz. O cursor caminha nesses dois planos a passos fornecidos pelo usuário. Os planos são apresentados em tela, e o usuário caminha com o cursor por teclado. Além dessas funções, o programa de interface mostra consecutivamente a imagem da malha, em perspectiva, as funções disponíveis e dados da malha editada. A tela de trabalho possui várias divisões: uma para mostra do menu de funções disponíveis, uma para os dados, duas para os campos de edição (planos xy e xz), uma para mostra da imagem projetada da malha, e uma para resultados (figura 5.5). O conjunto de funções disponíveis é dado por:

- Funções de edição, que compreende:

definição de intervalos entre coordenadas

escolha de manipulação por cursor/coordenada

introdução de pontos na malha de controle

alteração da malha de controle

informações numéricas adicionais sobre o objeto

- Funções de visualização da malha, que compreende:

projeção simultânea da malha de controle, enquanto é projetada

alteração da visualização da malha (escala, rotação, translação)

- Funções de gerenciamento de dados, que compreende:

gravação da malha obtida

gravação da matriz de visualização resultante

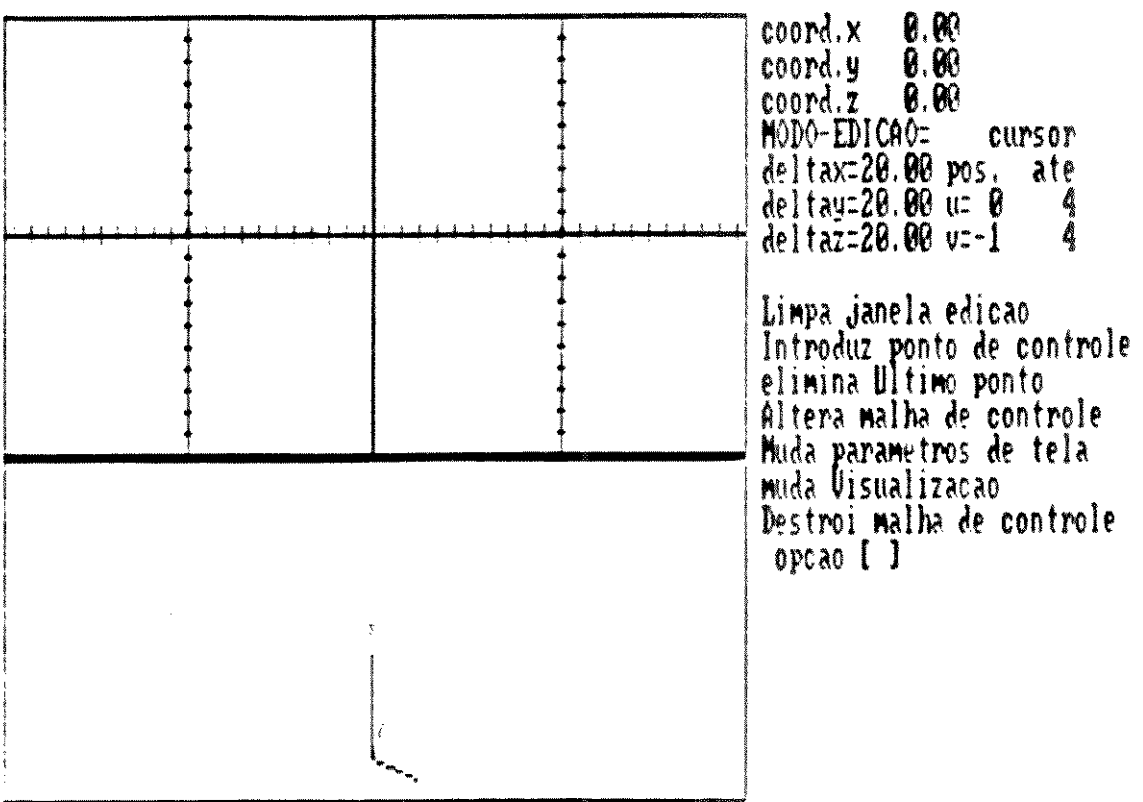


Fig. 5.5 Tela Inicial da Interface de entrada de dados

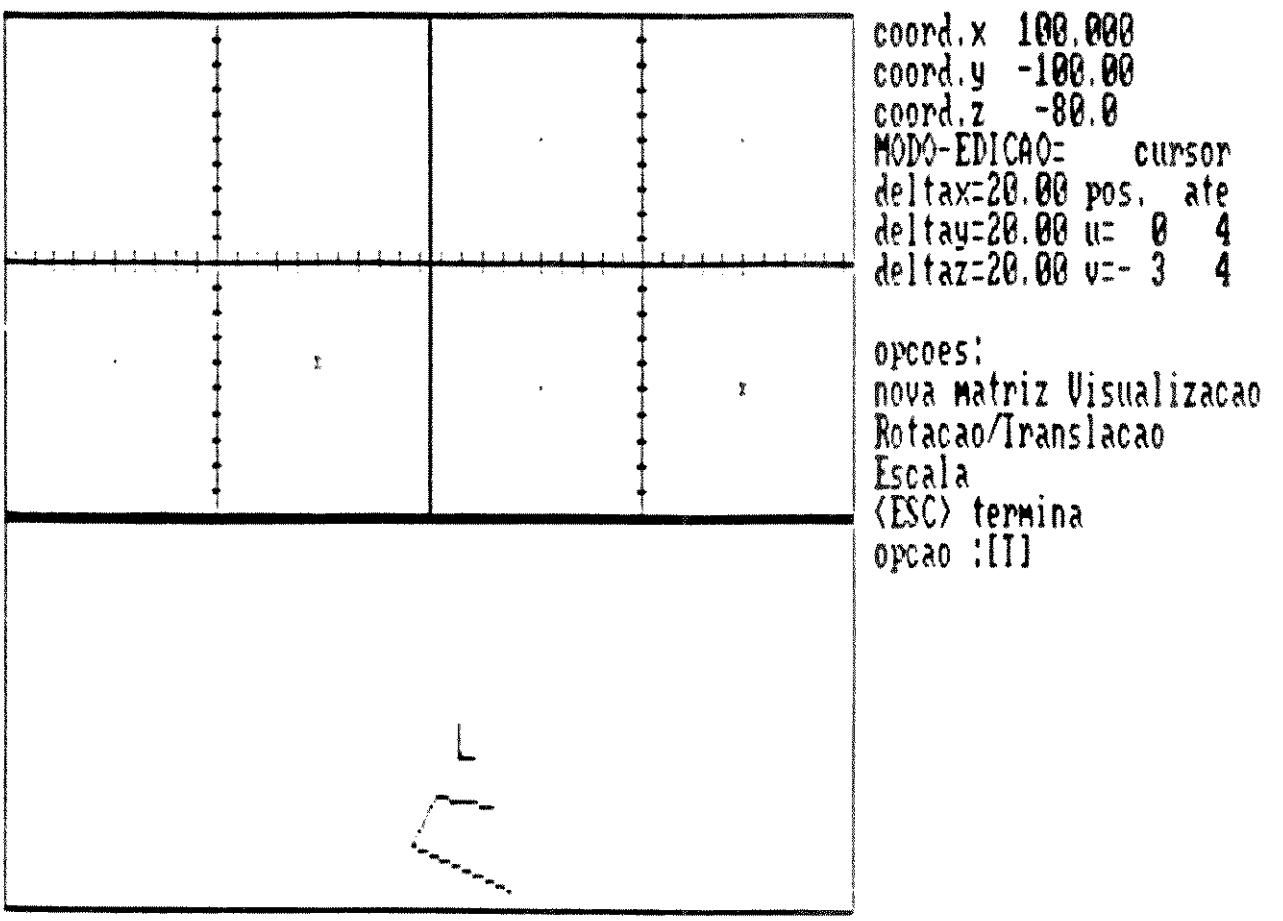
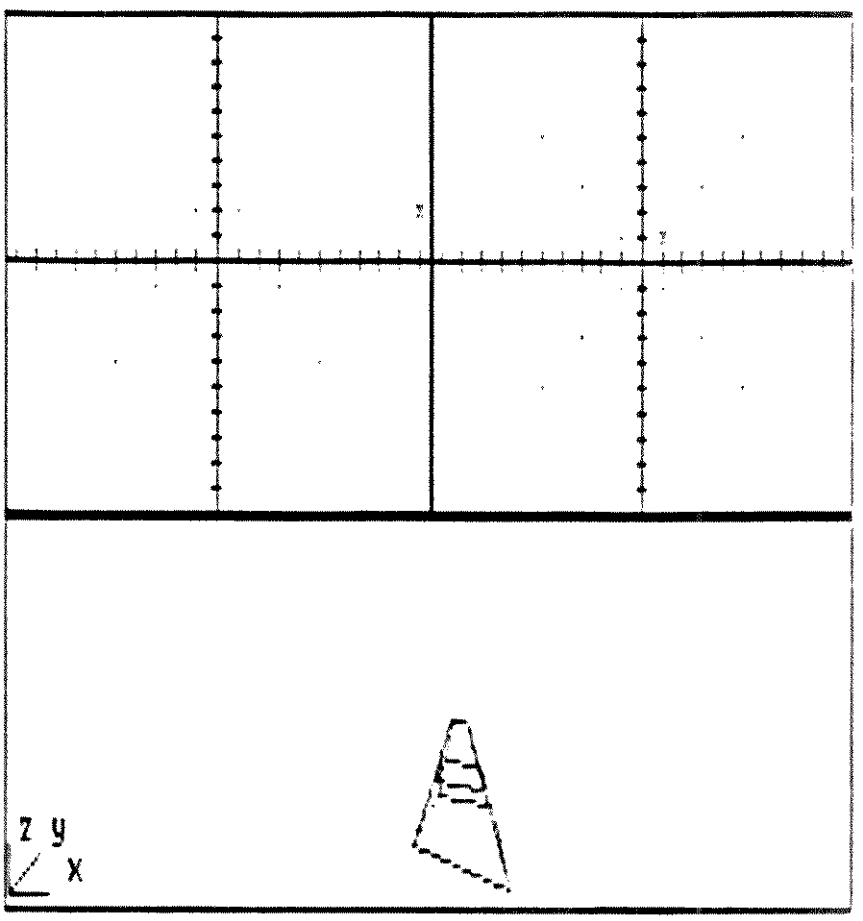


Fig. 5.6 Fases do projeto de uma malha pelo programa de Interface
a primeira fase: primeiros três pontos introduzidos

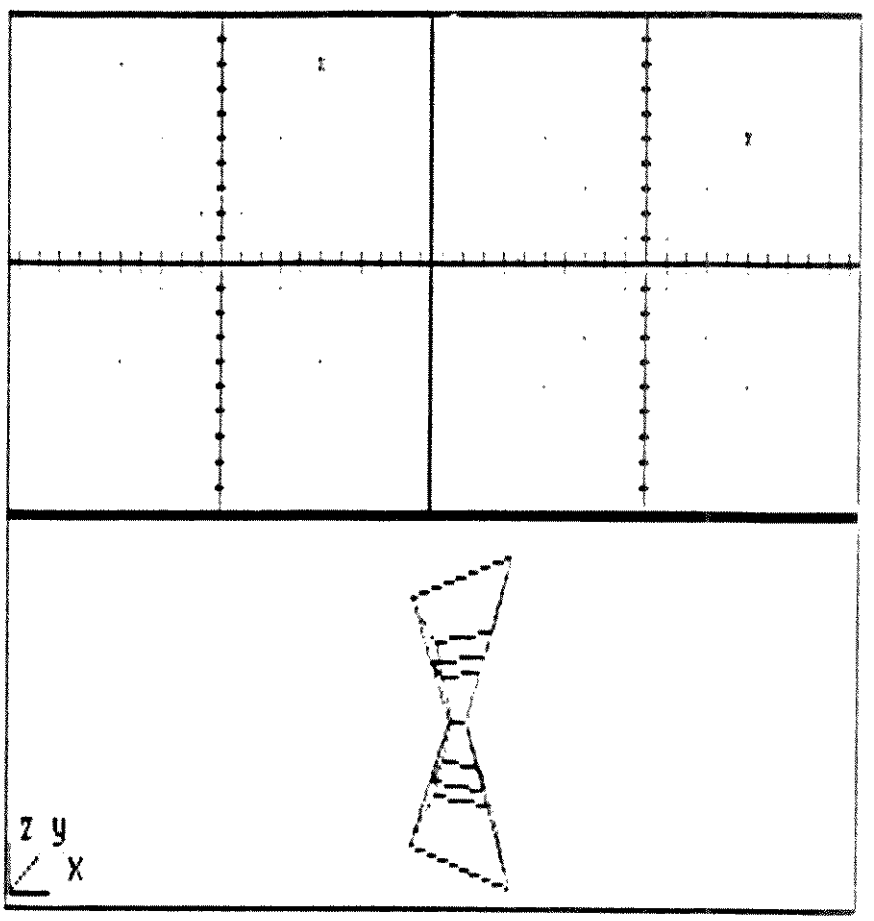


coord.x 20.000
coord.y 20.000
coord.z 40.0
MODO-EDICAO= cursor
deltax=20.00 pos. ate
deltay=20.00 u= 3 4
deltaz=20.00 v= 0 4

Limpa janela edicao
Introduz ponto de controle
elimina Ultimo ponto
Altera malha de controle
Muda parametros de tela
muda Visualizacao
Destroi malha de controle
opcao [1]

tecle <ESC> para finalizar leitura

Fig. 5.6 (continuação)
b segunda fase: metade da malha introduzida



coord.x 100.000
coord.y 100.000
coord.z 160.0
MODO-EDICAO= cursor
deltax=20.00 pos. ate
deltay=20.00 u= 4 4
deltaz=20.00 v= 4 4

Limpa janela edicao
Introduz ponto de controle
elimina Ultimo ponto
Altera malha de controle
Muda parametros de tela
muda Visualizacao
Destroi malha de controle
opcao [ ]

tecle <ESC> para finalizar leitura

Fig. 5.6 (continuação)
c terceira fase: malha completa

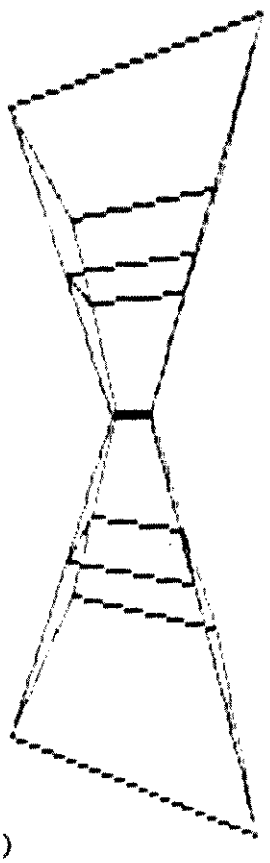
```
MALHA DESENHO

intervalo de variacao de: x =    20.00 y =    20.00 z =    20.00

distancia em pontos tela: x = 10 y = 5 z = 5

posicao do ultimo ponto introduzido na malha : dir u = 4 dir v = 4

MATRIZ DE TRANSFORMACAO - ULTIMA SITUACAO

   0.940     0.342    -0.000    88.925
   0.000     0.000     1.000   -50.000
  -0.342     0.940    -0.000   344.320
   0.000     0.000     0.000     1.000



deseja gravar resultados ?
```

d)

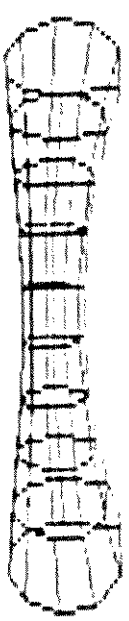
```
                Escolha uma opcao

          continua Editando

          Gera superficie

          (esc) termina

          opcao : [  ]
```

e)

Fig. 5.6 (continuação)
d dados auxiliares apresentados pelo programa
e menu final de opções

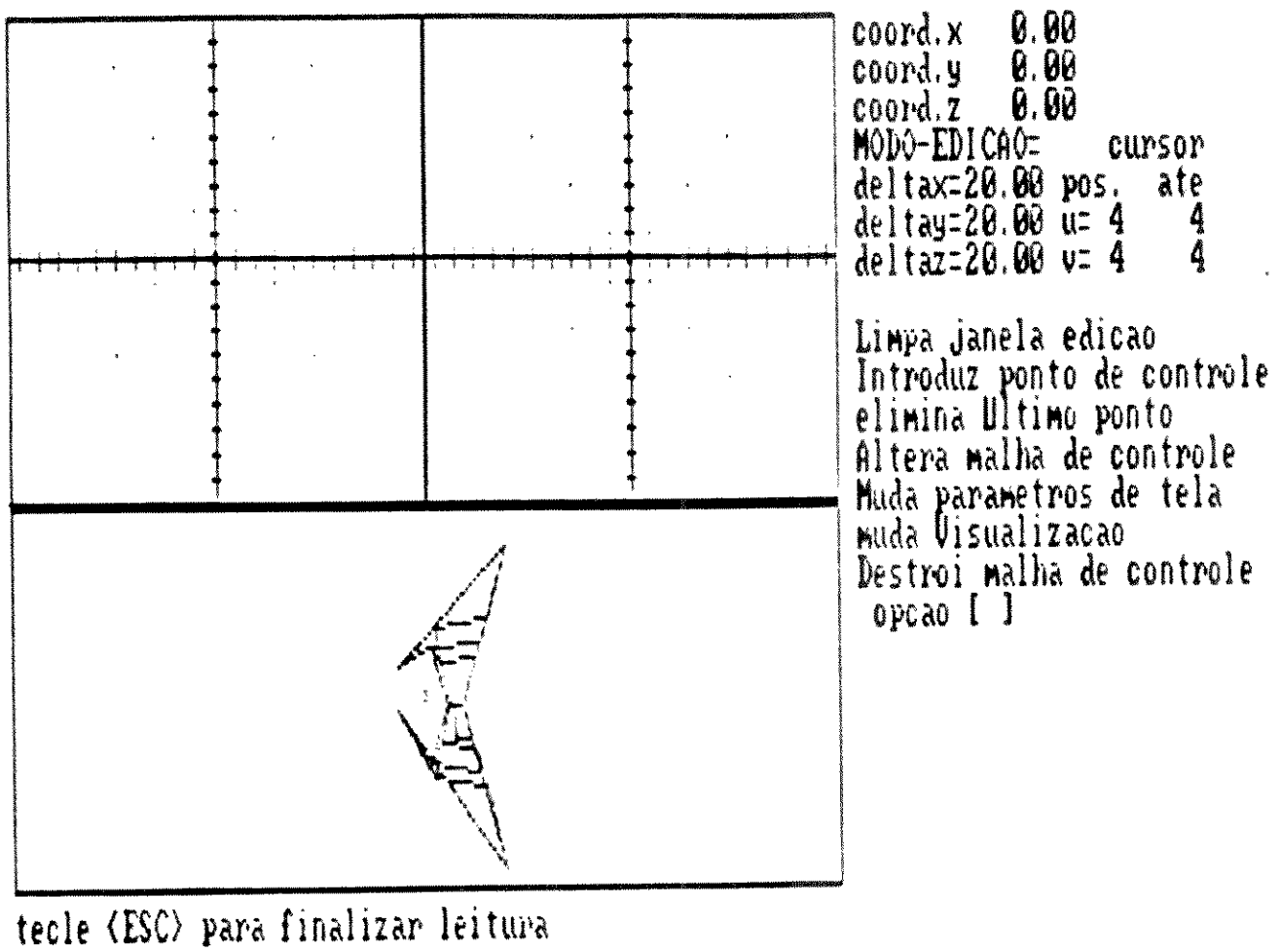


f)



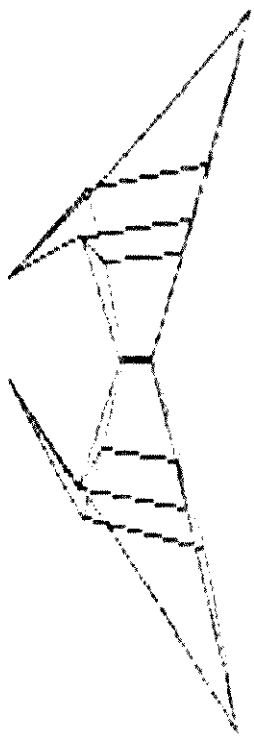
g)

Fig. 5.6 (continuação)
f malha de controle, ampliada
g superfície avaliada

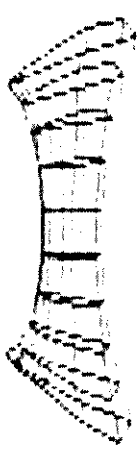


a)

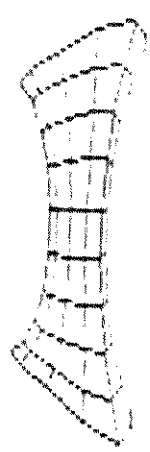
Fig 5.7 Alteração da Malha de controle pelo programa de Interface
a malha de controle alterada



b)

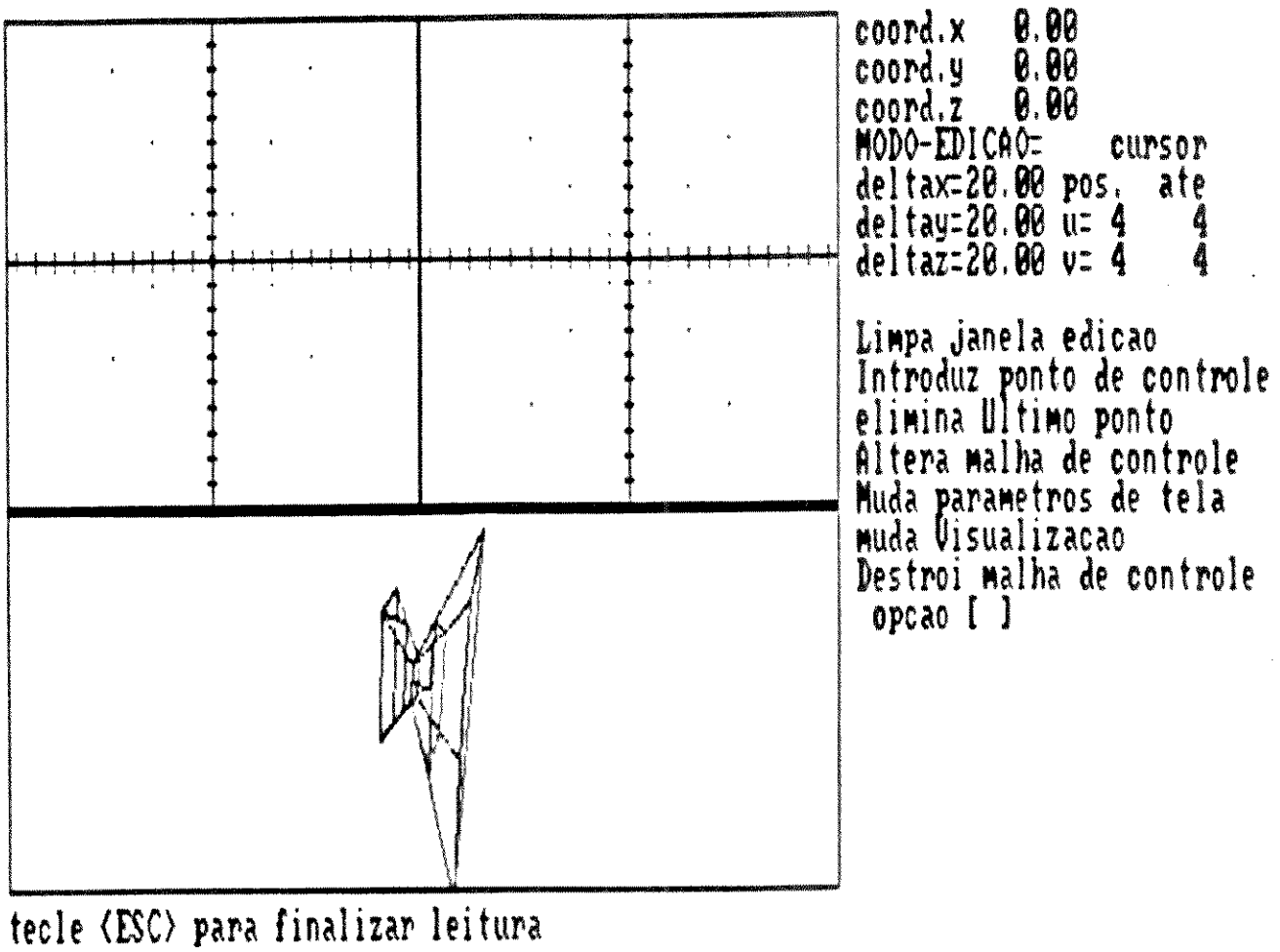


c)



d)

Fig. 5.7 (continuação)
b malha de controle, ampliada
c superfície avaliada
d superfície com retirada de linhas escondidas



a)

Fig 5.8 Alteração da Visualização da malha original.
a malha com visualização mudada

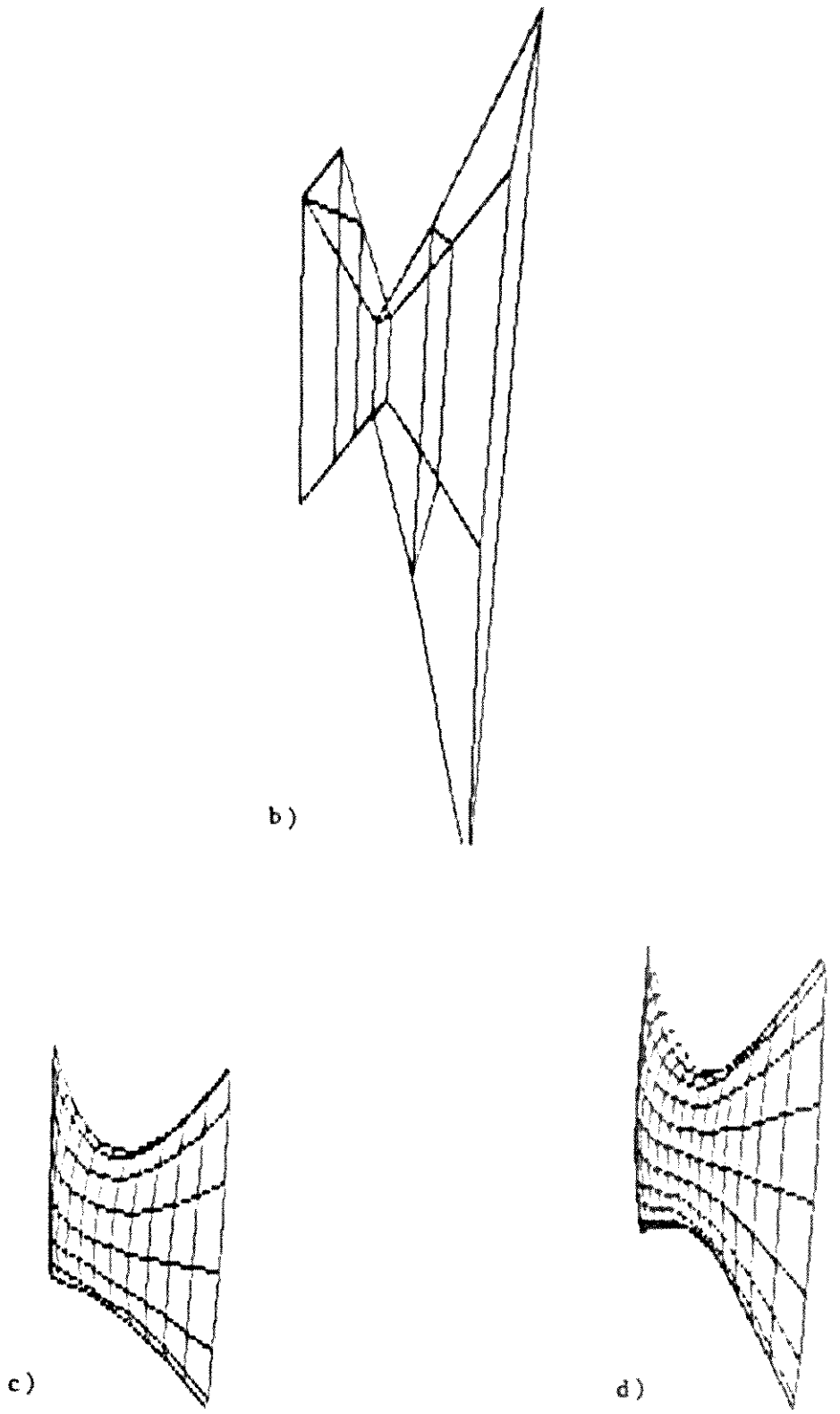


Fig. 5.8 (continuação)
b malha de controle, ampliada
c superfície avaliada
d superfície com retirada de linhas escondidas

5.4 Visualização da Superfície

O processo mais lento do programa diz respeito à visualização para retirada de superfícies escondidas. O método admite cenas complexas, limitadas apenas pela quantidade de memória da área de Heap do computador.

A tabela abaixo mostra os tempos de processamento para a retirada de linhas escondidas, para as cenas das figuras 5.1 a 5.4, para um computador baseado no microprocessador 80386, com e sem co-processador aritmético 80387.

TABELAS DE TEMPO DE PROCESSAMENTO DA GERAÇÃO DAS CENAS

FIG	SEM CO-PROCESSADOR	COM CO-PROCESSADOR
5.1	5 min	4 min
5.2	4 min	3.5 min
5.3	8 min	6 min
5.4	15 min	11 min

Observa-se que o tempo de processamento é relativamente alto, se se considerar o desejo (e, muitas vezes a necessidade) do usuário de gerar a imagem sem linhas escondidas várias vezes durante o mesmo projeto. O método de retirada de linhas escondidas por coordenadas de tela, pode gerar, por exemplo, a figura 5.1 com o tempo de 15 segundos, e a figura maior da cena 5.3 em 5 segundos.

Entretanto, o método de visualização de superfícies por

coordenadas de tela possui um domínio muito restrito de objetos possíveis de serem visualizados, o que inviabiliza sua aplicação para superfícies mais complexas. Ainda assim, é uma opção para as superfícies mais simples, dentro da coerência de imagem por ele exigida. Além disso, este método não permite a introdução de mais de um objeto na mesma cena.

Já o método de Ohno de visualização possui um domínio muito vasto de objetos representáveis, e uma precisão de imagem bastante boa. Permite também o tratamento de cenas complexas, limitadas em número de objetos apenas pela quantidade de memória disponível na máquina. Sua lentidão é bastante amenizada pelos recursos de otimização empregados na implementação do algoritmo.

5.5 Uso de técnica de desenvolvimento

Como resultado do desenvolvimento do sistema a partir de técnicas estruturadas obteve-se uma maior facilidade de implementação e manutenção do sistema, bem como uma documentação paralela rica como ferramenta de entendimento do sistema.

Um aspecto prático dessas vantagens foi a possibilidade de inclusão da modelagem por Bézier no sistema já pronto, visto que o sistema foi originalmente desenvolvido para aplicação de B-splines. A generalidade dos módulos de Introdução de Dados e de Apresentação da Cena permitiu a adaptação imediada do novo algoritmo de modelagem.

5.6 Características Finais do Sistema

O sistema como um todo gerou aproximadamente 10000 linhas de código, entre modelagem, interface de entrada e interface de saída.

Foi desenvolvido para aplicação em microcomputador to tipo IBM PC, tendo sido testado num computador baseado no microprocessador 80386, com e sem processador aritmético.

As opções residem em se escolher tanto a modelagem (Bézier ou B-spline), como o método de visualização (tanto por Ohno como por coordenadas de tela), através de conjunto de rotinas que são incluidas no programa, escrito em Pascal Turbo, versão 3.0.

A interface de entrada de dados tem um núcleo básico, de nome físico "sup_int.pas", e inclui várias bibliotecas. Delas, algumas são de propósito geral, e possuem nomes físicos "matriz1.pro" (manipulação de matrizes), "trans.pro" (manipulação de transformações de vizualização), e "graph.p" (rotinas gráficas do próprio Pascal). Outras rotinas são pacotes de módulos do próprio programa de interface: "interscon.pro" (rotinas de diálogo), "le.pro" (rotinas de leitura de dados) e "inigrav.pro" (rotinas de inicialização e finalização do sistema).

O programa de visualização tem um núcleo básico, de nome físico "cena.pas", e inclui as bibliotecas: "graph.p", "matriz1.pro" e "deftipo.pro" (definições das estruturas de dados do programa de visualização)

Durante todo o desenvolvimento, o sistema contou com uma rotina auxilias, alocada na biblioteca de nome físico "diag.pro", que gera um diagnóstico do estado de todas as estruturas de dados do sistema.

O código objeto da interface de entrada de dados ("sup_int.com"), e o código objeto da visualização ("supescon.com" e "cena.com") necessitam das bibliotecas da malha de controle (arquivos de extensão ".pon") e de matrizes de vizualização (arquivos de extensão ".mt"), sendo que o arquivo de nome "default.mt" deve estar presente em disco, no mesmo diretório do arquivo objeto.

CAPÍTULO VI
CONCLUSÕES E SUGESTÕES PARA NOVAS PESQUISAS

6.1 Conclusões sobre o trabalho

A modelagem matemática de superfícies representa uma opção bastante útil no projeto de objetos, e é atualmente suportada por vários sistema CAD, de objetivos diversos, em especial projeto mecânico. O desenvolvimento de software para CAD, e para computação gráfica de uma maneira geral, envolve trabalho em equipe, sendo de custo muito alto em termos de tempo e de pessoal de desenvolvimento. O custo do software supera em muito o custo do hardware. A justificativa para esta situação reside no porte e complexidade dos sistemas gráficos, além da preocupação com a interface homem-máquina, que por si só justifica boa parte do esforço de desenvolvimento dos sistemas gráficos.

Além dos aspectos já mencionados, uma necessidade é a otimização dos algoritmos gráficos, que normalmente são bastante custosos computacionalmente, tanto em termos de ocupação de memória como tempo de processamento, em especial para aplicações em microcomputadores.

O sistema desenvolvido, por ser de uso geral, tem três objetivos básicos:

1. possibilitar a implementação em microcomputador de sistemas de modelagem, que via de regra são excessivamente grandes e lentos, tendendo sempre a ser implementados em computador de médio/grande porte.

2. permitir uma programação clara, bem documentada e generalizada suficiente para permitir extensão e adaptação a outros sistemas de maior porte, que utilizem B-splines (ou Bézier) como parte da modelagem disponível.

3. elaborar um sistema de uso geral, responsável por cuidar dos aspectos de implementação de uma técnica de modelagem matemática superficial. Tais aspectos incluem interface com o usuário, apresentação com retirada de linhas escondidas, utilização de memória e estruturas de dados adequadas ao sistema e sua otimização.

Adicionalmente, o trabalho pretendeu avaliar a aplicação de técnicas de desenvolvimento de software para sistemas deste tipo e porte.

Nesse contexto, o trabalho atendeu aos seus objetivos. A interface utilizada se mostrou flexível, possibilitando a definição de formas de maneira intuitiva e de manipulação razoável pelo projetista. Neste aspecto, evidentemente o desenvolvimento se deparou com soluções de compromisso, pela impossibilidade de cuidar a um mesmo tempo de todas as características de um sistema de modelagem. Assim, a interface se preocupou em promover as principais funções de manipulação de uma malha de controle, sem se dedicar a todos os fatores humanos do desenvolvimento de uma interface gráfica.

Quanto à velocidade de processamento, como os demais algoritmos gráficos, a geração da imagem do objeto com alguma preocupação de melhoria de imagem é bastante lenta, pelo método de Ohno. Tal velocidade de processamento depende da precisão com que se deseja gerar a "rede" de pontos e da complexidade da cena.

Entretanto não se consegue otimizar os algoritmos muito mais do que foi feito. O método de visualização por coordenadas de tela é de execução bem mais rápida. No entanto, não se pode gerar qualquer tipo de objeto e nem montar cenas com tal algoritmo. Assim, ganha-se velocidade e perde-se generalidade.

Parte da desvantagem da lentidão do processo de visualização pode ser contornada, em termos de manipulação, tornando esporádica a execução do algoritmo de retirada de linhas escondidas, e gerando a superfície sem este tratamento durante a maior parte do processo de projeto interativo. O programa de visualização pode ser acionado em circunstâncias em que o projetista não consegue identificar corretamente a forma em função do excesso de linhas da imagem.

A modelagem em si se apresentou, tanto por um método como por outro, bastante flexível, permitindo a elaboração de objetos de formas variadas. Dos dois métodos implementados, B-spline permitiu maiores opções de projeto, como era esperado pelas próprias características matemáticas do método. Bézier, entretanto, é de avaliação mais rápida, e de melhor controle do problema de fronteira.

Em termos de equipamento, o uso de sistemas de maior porte certamente apoiaria melhor aspectos como ocupação de memória e tempo de processamento. Em microcomputador, há a necessidade de se elaborar ajustes tanto em um quanto em outro aspecto para poder adaptar os algoritmos às condições do equipamento. Como ambos os aspectos (tempo e memória) são via de regra conflitantes, o uso de microcomputador implica numa maior

concentração de esforços em desenvolver atividades de implementação. Ainda assim, é possível manipular o programa de modelagem em sistemas físicos desse porte. A quantidade de memória disponível na máquina limita a complexidade da cena. Quanto ao aspecto de precisão no cálculo, a busca de exatidão é muitas vezes desejável. Entretanto, a necessidade de precisao matemática é um aspecto bastante dependente do objetivo do sistema, isto é, em sistemas que deverão estar ligados a manufatura de objetos projetados, certamente a precisão é importante e deve ser de domínio do projetista; já em sistema cujo objetivo é a imagem do objeto, a precisão não pode (e não deve) ser ampliada a valores que aumentem o esforço computacional, com o objetivo de alcançar uma precisão da imagem que o equipamento não comporte, ou seja, a precisão e um fator que em computação depende decisivamente do equipamento objetivo do sistema. Como o sistema desenvolvido não objetiva nenhuma aplicação específica, nem é aplicado a um hardware definitivo, a precisão foi apenas superficialmente tratada.

Outro aspecto relacionado ao equipamento reside no fato de que os periféricos gráficos são fundamentais para sistemas gráficos de modelagem. O sistema seria melhor configurado com uso de dispositivos como vídeo de maior resolução, traçadores e mesa digitalizadora. Tais equipamentos são adaptáveis ao sistema com pequenas alterações. A modularidade do sistema concentrou os aspectos de saída em módulos bem localizados.

Quanto à utilidade da aplicação das técnicas estruturadas de análise, projeto e programação, o resultado é o esperado e comentado pela literatura, isto é, a fase de implementação

propriamente dita fica bastante aliviada, os testes são facilitados, e há uma clara noção do funcionamento do sistema mesmo antes de ser aplicado no computador. Esse resultado recomenda a aplicação de técnicas de Engenharia de Software em Sistemas gráficos, uma vez que suas características (porte, uso, necessidade de confiabilidade, tratamento de aspectos gerais e específicos) representam um conjunto de bons motivos para apoio em Metodologias de Desenvolvimento buscando bons sistemas como resultado.

A linguagem utilizada possui limitações de natureza gráfica. Por exemplo, a versão 3.0 do Turbo Pascal não admite o uso de placas gráficas de maior resolução nem o uso de cores na resolução utilizada. Versões mais novas da mesma linguagem incorporam tais características. Entretanto, o código fonte do sistema foi mantido na versão anterior para permitir maior facilidade de adaptação do sistema, já que é mais fácil adaptar um programa escrito em linguagem com um conjunto mais reduzido de funções específicas de uma versão única. Além do aspecto da portabilidade, influiu na decisão o objetivo final de incorporar esse sistema em outros de maior porte. Tal tarefa é mais fácil se o programa é desenvolvido em linguagem próxima do padrão. Entretanto uma tradução imediata do programa não é descartada, tanto para versões mais nova do Turbo Pascal (como 4.0 ou 5.0) como para linguagens como C, bastante usada em Computação Gráfica, ou outra da mesma filosofia do Pascal. A tradução é simples, concentrando-se todo o esforço em aspectos como rotinas gráficas, entrada e saída e manipulação de arquivos em disco.

A estrutura de dados empregada se mostrou eficiente, principalmente na proposta elaborada para o algoritmo de superfície escondidas, onde a economia de memória foi bastante grande pelo uso de listas encadeadas.

O desenvolvimento das estruturas de dados do sistema sem a aplicação do conceito de tipos abstrados de dados causou uma perda de generalidade prejudicial a posteriores desenvolvimentos. Entretanto, a opção por abandonar tal abordagem veio da necessidade da máxima redução de tempo de processamento, que seria substancialmente alterada no caso da utilização de ocultamento da informação.

6.2 Sugestões

O sistema em si possui muitos aspectos em que pode ser melhorado, em especial na manipulação de "patches" de Bézier e no tratamento mais elaborado da interface com o usuário. Entretanto o seu maior valor está na expansão e utilização do SMS como subsistema de outros de maior porte.

Uma expansão imediata seria incorporar ao sistema a definição de objetos por outros modelos, como CSG ou B-reps, por exemplo, assim como fazem os sistemas de CAD de grande porte. Vários artigos sugerem esquemas para essa adaptação [Fa 88a]

Logicamente tanto o sistema desenvolvido como o tema de modelagem matemática de objetos permanecem em aberto e sujeitos as modificações que permitam a utilização mais completa e específica da técnica nas diversas aplicações da Computação

Gráfica.

Muitos trabalhos recentes têm se preocupado em desenvolver a matemática dos modelos, de maneira a permitir com maior segurança e eficiência objetivos como subdivisão, conexão e "rendering" de superfícies definidas matematicamente [Fa 88a] [Ma 88]. Outra possibilidade de ampliação das possibilidades do tipo de modelagem aqui estudada reside na aplicação de algoritmos de processamento paralelo, visando uma redução do tempo de processamento dos algoritmos que, em sua maioria são excessivamente lentos, prejudicando a manipulação interativa de programas dessa natureza.

Uma área de pesquisa tem se preocupado em desenvolver regras combinatórias que permitam, a partir da obtenção de resultados locais em trechos da superficie, generalizar o processo por regras eficientes para todo o objeto sendo modelado. Tais técnicas envolvem conceitos matemáticos apurados, como noções de Geometria Diferencial e Topologia [Mi 89].

Outro tema a ser abordado se refere ao desenvolvimento de algoritmos paralelos de geração e manipulação gráfica, uma vez que num sistema de modelagem pode-se identificar muitas atividades que poderiam ser executadas a um mesmo tempo, com ou sem concorrência de estruturas de dados e arquivos.

Outra necessidade de expansão reside em se tentar adaptar o sistema a uma aplicação especifica de modelagem matemática. Tal atividade é apoiada pela documentação e modularidade do sistema.

BIBLIOGRAFIA

[Ba 79] Baer, A.; Eastman, C.,; Henrion, M.; "Geometric Modeling: A Survey" - Computer-Aided Design, vol. 11, no. 5, set 1979, p 253- 272.

[Bar 82] Barsky, B. A. - "End Conditions and Boundary Conditions for Uniform B-spline Curve and Surface Representations" - Computers in Industry - 3 (1982), p 17-29.

[Bar 84] Barsky, B. A. - "A description and Evaluation of various 3-D Models" - IEEE CG&A - jan 84 p38-51.

[Bar 85] Barsky,B. A; DeRose, T. D. - "The Beta2-spline :A Special Case of the Beta-spline Curve and Surface Representations" - IEEE CG&A, set 85 p. 46-57.

[Bo 81] Bohn, W. - "Generating the Bézier points of B-spline curves and surfaces" - Computer Aided Design - vol 13, no. 6, nov 81, p. 365-366.

[Boo 72] De Boor, C. - "On Calculating with B-splines" - Jour. Approx. Theory, (1972) 6, p.50-62.

[Boo 88] De Boor, C; Hollig,K. - "B-splines Without Divided Differences" - GEOMETRIC MODELING: ALGORITHMS AND NEW TRENDS - Gerald E. Farin (editor) - SIAM 1987, p 21-27.

[Ca 85] Casale, M. S.; Stanton, E. L.; "An Overview of Analytic Solid Modeling" - IEEE CG&A, fev. 85, p 45-56.

[Cat 78] Catmull,E; Clark, J. - "Recursively Generated B-spline Surfaces on Arbitrary Topological Meshes" - Computer Aided Design, vol 10, no 6, nov 1978.

[Ch 83] Chiyodura, H.; Kimura, F. - "Design of Solids with Free-Form Surfaces" - Computer Graphics , jul. 83, p 289-298

[Co 80] Cohen, E.; Lyche, T; Riesenfeld, R. - "Discrete B-Splines and Subdivision Techniques in Computer Aided Geometric Design and Computer Graphics" - Computer Graphics and Image Processing - 14, 1980, p. 87-111.

[Con 87] Connolly, A. B.; "An Application of Algebraic Topology to Solid Modeling in Molecular Biology" - The Visual Computer, no. 3, 1987, p 72-81.

[Coq 87] Coquillart,S.; "A Control-Point-Based Sweeping Technique" - IEEE CG&A ,nov. 87, p. 36-45.

[Do 78] Doo, D. ; Sabin, M. - "Behaviour of recursive division surfaces near extraordinary points" - Computer Aided Design - vol 10, no.6, nov 78, p. 356-360.

[Fa 88a] Farin, G. E. (editor) - GEOMETRIC MODELING: ALGORITHMS AND NEW TRENDS - SIAM 1987, p.

[Fa 88b] Farin, G. E. - CURVES AND SURFACES FOR COMPUTER AIDED GEOMETRIC DESIGN - A Pratical Guide - Academic Press , 1988, 334p.

[Fo 84] Foley, J. D.; Van Dam, A. FUNDAMENTALS OF INTERACTIVE COMPUTER GRAPHICS - Addison-Wesley Pub. Co. - 1985.

[For 79] Forrest, A. R. - "On the Rendering of Surfaces" - Computer Graphics - Ago.79 p.253-259.

[For 72] Forrest, A. R.; "On Coons and Other Methods for the Representation of Curved Surfaces" - Comp. Graph. and Ima. Proc. (1972) 1, 341-359.

[Fou 82] Fournier, A.; Fussell, D. ; Carpenter, L.; "Computer Rendering of Stochastic Models" - Commun. ACM, vol. 25, no. 6, jun. 1982, p 371-384.

[Ga 83] Gane, C; Sarson,T ; ANALISE ESTRUTURADA DE SISTEMAS - livros Tecnicos e Cientificos, 1983.

[Gi 78] Giloi, W. K.; INTERACTIVE COMPUTER GRAPHICS - Data Structures, Algorithms, Languages - Prentice-Hall, 1978.

[Go 88] Goldman, R. N. - "The Role of Surfaces in Solid Modeling" - em GEOMETRIC MODELING: ALGORITHMS AND NEW TRENDS - Gerald E. Farin (editor) - SIAM 1987, p 69 -90.

[Goo 86] Goodman, T. N. T.; Unsworth, K. - "manipulating Shape and Producing Geometric Continuity in Beta-spline Curves " - IEEE CG&A - fev. 86 - p 50-56.

[Gr 75] Griffiths, J. G. - "A Data-structure for the Elimination of Hidden Surfaces by Patch Subdivision" - Computer Aided Design - vol. 7, no.3, jul 1975 - p. 171-178.

[Gr 78] Griffiths, J. G. - "A Surface Display Algorithm" - Computer Aided Design - vol. 10, no. 1, jan 1978, p.65-73.

[Gr 84] Griffiths, J. G. - "A Depth-coherence scanline algorithm for displaying curved surfaces" - Computer Aided Design - vol 16, no. 2, mar. 1984, p. 91-101.

[Ha 87] Harrington, S - COMPUTER GRAPHICS - A Programming Approach, McGraw Hill, 2a. edição, 1987, 465 p.

[Ka 82] Kajiva, J. T. "Ray Tracing Parametric Patches" - Computer Graphics, vol 16, no. 3, jul 1982, p 245-254.

[La 80] Lane, J. M.; Carpenter, L. C.; Whitted, T.; Blinn, J. F. - "Scan Line Methods for Displaying Parametrically Defined Surfaces" - Communications ACM - vo. 23, no. 1, jan 1980, p.23-34.

[Li 88] Lichten, L.; Samek, M. - "Integrating Sculptured Surfaces into a Polyhedral Solid Modeling System" - em GEOMETRIC MODELING: ALGORITHMS AND NEW TRENDS - Gerald E. Farin (editor) - SIAM 1987, p. 109 - 129.

[Ma 85] Maguirre, M. C. - "A Review of Human Factors Guidelines and Techniques for the Design of Graphical Human-computer Interfaces" - Comput. & Graphics, vol. 9, no. 3, 1985, p 221-235.

[Ma 88] Mäntylä, M. - AN INTRODUCTION TO SOLID MODELING - Computer Science Press - 1988 - 401p.

[Mi 89a] Minghim, R.; Teixeira, E. P. - Linguagens para descrição de Formas - RT/DCA 028/89 - FEE - UNICAMP - 1986, 115p.

[Mi 89b] Minghim, R; - Representação de Curvas e Superficies - RT/DCA 029/89 - FEE - UNICAMP - nov.87, 85 p.

[Mi 89c] Minghim, R - Modelagem de Objetos em Cena - RT/DCA 027/89 - FEE - UNICAMP - nov 1987.

[Mir 89] Miranda, J.; Tavares, G. - MÉTODOS SIMPLICIAIS EM COMPUTAÇÃO GRAFICA - 17o. Coloquio Brasileiro de Matemática - SBM - 1989.

[Na 88] Nahas, M.; Huitric, H.; Saintourens, M. - "Animation of a B-spline figure" - The Visual Computer (1988) 3, p.272-276.

[Ne 83] Newman, W. M.; Sproull, R. F. - PRINCIPLES OF INTERACTIVE COMPUTER GRAPHICS - 2a. edição - Prentice-Hall - 1983.

[Oh 83] Ohno, Y. - "A hidden line elimination method for curved surfaces" - Computer Aided Design , vol 15, no. 4, p 209-216.

[Pa 88] Page-Jones, M.; PROJETO ESTRUTURADO DE SISTEMAS - McGraw-Hill, 1988 - 396p.

[Pe 86] Penna, M. A.; Patterson, R. R. - PROJECTIVE GEOMETRY AND ITS APPLICATIONS TO COMPUTER GRAPHICS - Prentice-Hall, Enlewood Cliffs, 1980, 403 p.

[Pu 87] Pueyo, X.; Brunet, P. - " A Parametric-Space-Based Scan-Line Algorithm for Rendering Bicubic Surfaces" - IEEE CG & A - nov. 1987, p. 17-25.

[Re 80] Requicha, A. A. G.; " Representation for Rigid Solids: Theory, Methods and Systems " - Computing Surveys, vol. 12, no. 4, dez. 1980.

[Ro 76] Rogers, D. F.; Adams, J. A. - MATHEMATICAL ELEMENTS FOR COMPUTER GRAPHICS - McGraw-Hill - 1976

[Ro 83] Rogers, D. F.; Satterfield, S. G.; Rodriguez, F. A. - "Ship Hulls, B-Spline Surfaces and CAD/CAM" - IEEE CG&A - dez. 1983, p 37-45.

[Rock 87] Rockwood, A. - "A Generalized Scanning Thecnique for Display of Parametrically Defined Surfaces" - IEEE CG & A - ago. 1987 - p. 15-26.

[Sc 82] Schweitzer, D.; Cobb, E. S. - "Scanline Rendering of Parametric Surfaces" - Computer Graphics - vol 16, no.3, jul 1982 - p. 265-271.

[Sri 81] - Srihari, S. N. ; "Representation of Three-Dimensional Digital Images" - Computing Surveys. vol 13, no. 4, dez 1981, p 438-465.

[Swe 86] Sweeney, M. A. J.; Bartels, R. H. - "Ray Tracing Free-Form B-Spline Surfaces" - IEEE CG & A - fev. 1986, p. 41 - 49.

[To 86] Tozzi, C. L. - PAC - PROJETO AUXILIADO POR COMPUTADOR - texto 1 - I EBAI - Papirus/Editora UNICAMP - 1986.

[Va 85] Vanderschel, D.; "A Hierarchical Data Structure for Representing the Spatial Decomposition of 3D Objects" - IEEE CG&A, vol. 5, no. 4, abril 1985, p 24 - 41.

[We 85] Weber, H. R. - "Meditation on Man-Machine Interfaces or our Personal Role in Graphics Dialogue Programming" - Comput. & Graphics - vol. 9, no. 3, 1985, p. 237 - 245.

[Wo 87] Woodward, C. D. - "B2-splines: A Local Representation for Cubic Spline Interpolation" - The Visual Computer - (1987) 3, p 152-161.

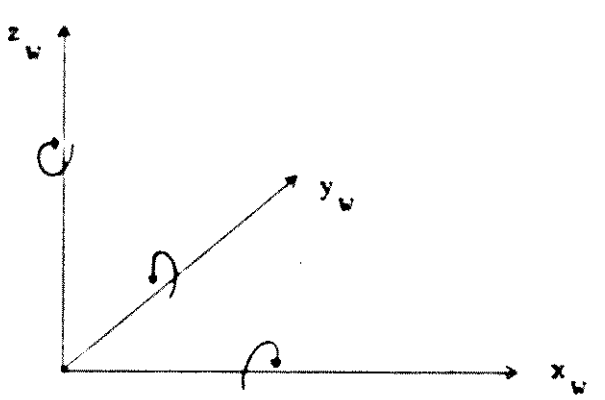
APENDICE 1
VISUALIZAÇÃO E PERSPECTIVA
(fonte principal: [Ne 83])

A.1 Visualização

É usual em sistemas de modelamento a definição de objetos segundo um sistema de coordenadas conveniente à manipulação dos dados de entrada. Esse sistema é chamado Sistema de Coordenadas do Mundo Real. Para a visualização do objeto, é conveniente usar um outro sistema de coordenadas cartesianas, chamado Sistema de Coordenadas do Observador, melhor adequado para a observação do objeto sendo modelado.

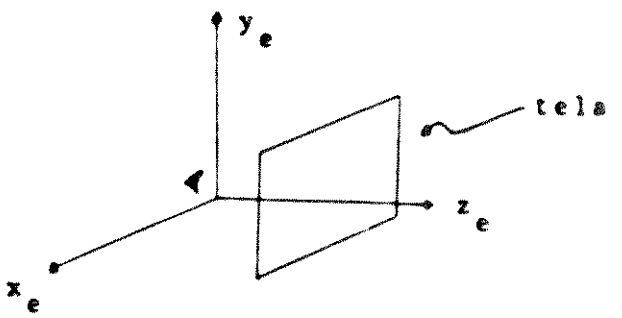
O processo de visualização consiste em determinar um conjunto de transformações (rotação, translação, escala) que converta o sistema de coordenadas do mundo real para o sistema de coordenadas do observador. As convenções (clássicas) utilizadas para as posições dos eixos são:

- sistemas de coordenadas mundo:



sentido positivo de rotação: horário, olhando_se para o centro do sc a partir do eixo.

- sistema de coordenadas do observador:



Em algum momento da transformação de visualização, que transporta um sistema para o outro, é necessário fazer um ajuste de convenção, porque a orientação do eixo z é invertida de um sistema para outro. O sistema do observador será chamado **e** e o mundo de **w**.

Para se obter a transformação e visualização existem dois tipos de convenção:

1. **Convenção direta**, ou da esquerda para direita ('MC')

Nesta convenção as coordenadas no sistema do observador são obtidas pela equação matricial:

$C_e = Mt_{ew} * C_w$, onde:

$C_e = [x_e\ y_e\ z_e\ 1]^t$, coordenadas de um ponto no espaço, expressa no sistema de coordenadas do observador.

Mt_{ew} - Matriz de transformação das coordenadas mundo para as do observador, obtida pela convenção esquerda para direita.

$C_w = [x_w\ y_w\ z_w\ 1]^t$, coordenadas mundo dos vertices (pontos a serem transformados)

Obter a matriz Mt consiste basicamente em determinar as transformações a serem realizadas no sistema do observador até que ele coincida com o sistema do mundo real.

Essas transformações são realizadas através de operações sucessivas de translação, escala e rotação com relação a quaisquer dos eixos.

O ajuste na convenção de orientação é feito pela inversão da direção do eixo z, ou seja, fazendo $z_e = -z_w$, que corresponde a uma operação de escala com parâmetros $x_e = 1$, $y_e = 1$, $z_e = -1$.

Na convenção direta, corresponde a realizar como primeira operação de transformação, aquela indicada pela matriz:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

As demais operações são dadas por:

- translação:

$$trans(tx,ty,tz) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Assim, $C_e = trans(tx,ty,tz) * C_s$

- rotação:

$$rot(x,a) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(a) & sen(a) & 0 \\ 0 & -sen(a) & cos(a) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$rot(y,a) = \begin{pmatrix} cos(a) & 0 & -sen(a) & 0 \\ 0 & 1 & 0 & 0 \\ sen(a) & 0 & cos(a) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$rot(z,a) = \begin{pmatrix} cos(a) & sen(a) & 0 & 0 \\ -sen(a) & cos(a) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Assim, $C_2 = rot(eixo,ângulo) * C_1$

2. Convenção inversa ou direita para esquerda ('CM')

Nesta convenção, as coordenadas no sistema do observador são dadas pela equação matricial:

$C_o = C_u * Mt_{uo}$, onde:

$C_o = [x_o\ y_o\ z_o\ 1]$

Mt_{uo} - Matriz de transformação das coordenadas mundo para o observador pela convenção direita para esquerda. Para obtê-la, o sistema de coordenadas mundo é movimentado pela operação inversa de movimentação do ponto até coincidir com o sistema de coordenadas. (ver [Ne 83])

As matrizes de rotação e translação para esta convenção são as transpostas das matrizes da convenção anterior. Neste caso, a multiplicação pela matriz de ajuste de convenção:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

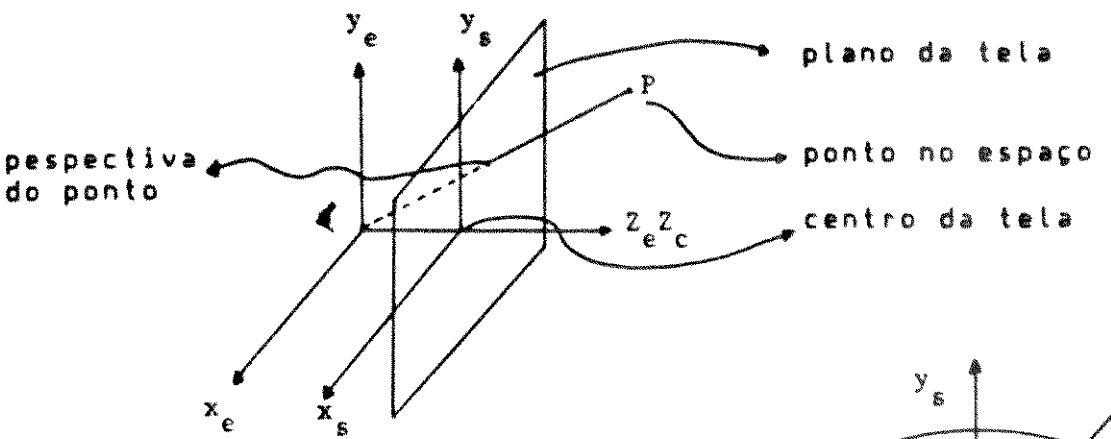
é a última operação a ser realizada.

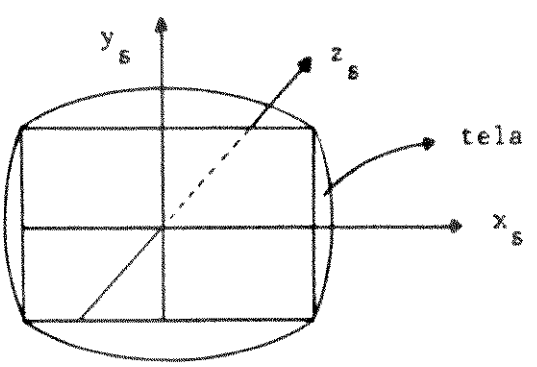
- As operações consecutivas de transformações são refletidas por multiplicação das matrizes de transformação individuais.

A.2 Perspectiva

A perspectiva de um ponto consiste em projetar o ponto no espaço a ser visualizado sobre um plano (localização do plano da tela no sistema do observador).

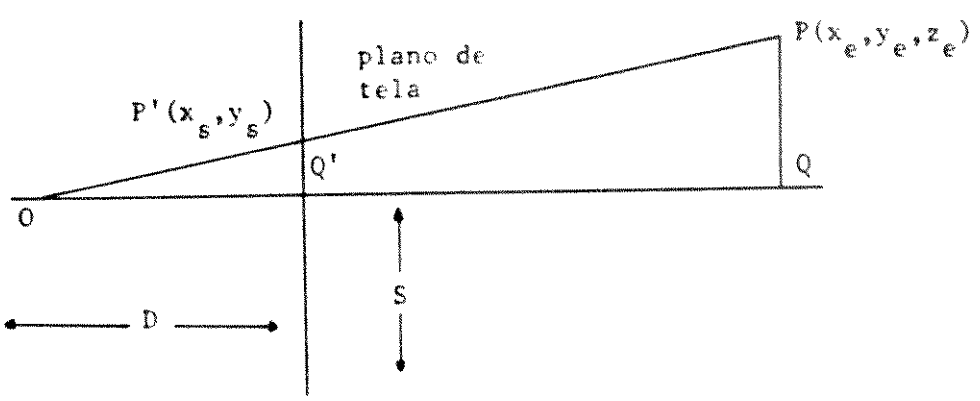
Isso é feito traçando-se uma reta que une o ponto a ser projetado ao observador (origem do sistema). O ponto onde a reta corta o plano é a projeção, como mostra a figura a seguir:





posição dos eixos, relativos à tela

A visão do sistema, segundo projeção ortogonal no plano $z_e\ y_e$, fornece:

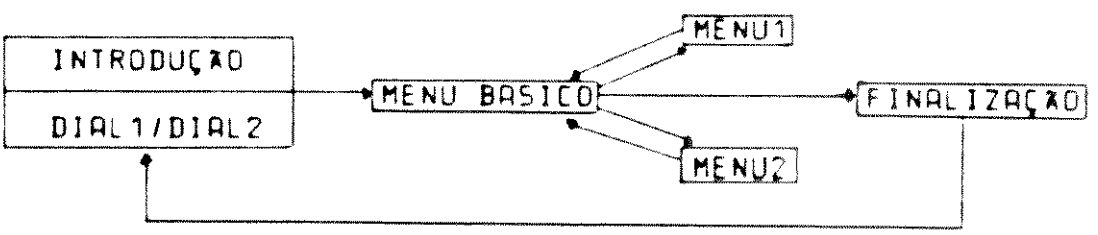


Por semelhança de triângulos:

$$\frac{x_e}{D} = \frac{x_e}{z_e} \qquad \frac{y_e}{D} = \frac{y_e}{z_e} \qquad x_e = \frac{D\ x_e}{S\ z_e} \qquad y_e = \frac{D\ y_e}{S\ z_e}$$

APENDICE 2
RESUMO DAS FUNÇÕES DA INTERFACE DE ENTRADA DE DADOS DO SMS

O esquema abaixo apresenta a sequência de operações possíveis de se executar pela interface que auxilia a entrada de malhas de controle para as superficies modeladas pelo sistema.



INTRODUÇÃO

Na introdução da edição de malha de controle, o programa pergunta se o usuário deseja manipular uma malha nova (DIAL1) ou uma malha já existente (DIAL2).

DIAL1

No diálogo 1 o programa pergunta que nome o usuário deseja dar à malha nova.

O usuário deve fornecer o nome da malha, um incremento para as coordenadas x, y e z e o numero de pontos de controle nas duas direções de variação dos parâmetros.

O incremento das coordenadas serve para orientar o "passo" do cursor na edição, uma vez que se trabalha com numero reais.

Se o usuário fornecer um nome de malha de controle que já

existe, o programa pede para confirmar, já que malha com mesmo nome é destruida.

DIAL2

O diálogo 2 pede ao usuário o nome da malha de controle que ele deseja manipular. Se o arquivo da malha citada não existir, o programa repete a introdução da malha.

Os arquivos devem estar no mesmo diretório do disco em que se encontra o programa objeto.

Tanto a partir de DIAL1 quanto de DIAL2, o programa entra no MENU BASICO.

MENU BASICO

Corresponde ao menu de edição. Neste ponto o usuário possui na tela:

- as informações:
  - os dois planos de edição de malhas (xy e xz),
  - a malha de controle projetada,
  - as coordenadas x, y e z do ponto no espaço onde o cursor está localizado,
  - o modo de edição atualmente em curso,
  - os indices, nas direções u e v, do último ponto da malha introzido e
  - os limites máximos dos índices da malha de controle, nas duas direções de variação dos parâmetros.
- as opções:
  - Limpa Janela de Edição: Descongestiona os planos de

edição da malha, apagando a janela e re-inicializando os eixos.

Introduz Ponto de Controle: entra no modo de edição de ponto. Se a malha está completa, fornece uma mensagem e volta ao menu básico. Do contrário, libera o cursor para caminhar sobre os planos de edição (no modo cursor) ou pede as coordenadas, em valores reais, do ponto a ser introduzido (no modo coordenada). Os pontos são introduzidos em ordem; para cada índice na direção u, são introduzidos os pontos para todos os índices na direção v.

Elimina último Ponto: retira o último ponto de controle introduzido da estrutura de dados e da malha projetada, e atualiza as informações de tela.

Altera Malha de Controle: permite ao usuário mudar a posição de um ponto de controle da malha. Um cursor caminha sobre a malha projetada. O usuário localiza o ponto a ser alterado posicionando o cursor sobre a projeção do ponto. Se o cursor não estiver bem localizado sobre o ponto, o programa busca os pontos mais próximos daquele selecionado. Se mais de um ponto for selecionado, o programa lista todos os pontos, e pede ao usuário que escolha um deles, ou nenhum. Após a seleção do usuário, o programa entra no modo de edição de ponto normal.

Muda Parâmetros de Tela: Fornece ao usuário um menu de opções que alteram os parâmetros de edição (MENU1).

Muda Visualização: Fornece ao usuário um menu de opções que alteram a visualização da malha de controle projetada (MENU2).

Destroi Malha de Controle: Elimina da base de dados as malhas de controle escolhidas pelo usuário.

A tecla (esc) para finalizar a leitura de malha.

MENU1

Possui as opções de mudança de parâmetros vários de edição e tela. São as seguintes:

Altera Deslocamento: Permite ao usuário mudar os incrementos através dos quais o cursor caminha "sobre" os planos de edição.

Escala Edição: Aumenta ou diminui o "campo", em tela, que os planos de edição abrangem.

Altera Modo de Edição: Troca os modos de edição de cursor para coordenada ou vice-versa.

Coloca/Retira Eixos: Os eixos de coordenadas do sistema mundo estão inicialmente posicionados sobre a figura da malha de controle. O usuário pode, com essa opção, alterar a posição dos eixos, tirando da região da figura. Nesse caso, apenas uma orientação dos eixos de coordenadas aparece no canto esquerdo da janela de projeção.

A tecla (esc) para voltar ao MENU BASICO.

MENU2

Diz respeito às opções do usuário para alterar a visualização da malha de controle, que posteriormente será refletida na visualização da superfície avaliada. São elas:

Nova Transformação: Recupera de disco outra tranformação de visualização, desde que o usuário saiba o nome do arquivo que deseja (por exemplo, a visualização utilizada para uma outra malha).

Rotação ou Translação: Com o uso das teclas de seta do teclado usual, o usuário pode realizar sucessivas rotações e

translações, a incrementos constantes, em torno dos três eixos de coordenadas.

Escala: O usuário pode aumentar ou diminuir a imagem da malha projetada.

A tecla (esc) para retornar ao MENU BASICO.

FINALIZAÇÃO

O programa fornece sucessivamente os dados da malha de controle introduzida e da sua imagem, e pergunta ao usuário se deseja gravar os resultados, que compreendem a própria malha de controle e a visualização obtida.

Em seguida fornece as opções de continuar editanto a malha, gerar a superfície ou terminar o programa.

OBSERVAÇÃO: Os arquivos mínimos para executar corretamente o programa de interface são: o código objeto do progrma (de nome "sup_int.com") e a visualização "default" (de nome "default.mt").

APENDICE 3
TABELAS DE CLASSIFICAÇÃO DE QUADRILATEROS
Fonte: (OH 83)

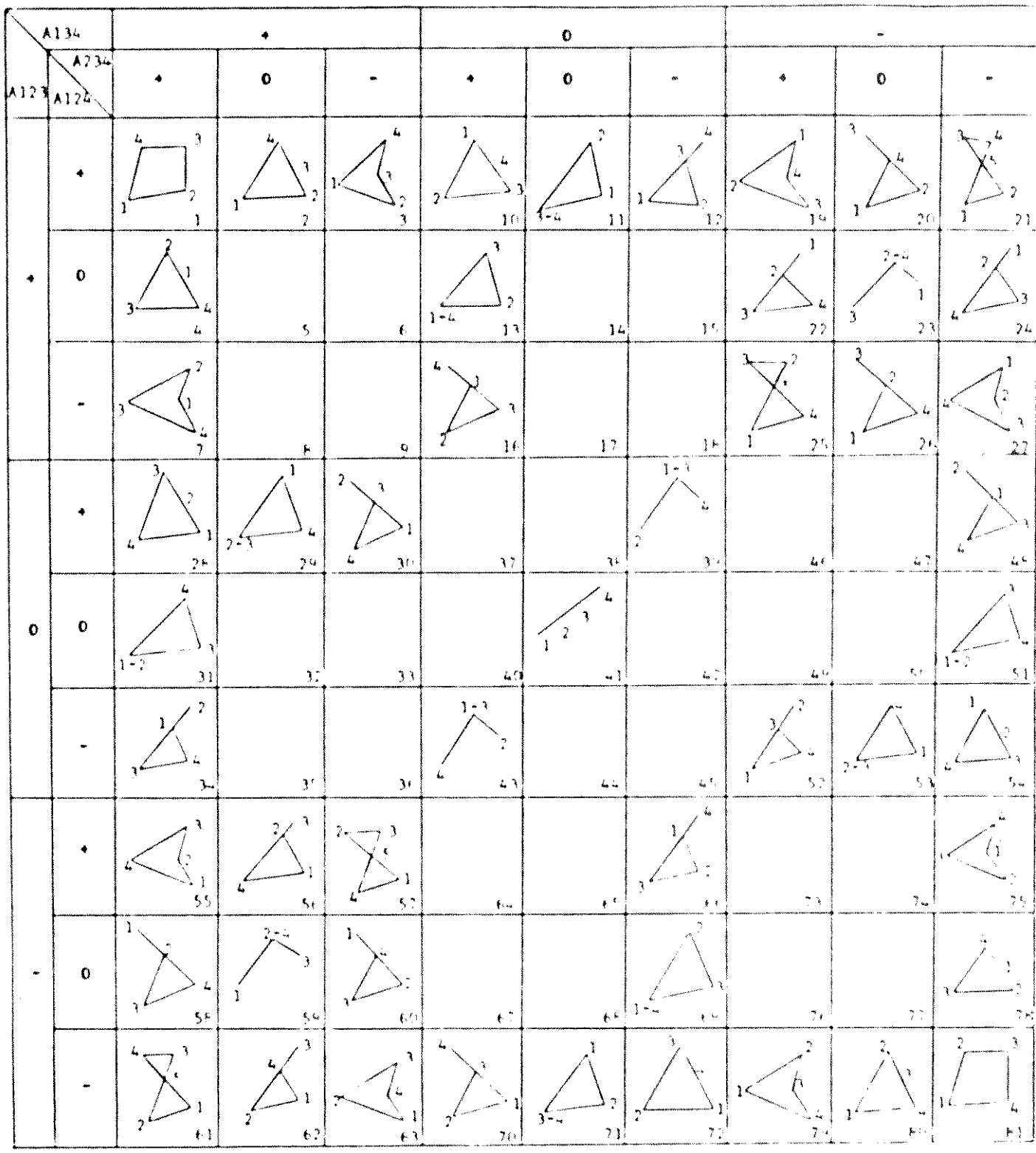


Figura 1: Padrões de quadriláteros projetados. Células em branco indicam que a combinação é impossível. O número no canto inferior esquerdo é o código correspondente.

A123 \ A134 / A234 / A124	+			0			-		
	+	0	-	+	0	-	+	0	-
+	+	123. 134 1	124 2	123. 134 3	123 10	123 11	123 12	124. 234 19	124 20	125. 354 21
	0	234 4	5	6	123 13	14	15	234 22	nenhum 23	243 24
	-	123. 134 7	8	9	123 16	17	18	145. 235 25	142 26	142. 243 27
0	+	134 28	134 29	134 30	37	38	nenhum 39	46	47	143 48
	0	134 31	32	33	40	nenhum 41	42	49	50	143 51
	-	134 34	35	36	nenhum 43	44	45	143 52	143 53	143 54
-	+	124. 234 55	124 56	154. 253 57	64	65	132 66	73	74	132. 143 75
	0	234 58	nenhum 59	243 60	67	68	132 69	76	77	243 78
	-	152. 345 61	142 62	142. 243 63	132 70	132 71	132 72	132. 143 79	142 80	132. 143 81

Figura 2: Triângulos que constituem os quadriláteros projetados mostrados na figura 1.

APÊNDICE 4

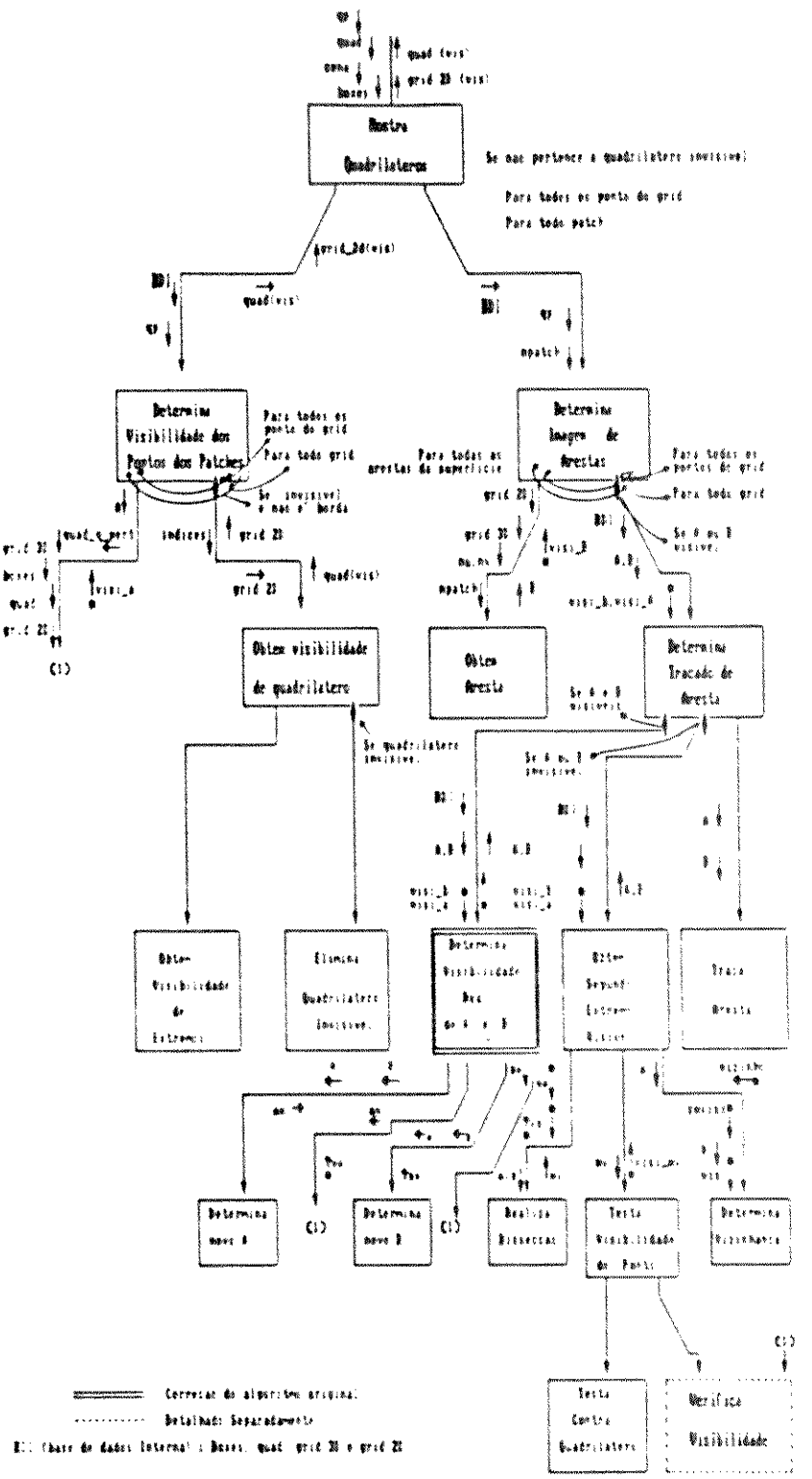


Figura 1: Diagrama de Estrutura da Apresentação de Quadriláteros, segundo lógica implementada pelo autor.

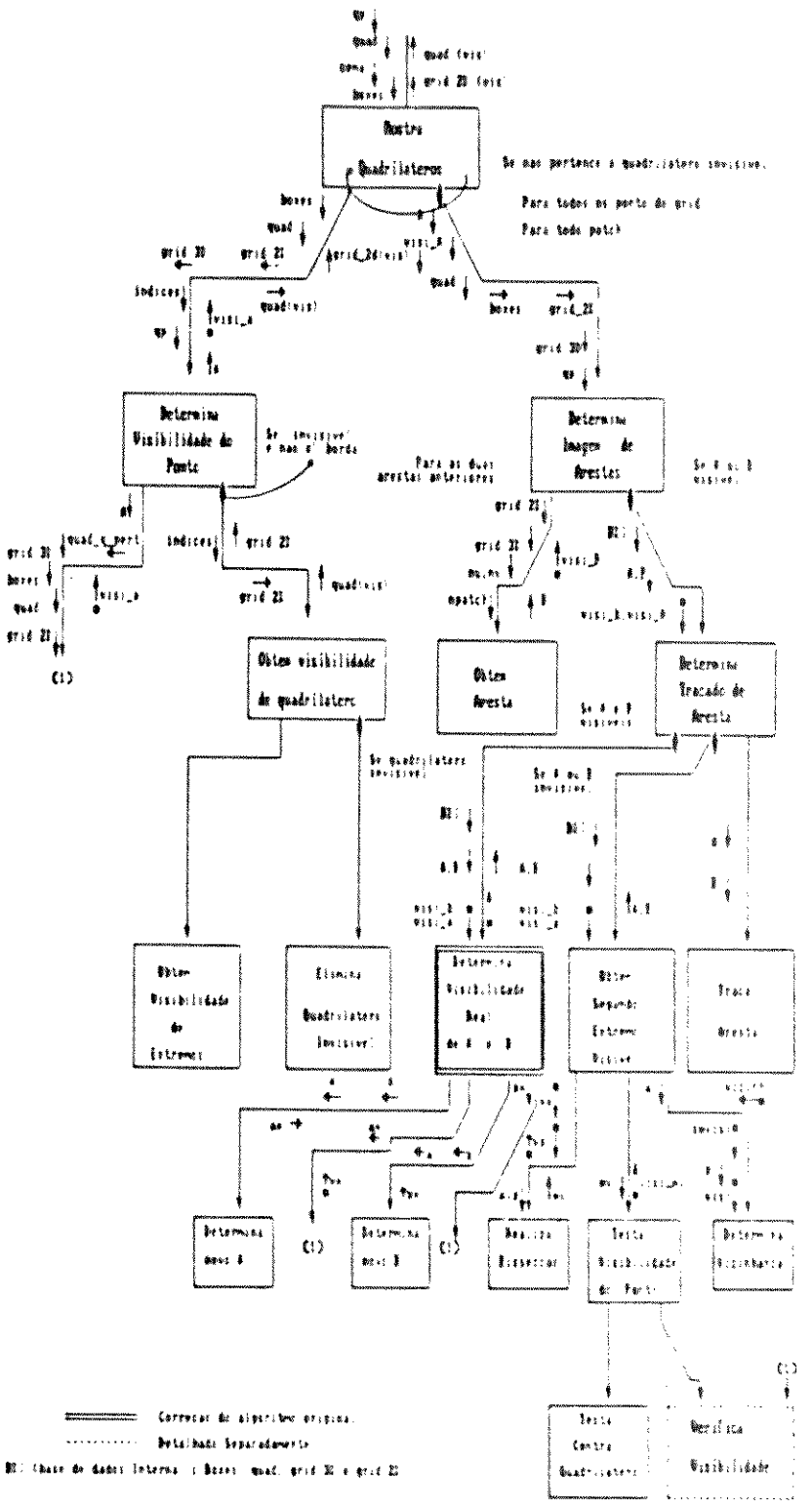


Figura 2: Diagrama da Apresentação de Quadriláteros, segundo sugestão do autor, e implementada no SMS.

APÊNDICE 5

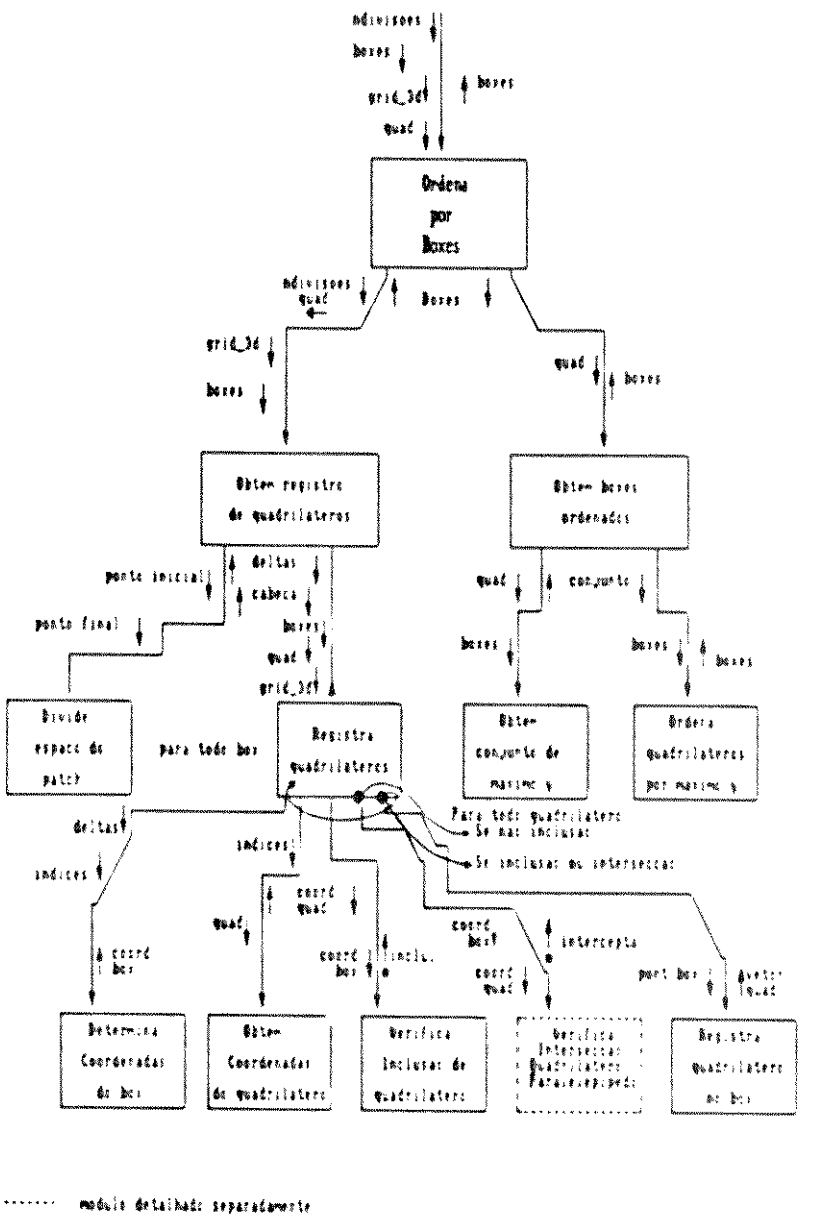


Figura 1: Diagrama de Estrutura da separação dos quadriláteros em regiões do espaço ("boxes"), e posterior ordenação.

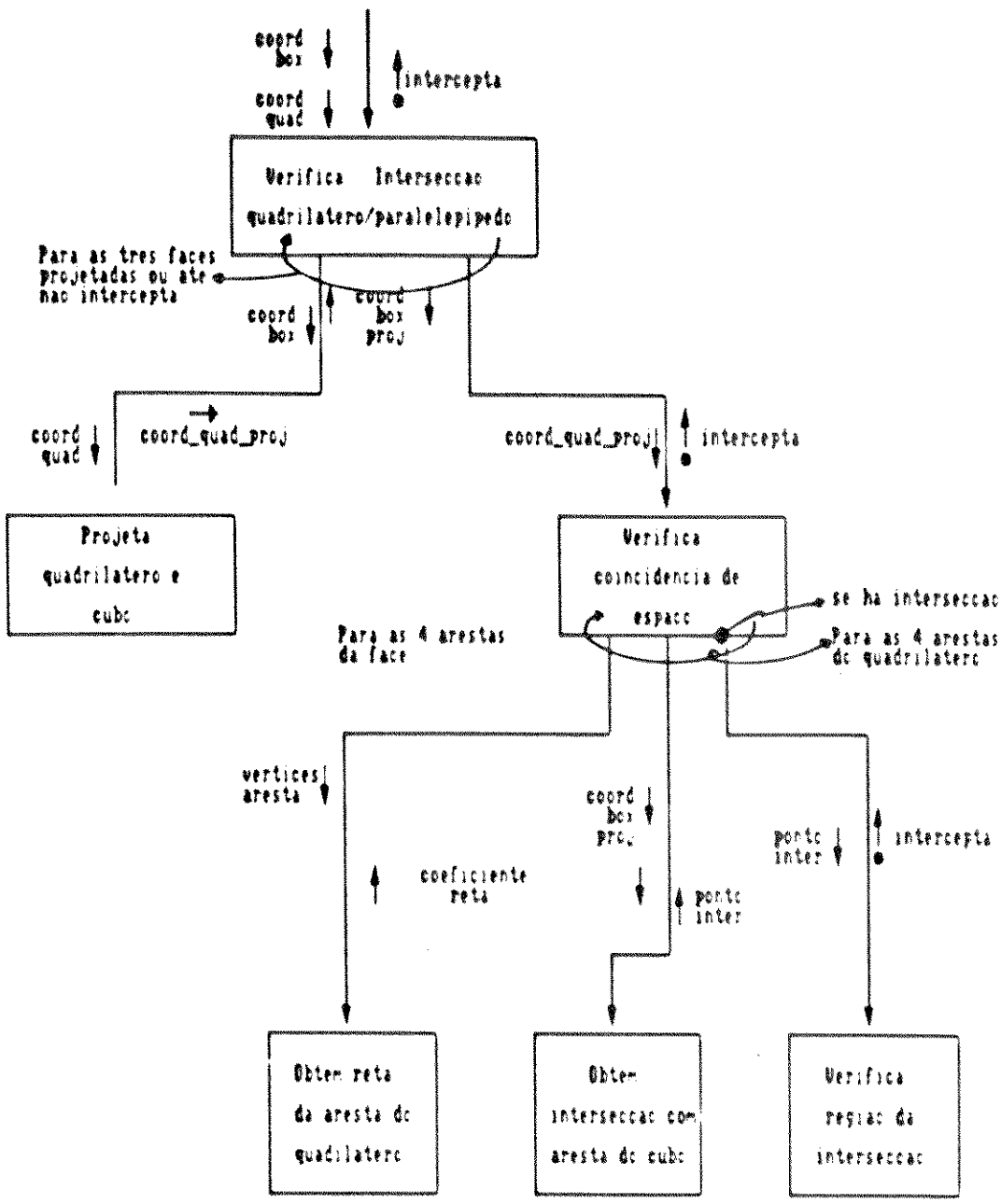


Figura 2: Diagrama de Estrutura da Verificação de Intersecções.

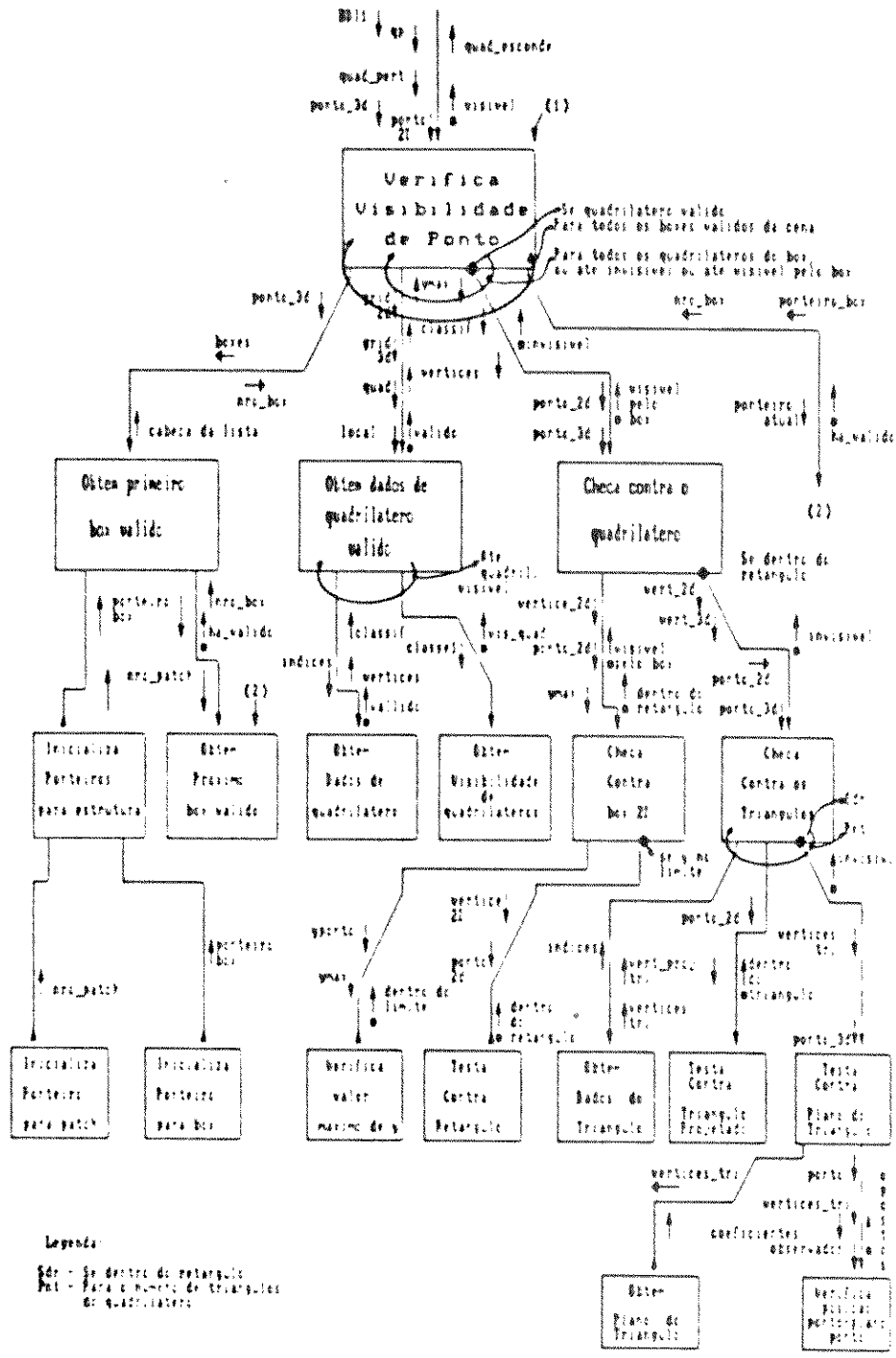


Figura 3: Diagrama de Estrutura da Verificação de Visibilidade de um ponto.