UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE UM AMBIENTE PARA GERAÇÃO DE SUPERFÍCIES 3D COM USO DE SPLINE BÉZIER

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

FERNANDA ANDRADE BORDALLO DA SILVA

BLUMENAU, JUNHO/2000

2000/1-22

PROTÓTIPO DE UM AMBIENTE PARA GERAÇÃO DE SUPERFÍCIES 3D COM USO DE SPLINE BÉZIER

FERNANDA ANDRADE BORDALLO DA SILVA

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

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Dalton Solano dos Reis — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Dalton Solano dos Reis

_____

Prof.

_____

Prof.

AGRADECIMENTOS

A minha mãe, Carmen Lúcia, por sempre ter acreditado e me apoiado durante estes anos em que me dediquei a formação profissional.

Ao meu namorado Mike, pela ajuda com o Delphi e pelo apoio durante todo tempo de realização deste trabalho.

Ao meu orientador, prof. Dalton Solano dos Reis, por ter dado várias idéias de como fazer o trabalho.

A empresa em que trabalho, por ter compreendido a importância deste trabalho para minha formação.

SUMÁRIO

LISTA DE FIGURAS

RESUMO

Este trabalho apresenta um estudo sobre uma das técnicas para modelagem de objetos em três dimensões, a técnica de *Bézier*. A fundamentação teórica possibilita o entendimento da equação de *Bézier* para a suavização de curvas e a construção de superfícies. A revisão biográfica também fundamenta alguns tipos de projeções, bidimensonais e tridimensionais, para a visualização do objeto modelado. Através desta pesquisa foi possível o desenvolvimento de um protótipo de manipulação de superfícies 3D com representação em dois tipos de projeções, em perspectiva e projeções ortogonais. Este documento descreve todo o processo de implementação da fórmula de *Bézier* e suas particularidades, junto com os componentes usados para a representação.

ABSTRACT

This work present an study about method object's modeling in three dimensions, it´s Bézier's method. The theoretical base give the entretenement od Bézier´s equation to better curve and the surface's built . The bibliographical revision ground any kind of projections, bi-directional and three-dimensional, to zoom the modeled object. Through this research be possible development of a prototype of surface's manipulation 3D with two kinds of representation projection, and projection perpendicular's (orthographic) perspective. This docs describe all process of formula's Bézier implementation, together the components used to the representation.

1 INTRODUÇÃO

Este trabalho aborda a representação e visualização de superfícies em três dimensões. A modelagem de superfícies 3D é necessária em diversas áreas, entre elas a modelagem de objetos que não existem no mundo real, ou seja, a projeção e a visualização de um objeto criado ([COR1994]). As superfícies 3D também são usadas no estudo de objetos que já existem, para que estes possam ser aperfeiçoados.

As superfícies podem ser modeladas utilizando um conjunto de objetos que, interligados, formam uma malha. Estes objetos podem ser representados por poliedros ou *splines*. Onde, *splines* são técnicas usadas para suavização de curvas através de pontos de controle, equações paramétricas, malha de polígonos, etc. ([FOL1990]).

Este trabalho adota a técnica de suavização de curvas por pontos de controle e equações paramétricas, para representação de superfícies, chamada curva de *Bézier*.

Equações paramétricas constituem o meio mais utilizado para representação de objetos em modelagem geométrica. Assim como as curvas *Bézier* tem um polígono característico, a superfície *Bézier* tem um poliedro[1] característico. Pontos da superfície *Bézier* são dados por uma simples extensão da equação geral para pontos da curva *Bézier* ([MOR1985]).

Sabe-se ainda, que superfícies bicúbicas[2] são as formas mais frequentemente utilizadas, devido ao fato de que superfícies de menor grau apresentam pouca flexibilidade sobre o controle de suas formas, enquanto que superfícies de maior grau podem introduzir torções indesejáveis, além de requisitarem maior esforço computacional ([FOL1990]).

Para a visualização da superfície tridimensional, pode-se utilizar as projeções ortogonais (em duas dimensões), as quais possibilitam o efeito de fotografia na representação de figuras 3D, ou seja, mostra de diversos ângulos do objeto. Permitindo preservar o máximo possível as medidas de elementos geométricos da figura tridimensional na representação 2D ([PER1989]).

---

[1] Poliedro é um conjunto de faces.

[2] Superfícies cujo polinomial de terceiro grau (cúbico) é formado por duas variáveis.

Uma outra forma de visualização tridimensional, em perspectiva, foi conseguida através da utilização dos próprios componentes do ambiente de desenvolvimento (ver seção 5.1) e através da exportação dos objetos para um arquivo no formato DXF (*Drawing Interchange Format*) ([BRO1995]).

1.1 MOTIVAÇÃO

A representação de superfícies curvas em projetos computacionais sempre foi uma incógnita, no que diz respeito a recursos computacionais disponíveis e na reprodução “fiel” do objeto a ser modelado ([COR1994]). Por esta razão, torna-se importante o estudo de representações de superfícies aliada à ciências exatas (matemática), que determina em muitas vezes a melhor opção para a representação do objeto e consequentemente um menor custo computacional.

1.2 OBJETIVOS

Estudo e implementação de um protótipo de software para geração de superfícies 3D com o uso de malha de *splines Bézier* com visualização por projeções ortogonais e projeções em perspectiva com a utilização dos componentes do ambiente de desenvolvimento e exportação para arquivos DXF.

1.3 ORGANIZAÇÃO DO TEXTO

Este trabalho de conclusão de curso está organizado de forma a permitir maior entendimento sobre o assunto proposto, da seguinte maneira:

Capítulo 2 – Conceito básico de *Splines* – fornece os conceitos básicos de curvas e superfícies, dando mais ênfase à técnica específica, de *Bézier*, para geração de curvas e superfícies;

Capítulo 3 – Conceitos de Projeções – descreve os conceitos básicos de tipos de projeções que podem ser utilizadas na visualização de objetos tridimensionais num plano. Este capítulo fornece um conhecimento introdutório para se entender Câmera Sintética, descrito no capítulo seguinte;

Capítulo 4 – Ambiente de visualização 3D – dá uma visão geral sobre o conceito de câmera sintética e formação de arquivos de formatos DXF que possibilitam a exportação para ambientes de visualização tridimensional;

Capítulo 5 – Desenvolvimento do Protótipo – descreve a especificação e implementação do protótipo, ou seja, a definição e códigos fontes dos cálculos;

Capítulo 6 – Conclusão – mostra o resultado do trabalho e suas limitações e possíveis melhoramentos.

2 CONCEITOS BÁSICOS DE SPLINES

As curvas podem ser representadas de várias formas, entre elas está a representação por polinomiais paramétricas.

Curvas polinomiais paramétricas determinam pontos numa curva 3D usando três polinomiais com um parâmetro *u*, um para cada x, y e z. Embora vários graus de polinomiais possam ser usados, o caso mais comum é a representação por polinomiais cúbicas (que possui maior potência igual a três). Neste caso é mais conhecido como curva cúbica ([FOL1990]).

Superfícies polinomiais bivariáveis paramétricas determinam coordenadas para pontos numa superfície curva usando três polinomiais bivariáveis, uma para cada x, y e z. Os limites da superfície são curvas polinomiais paramétricas. Assim como as curvas, os polinomiais bivariáveis também podem usar vários graus, porém o caso mais comum é a polinomial cúbica para as duas variáveis. Este tipo de representação de superfície é conhecido como superfícies bicúbicas paramétricas ([FOL1990]).

Superfícies bicúbicas são as formas mais frequentemente utilizadas, devido ao fato de que superfícies de menor grau apresentam pouca flexibilidade sobre o controle de suas formas, enquanto que superfícies de maior grau podem introduzir torções indesejáveis, além de requisitarem maior esforço computacional ([FOL1990]).

2.1 CONCEITOS DE CURVAS E SURPERFÍCIES

Uma curva é um conjunto limitado de pontos, cujas coordenadas são dadas por funções matemáticas simples e contínuas de um parâmetro no formato:

a) $x = x(u)$,
b) $y = y(u)$,
c) $z = z(u)$.

A variável paramétrica u é definida pelo intervalo $u \in [0, 1]$, e o sentido positivo da curva é o sentido no qual u é incrementado. A curva é limitada porque possui dois pontos finais definidos, um para $u = 0$ e outro para $u = 1$ ([MOR1985]).

O elemento básico usado para modelar uma superfície é o *patch*. O *patch* é um conjunto de pontos que delimitam uma curva, cujas coordenadas são dadas por funções matemáticas simples e contínuas de dois parâmetros no formato:

a) $x = x(u,w)$,

b) $y = y(u,w)$,

c) $z = z(u,w)$.

As variáveis paramétricas *u* e *w* são definidas ao intervalo $u, w \in [0, 1]$ ([MOR1985]).

2.1.1 FORMAS DE REPRESENTAR SUPERFÍCIES OU VOLUMES

A representação de formas geométricas complexas podem ser realizadas a partir de elementos lineares como segmento de reta e a aproximação das curvas através de poliedros com muitas faces.

A representação de curvas realizada através de poliedros, dar-se-á pela junção de vários polígonos planos que resulta num poliedro muito próximo de uma figura curva.

Para satisfazer um grau de realismo desejável do objeto modelado, requer-se um número muito alto de faces do poliedro, o que acarreta num aumento na estrutura de dados e no processamento do mesmo ([BOR1989]) e ([PER1989]).

2.2 CURVAS E SUPERFÍCIES BÉZIER

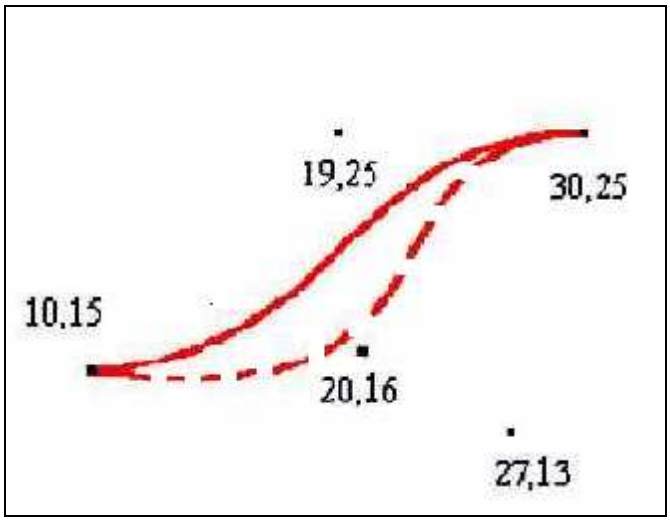
Curvas (*spline*) e superfícies *Bernstein Bézier*, ou melhor, curvas e superfícies *Bézier* constituem uma das primeiras tentativas de desenvolver uma interface para CAD mais flexiva e intuitiva. Esta técnica foi usada por alguns anos pela Renault no desenho da parte externa e de painéis externos de automóveis ([BAR1992]).

2.2.1 CONCEITOS E FUNDAMENTOS DA CURVA BÉZIER

A técnica *Bézier* especifica dois pontos, inicial, final e outros dois pontos intermediários. Os pontos inicial e final são determinados por vetores P_1P_2 e P_3P_4. A função interpola os pontos inicial e final e aproxima os outros dois ([FOL1990]).

Esta técnica provê o controle global dos pontos, ou seja, movendo qualquer ponto de controle toda a curva será modificada. Isto pode ser notado na figura 1, pois na alteração do nó (20,16), curva contínua, para (27,13), curva tracejada, todo o segmento de curva foi alterado ([NEW1991]).

Figura 1: Curva *Bézier*.



O procedimento de aproximação desenvolvido por *Bézier*, não é explicitamente uma função estática, mas é uma fórmula, onde cada termo envolve um produto ([HIL1990]).

A curva *Bézier* $p(t)$, é baseada em $(n + 1)$ pontos de controle p_0, p_1, ..., p_n, e é representada pelo polinômio:

$$\boldsymbol{p}(u) = \sum_{i=0}^{n} \boldsymbol{p}_i B_{i,n}(u) \ , \qquad u \in [0,1] \ .$$

A qual é conhecida como Bernstein Polinomial para Hill ([HIL1990]), Mortenson ([MOR1985]), e para Foley ([FOL1990]), e como uma função combinatória para Newman ([NEW1991]), onde a função $B_{i,n}(u)$ é definida como:

$$B_{i,n}(u) = C(n,i)u^i(1-u)^{n-i} .$$

Onde $C(n,i)$ é a função coeficiente binomial para $C(n,i) = \dfrac{n!}{i!(n-i)!}$ e 0 (zero) para outros casos ([MOR1985]).

Foley ([FOL1990]), e Mortenson ([MOR1985]), expandem a equação para curvas definidas para 3, 4, 5 e 6 pontos, para se familiarizar com a forma de obter o polinômio.

Lembrando que a ordem é igual ao número de pontos.

Para ordem $n = 3$ pontos, e grau $m = 2$:

$$\boldsymbol{p}(u) = (1-u)^2\,\boldsymbol{p}_0 + 2u(1-u)\,\boldsymbol{p}_1 + u^2\,\boldsymbol{p}_2 .$$

Para ordem $n = 4$ pontos, e grau $m = 3$:

$$\boldsymbol{p}(u) = (1-u)^3\,\boldsymbol{p}_0 + 3u(1-u)^2\,\boldsymbol{p}_1 + 3u^2(1-u)\,\boldsymbol{p}_2 + u^3\,\boldsymbol{p}_3 .$$

Para ordem $n = 5$ pontos, e grau $m = 4$:

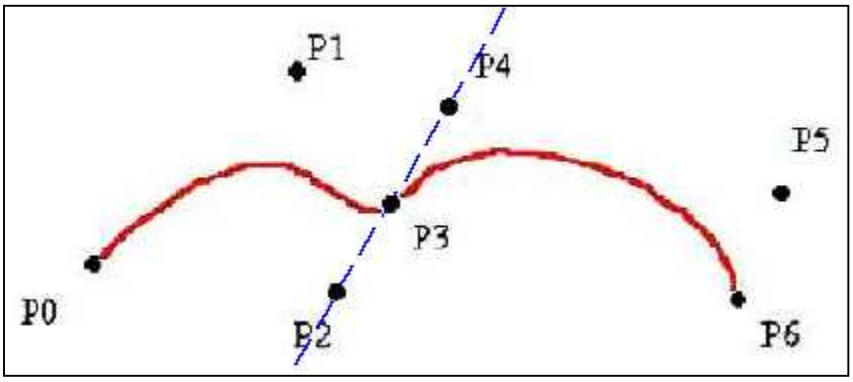
$$\boldsymbol{p}(u) = (1-u)^4\,\boldsymbol{p}_0 + 4u(1-u)^3\,\boldsymbol{p}_1 + 6u^2(1-u)^2\,\boldsymbol{p}_2 + 4u^3(1-u)\,\boldsymbol{p}_3 + u^4\,\boldsymbol{p}_4 .$$

Para ordem $n = 6$ pontos, e grau $m = 5$:

$$\boldsymbol{p}(u) = (1-u)^5\,\boldsymbol{p}_0 + 5u(1-u)^4\,\boldsymbol{p}_1 + 10u^2(1-u)^3\,\boldsymbol{p}_2 + 10u^3(1-u)^2\,\boldsymbol{p}_3 + 5u^4(1-u)\,\boldsymbol{p}_4 + u^5\,\boldsymbol{p}_5 .$$

Na figura 2 são mostrados dois segmentos de curvas *Bézier* com um ponto de interpolação em comum. A continuidade provém deste ponto, quando $k(P_3 - P_2) = k(P_4 - P_3)$, para $k > 0$. Isto é, os três pontos P_2, P_3 e P_4 devem ser distintos e colineares ([FOL1990]).

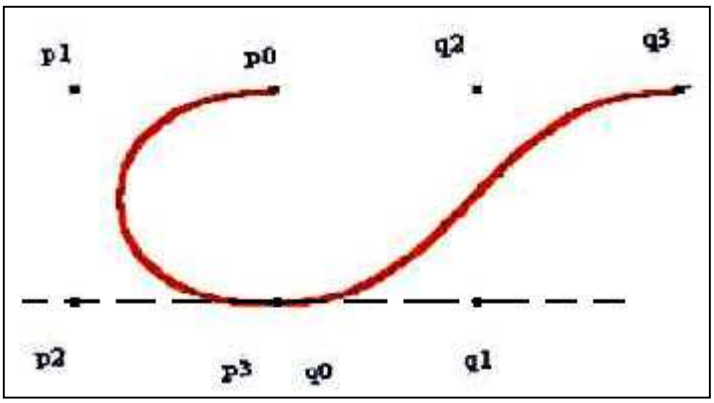
Figura 2: Duas Curvas *Bézier* ligadas pelo ponto *P*3. Os Pontos *P*2, *P*3 e *P*4 são colineares.



Fonte: [FOL1990]

Para Mortenson ([MOR1985]), as condições de continuidade entre os dois segmentos *Bézier* adjacentes de uma curva composta, são relativamente fáceis de especificar. Define-se dois segmentos adjacentes como na figura 3, pelos nós p_i e q_i, a continuidade é estabelecida fazendo com que os nós p_{n-1}, $p_n = q_n$, q_{n+1}, sejam colineares.

Figura 3: Curva *Bézier* composta por dois segmentos, onde *p*2, *p*3=*q*0 e *q*1 são colineares.

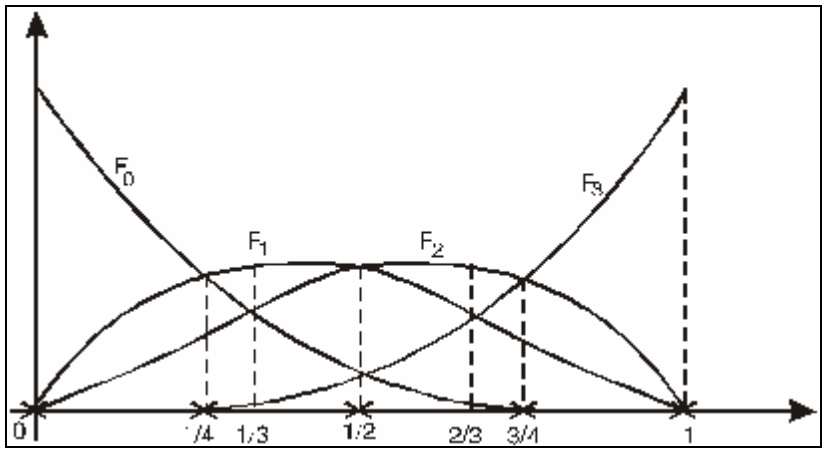


Considere, então, duas curvas de *Bézier* tais que o último ponto da primeira e o primeiro da segunda sejam coincidentes. Conforme viu-se acima, verificou-se que essas curvas podem ser emendadas de forma que a curva união seja suave, bastando que o penúltimo ponto da primeira, o ponto comum às duas e o segundo ponto da outra sejam colineares. Nessas condições tem-se que a derivada em $u = 1$ da primeira curva e a derivada em $u = 0$ da segunda possuem a mesma direção. Isso é o bastante para garantir que no ponto da emenda haverá uma única direção tangente à curva união. Não há, portanto, necessidade, no caso, de fazer com que essas derivadas se igualem.

Assim, dada uma curva de *Bézier*, emendar nela uma outra, de modo que a curva resultante continue suave, é uma simples questão de introduzir, nas posições devidas, pontos de controle que obriguem a condição de colinearidade acima a ser cumprida. Deve ser lembrado, entretanto, que ao fazer isso, modificamos a expressão de ambas as curvas, mesmo que elas já tenham, originalmente, uma extremidade comum e que os pontos acrescentados estejam bem próximos dessa extremidade. Isso se dá porque o controle local de uma curva de *Bézier* não é completo, uma vez que todas as $B_{i,n}$ são não nulas no intervalo [0,1]. Deve-se observar, porém, que o valor máximo de cada $B_{i,n}$ se dá para $t = i/n$ e em torno desse ponto, no intervalo $[i/(n + 1), (i + 1)/(n + 1)]$, o valor de $B_{i,n}$ predomina sobre os demais. Dessa forma o intervalo [0,1] fica particionado em $(n + 1)$ intervalos iguais, em cada um deles preponderando a influência de um p_i diferente. Isso é mostrado na Figura 4 para o caso em que n = 3 [PER 89].

Uma alternativa para melhorar o controle local é exatamente, ao invés de trabalhar com uma única curva de *Bézier*, utilizar várias delas de grau menor, emendadas em extremidades comuns. Observa-se, então, que se for exigido apenas que a curva composta seja suave, pode-se fazer essas emendas de modo simples, apesar da ressalva que fizeram quanto a isso. Se, entretanto, for preciso garantir que essa curva possua um grau maior de diferenciabilidade, fazer as emendas da maneira devida pode ficar complicado ([ROG1990]).

Figura 4 - Curvas de *Bézier* para n=3.



Outro inconveniente das curvas de *Bézier* é que o grau e a própria definição das $B_{i,n}$ dependem do número de pontos de controle. Isso significa que o projetista, ao acrescentar mais um ponto para tentar acertar a forma de um determinado trecho da curva, torna mais complicada a expressão de toda a curva, mesmo onde a influência desse ponto é pequena ([ROG1990]).

Deve-se observar, portanto, que a formulação de *Bezier* para curvas se estende com facilidade para superfícies.

2.2.2 CONCEITOS E FUNDAMENTOS DA SUPERFÍCIE BÉZIER

Assim como as curvas *Bézier* tem um polígono característico, a superfície *Bézier* tem um poliedro característico. Pontos da superfície *Bézier* são dados por uma simples extensão da equação geral para pontos da curva *Bézier* ([MOR1985]), como:

$$\boldsymbol{p}(u,w) = \sum_{i=0}^{m} \sum_{j=0}^{n} \boldsymbol{p}_{ij} B_{i,m}(u) B_{j,n}(w) \qquad u, w \in [0,1].$$

Onde o P_{ij} são pontos que formam uma matriz (m + 1) x (n + 1), e $B_{i,m}(u)$ e $B_{i,n}(w)$ são as definições das curvas *Bézier* ([MOR1985]).

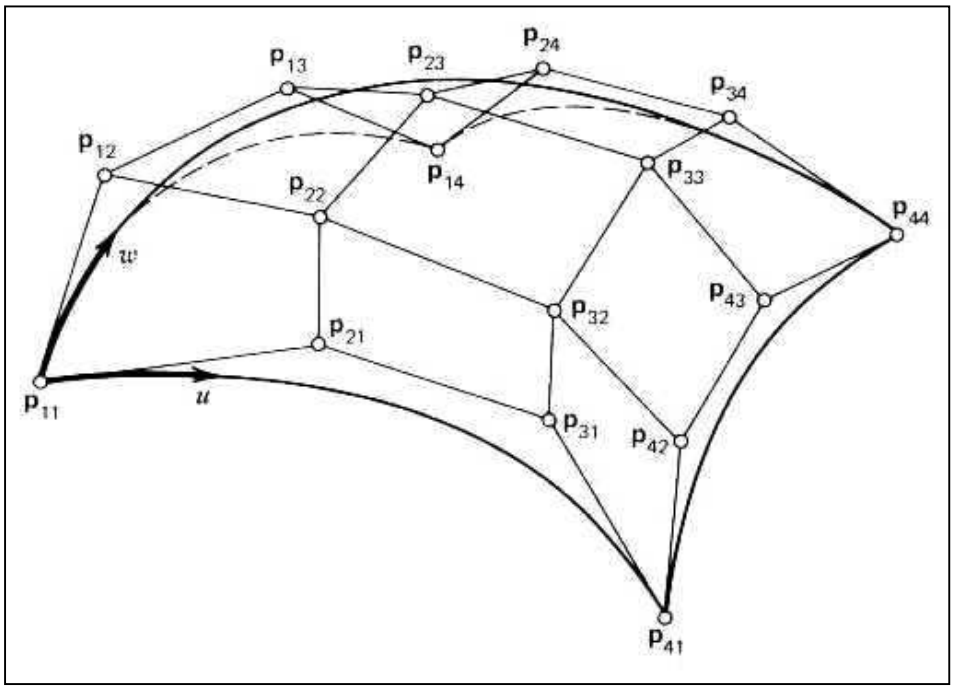
Usando a representação binomial da curva cúbica *Bézier*, a equação matriz para um *patch* definido pelo conjunto de pontos 4 x 4 é ([MOR1985]):

$$\boldsymbol{p}(u,w) = \begin{bmatrix}(1-u)^3 & 3u(1-u)^2 & 3u^2(1-u) & u^3\end{bmatrix} \boldsymbol{P} \begin{bmatrix}(1-w)^3 \\ 3w(1-w)^2 \\ 3w^2(1-w) \\ w^3\end{bmatrix}$$

onde $$\boldsymbol{P} = \begin{bmatrix}\boldsymbol{p}_{11} & \boldsymbol{p}_{12} & \boldsymbol{p}_{13} & \boldsymbol{p}_{14} \\ \boldsymbol{p}_{21} & \boldsymbol{p}_{22} & \boldsymbol{p}_{23} & \boldsymbol{p}_{24} \\ \boldsymbol{p}_{31} & \boldsymbol{p}_{32} & \boldsymbol{p}_{33} & \boldsymbol{p}_{34} \\ \boldsymbol{p}_{41} & \boldsymbol{p}_{42} & \boldsymbol{p}_{43} & \boldsymbol{p}_{44}\end{bmatrix}.$$

A matriz $\boldsymbol{P}$ contém os pontos do poliedro, e assim, o *patch* da superfície *Bézier*. Esta afirmação é representada na figura 5 ([MOR1985]).

Figura 5: Superfície *Bézier*.



Fonte: [FOL1990]

Na fórmula *Bézier*, apenas quatro pontos ($\boldsymbol{p}_{11}$, $\boldsymbol{p}_{41}$, $\boldsymbol{p}_{14}$ e $\boldsymbol{p}_{44}$) fazem parte do *patch*. Os pontos $\boldsymbol{p}_{21}$, $\boldsymbol{p}_{31}$, $\boldsymbol{p}_{12}$, $\boldsymbol{p}_{13}$, $\boldsymbol{p}_{42}$, $\boldsymbol{p}_{43}$, $\boldsymbol{p}_{24}$ e $\boldsymbol{p}_{34}$ controlam a inclinação da curva. Os quatro pontos restantes $\boldsymbol{p}_{22}$, $\boldsymbol{p}_{32}$, $\boldsymbol{p}_{23}$ e $\boldsymbol{p}_{33}$ controlam a inclinação dos cruzamentos junto com o limite da curva do mesmo modo como os pontos traçados para o *patch* bicúbico ([MOR1985]).

Como a figura 5 mostra, a superfície *Bézier* é completamente definida por uma rede de pontos desenhados descrevendo dois grupos de curvas *Bézier* na superfície. Cada curva é definida por um polígono de quatro pontos ou vértices ([MOR1985]).

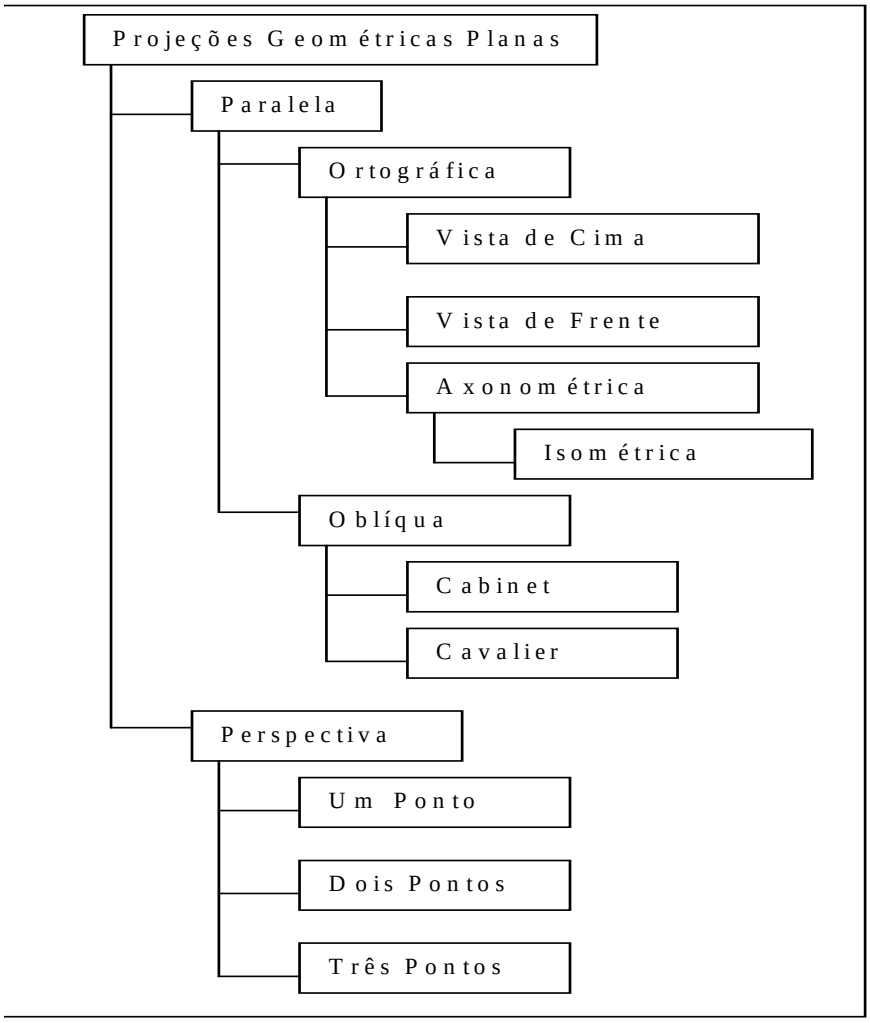
3 CONCEITOS DE PROJEÇÕES

Projeção é a forma de representação de figuras em três dimensões numa superfície plana, como por exemplo, a tela do computador ([BOR1989]). As projeções são usadas para atingir os objetivos abaixo, ainda que isoladamente ([PER1989]):

a) garantir as medidas, da figura tridimensional, na representação 2D;
b) dar uma visão da figura espacial como um todo, possibilitando a idealização do objeto real;
c) permitir um efeito visual parecido com o de uma fotografia.

A projeção de figuras em 3D sobre o plano pode pertencer a duas classes distintas, paralela e a perspectiva ([BOR1989]). Dentro destas classes estão as suas subclasses como mostra a figura 6.

Figura 6 - Hierarquia das projeções.



Projeção Paralela é aquela em que o objeto é inserido ao plano a partir da projeção de seus pontos por meio de linhas paralelas. Este tipo de projeção geralmente é usado para preservar as dimensões das linhas do objeto, porém está técnica não possibilita a visão realista do objeto ([BOR1989]).

Projeção Perspectiva é aquela em que os pontos do objetos são projetados por linhas que convergem para um ponto (centro de projeção), ou seja, como se o centro de projeção estivesse próximo ao objeto ([BOR1989]).

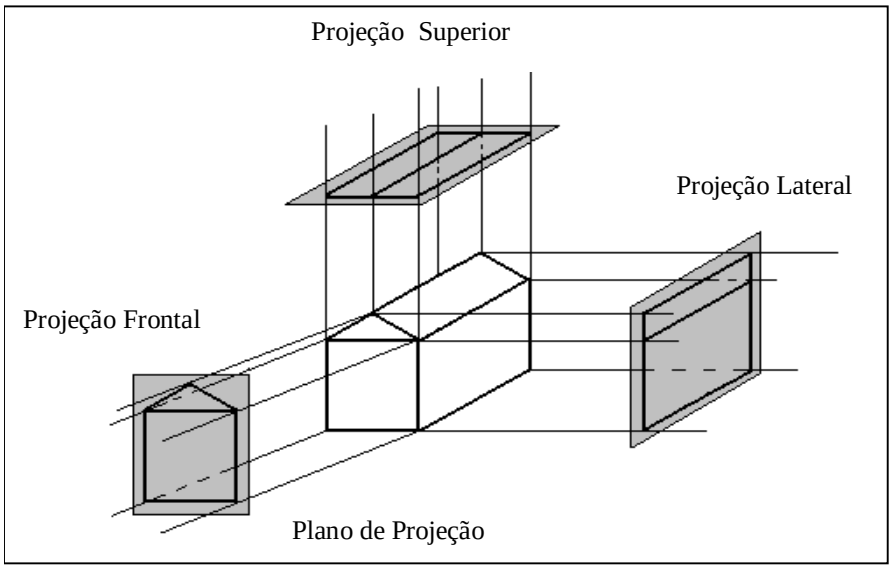
3.1 PROJEÇÃO PARALELA

As projeções paralelas são caracterizadas pelo ângulo formado entre a direção de projeção e o plano de projeção. Existem três tipos mais usados de projeções paralelas: projeção ortográfica, isométrica e oblíqua ([BOR1989]).

3.1.1 PROJEÇÃO ORTOGRÁFICA

Na projeção ortográfica, a direção dos raios projetantes é perpendicular ao plano de projeção, como mostra a figura 7. O cálculo é feito a partir da eliminação de uma das coordenadas 3D, associando as outras coordenadas aos dois eixos do plano de projeção.

Figura 7: Construção de três projeções ortográficas.



Fonte: [FOL1990]

Assim chega-se às seguintes fórmulas ([BOR1989]):

a) vista superior - Não considerando a altura, portanto: $xp = x$, $yp = z$;
b) vista de frente - Não considerando profundidade, portanto: $xp = x$, $yp = y$;
c) vista lateral direita - Não considerando a largura, portanto: $xp = z$, $yp = y$.

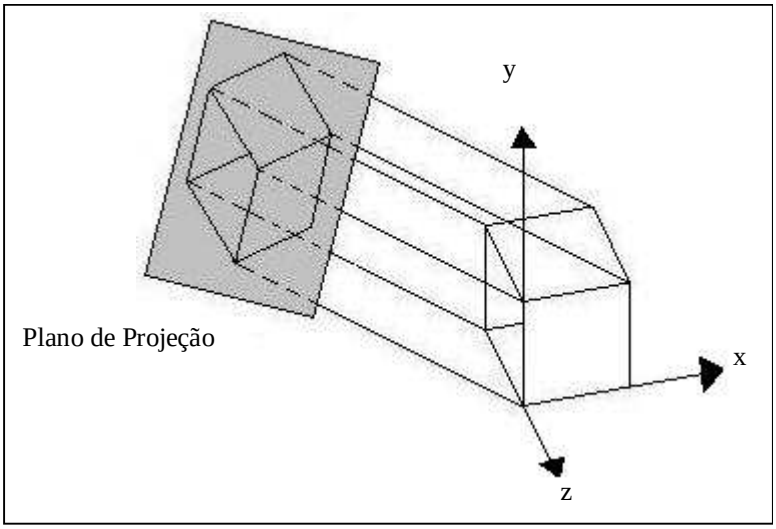
As coordenadas de projeção possuem as mesmas dimensões do objeto. Quando essas coordenadas são colocadas na tela, é normalmente feita uma operação de escala, devido ao fato de que um ponto na tela não tem a mesma dimensão de uma unidade da projeção. Deve-se também definir o ponto que representará o ponto de origem (0,0) ([BOR1989]).

3.1.2 PROJEÇÃO ISOMÉTRICA

O plano de projeção não é perpendicular a nenhum dos eixos principais do objeto. O paralelismo de linhas é preservado, mas não os ângulos, e as distâncias podem ser medidas usando-se um fator de escala para cada eixo ([BOR1989]), ver figura 8.

As coordenadas do ponto projetado, para cada caso possível de projeção isométrica, podem ser facilmente deduzidas analisando-se a projeção dos eixos que definem o mundo ([BOR1989]).

Figura 8 - Um cubo em projeção isométrica.



Fonte: [FOL1990]

Na figura 9 pode-se verificar os seguintes itens ([BOR1989]):

a) cada coordenada *x* e *z* de um ponto no mundo terá um correspondente em *xp* e um em *yp*;

b) a coordenada *xp* do ponto projetado é função das coordenadas *x* e *z* do ponto correspondente;

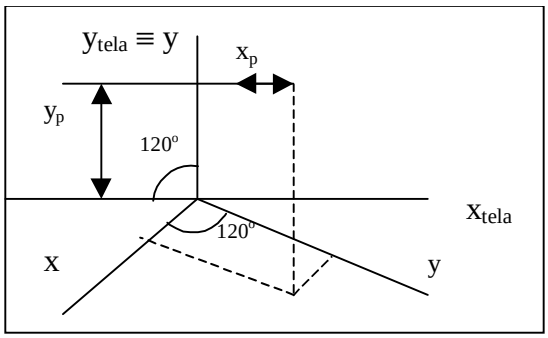
c) a coordenada *yp* do ponto projetado é função das coordenadas *x*, *y* e *z* do ponto correspondente no mundo;

d) à medida em que se anda em *x*, de uma em uma unidade, mantendo fixos *y* e *z*, o ponto projetado (*xp*,*yp*) se desloca:
   - para a esquerda *x* cos(30),
   - para baixo *x* sen(30);

e) à medida em que se anda em *z*, de uma em uma unidade, mantendo fixos *x* e *y*, o ponto projetado (*xp*,*yp*) se desloca
   - para a direita *z* cos(30),
   - para baixo *z* sen(30);

f) à medida em que se anda em *y*, de uma em uma unidade, mantendo fixos *x* e *z*, o ponto projetado (*xp*,*yp*) se desloca para cima nesta mesma proporção.

Figura 9 - Um ponto em projeção isométrica.



Fonte: [BOR1989]

Assim, unindo as seis observações acima, determina-se a fórmula final:

a) $xp = - x \cos(30) + z \cos(30)$,

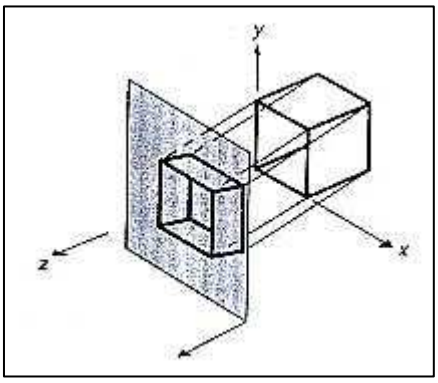
b) $yp = - x \operatorname{sen}(30) - z \operatorname{sen}(30) + y$.

3.1.3 PROJEÇÃO OBLÍQUA

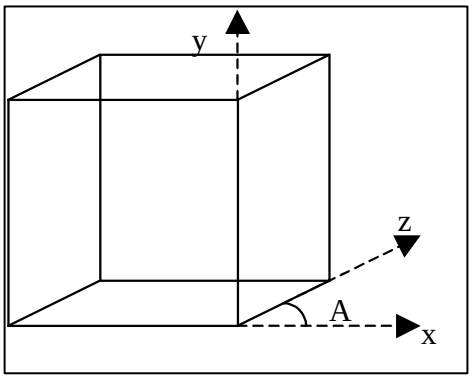
Caracterizam-se por um plano de projeção normal a um dos eixos principais e uma direção de projeção não perpendicular ao plano. Uma maneira simples de entender a fórmula é olhar os desenhos das projeções do cubo na parede ([BOR1989]).

Utilizando a figura 10, fixa-se uma das arestas. A cada unidade deslocada em *x* e *y* por uma aresta do cubo, a projeção também anda de *x* ou *y* no desenho ([BOR1989]).

Figura 10 - Sombra do cubo na parede.



Fonte: [FOL1990]



Fonte: [BOR1989]

A cada unidade que aprofundada em z, a projeção anda:

- no sentido horizontal de (L.cos(A));
- no sentido vertical de (L.sen(A)).

Este L é um fator que depende do grau de acentuação do efeito de profundidade, ou seja, mais ou menos acentuado ([BOR1989]).

Deste modo pode-se chegar a fórmula:

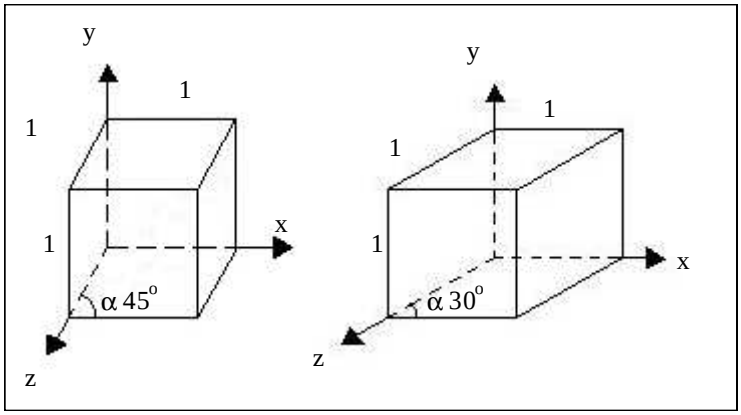
- $xp = x + L\ \cos(A)z$,
- $yp = y + L\ \text{sen}(A)z$.

O valor adequado de L é a cotangente do ângulo, deduzido matematicamente pela projeção ([BOR1989]).

Os dois tipos mais usados de projeção oblíqua são as projeções Cavalier e Cabinet.

Na projeção Cavalier, a direção de projeção forma um ângulo de 45 graus com o plano de projeção, como motra a figura 11.

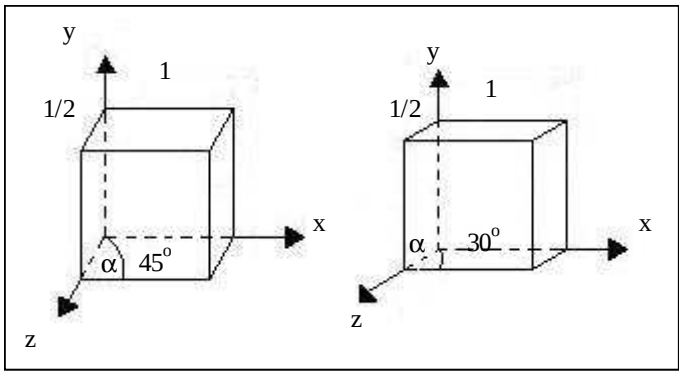
Figura 11 - Projeção Cavalier de cubo em um plano com $z = 0$.



Fonte: [FOL1990]

Na projeção Cabinet, figura 12, a direção de projeção forma um ângulo cuja tangente é 2 (63.4 graus). Consequentemente, linhas perpendiculares ao plano de projeção são projetadas com a metade de seu comprimento.

Figura 12 - Projeção Cabinet de um cubo num plano com $z = 0$.



Fonte: [FOL1990]

3.2 PROJEÇÃO PERSPECTIVA

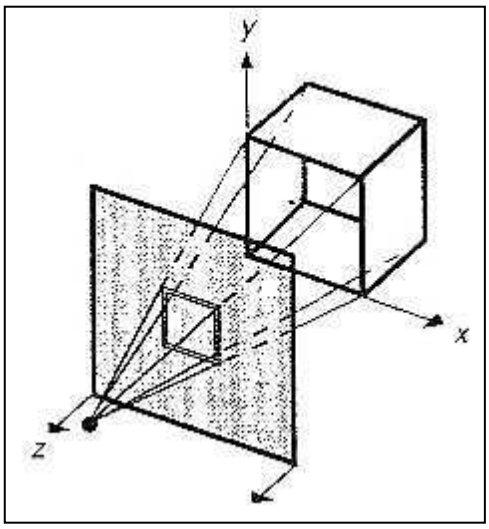
É especificada definindo-se o centro de projeção, para onde convergem as linhas de projeção. Usando uma lanterna próxima ao objeto, forma-se um cone de luz, que produz uma sombra maior que o objeto.

Qualquer conjunto de linhas paralelas de imagens, que não sejam paralelas ao plano de projeção, irá convergir para um ponto de fuga. O ponto de fuga de um conjunto de linhas

paralelas que sejam paralelas a um dos eixos principais é chamado de "ponto de fuga principal".

Na projeção perspectiva as dimensões não são preservadas. Quanto mais distante estiver o objeto do centro de projeção, menor será sua projeção perspectiva. Apenas as linhas que são paralelas ao plano de projeção serão projetadas como paralelas, mostrado na figura 13. Desta forma está técnica não deve ser usada quando existe grande importância em saber as medidas e o formato exato dos objetos, mas sim nos casos em que se deseja realismo.

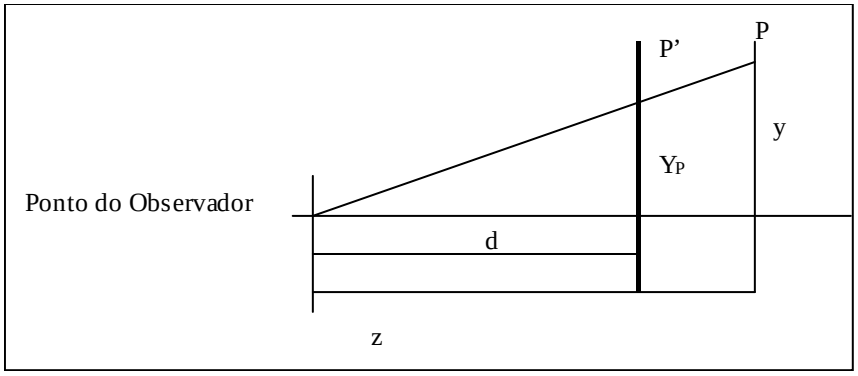
Figura 13 - Projeção em perspectiva do cubo partindo de um ponto paralelo ao eixo *z*.



Fonte: [FOL1990]

Para melhor entendimento, ilustra-se, na figura 14, um observador olhando para um objeto representado pelo ponto *p*. Através desta figura deduz-se o cálculo das coordenadas da projeção por semelhança de triângulos. Desta forma supondo que o observador está na posição (0,0,0) do espaço, temos $\frac{yp}{d}=\frac{y}{z}$. Igualmente pode-se deduzir com vista superior $\frac{xp}{d}=\frac{dy}{z}$, ou seja, $xp=\frac{dx}{z}$ e $yp=\frac{dy}{z}$ ([BOR1989]).

Figura 14 - Perspectiva criada por uma visão através do vidro.



Fonte: [BOR1989]

A matriz utilizada no cálculo da projeção perspectiva em coordenadas homogêneas para o plano (*x*,*y*), é ([ROG1990]):

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/d & 1 \end{bmatrix}.$$

Portanto, a projeção perspectiva é dada por:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/d & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

Multiplicando-se as matrizes, obtém-se:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \\ z/d+1 \end{bmatrix} \quad \text{ou} \quad \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{x}{z/d+1} \\ \frac{y}{z/d+1} \\ 0 \\ 1 \end{bmatrix}.$$

logo: $x' = \frac{x}{z/d+1}$ e $y' = \frac{y}{z/d+1}$, onde:

a) x' e y', correspondem às coordenadas 2D após aplicação da projeção perspectiva;

b) *x*, *y* e *z*, correspondem às coordenadas 3D antes da aplicação da projeção perspectiva;

c) *d*, corresponde à distância do centro de projeção (DCP).

4 AMBIENTE DE VISUALIZAÇÃO 3D

Para a visualização em três dimensões, este trabalho utilizou como recurso um ambiente que oferece as opções de câmera sintética, através da exportação de um arquivo de formato DXF, que possui a codificação da superfície para este ambiente.

A projeção em perspectiva também foi conseguida utilizando métodos dos componentes do próprio ambiente de desenvolvimento, Delphi, que possibilita a representação 3D.

4.1 CÂMERA SINTÉTICA

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

A visualização de objetos tridimensionais num plano consiste em exibir os pontos dos objetos após o cálculo da projeção perspectiva dos mesmos ([ZIN1993]).

Para se calcular os pontos na janela de exibição que irão representar os pontos do objeto, deve-se inicialmente transformar os pontos do sistema de referência do universo para o sistema de referência da câmera, e então aplicar a projeção perspectiva. Por último, mapear as coordenadas do plano de projeções[3] para as coordenadas da janela de exibição[4] .

O universo é um modelo de espaço no qual os objetos a serem exibidos, encontram-se descritos, ou seja, é o espaço disponível que os objetos podem ser representados ([ZIN1993]).

A câmera é a representação atribuída a um observador de um cenário qualquer, ou seja, é a visão do observador de uma determinada imagem e a representação desta em um plano. É com base na sua localização, orientação, entre outros, que se obtém uma imagem como um todo ou uma parte do cenário disponível ([ZIN1993]).

---

[3] Plano em que será projetada uma porção do universo selecionada para visualização.

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

A utilização de uma câmera sintética possibilita ao observador visualizar o objeto através de pontos de vista diferentes (posição, alvo, ângulos e foco) ([REI1995]).

4.2 ARQUIVOS DXF

O arquivo DXF (*Drawing Interchange Format*) é um formato de arquivo produzido pela empresa AutoDesk para a integração com outros programas aplicativos, tais como CorelDraw, AutoCAD, etc.. Este formato é o mais usado na integração CAD (*Computer Aided Design*) e é suportado por praticamente todos os softwares CAD.

Segundo Brown ([BRO1995]), o arquivo DXF é composto por um conjunto de grupos de dados. Cada grupo é composto por duas linhas de texto ASCII. A primeira linha é um valor inteiro no qual identifica o valor do dado na segunda linha. O valor do dado na segunda linha pode ser um valor inteiro, ponto flutuante ou uma cadeia de caracteres. Usando este esquema, grupo de dados que não são reconhecidos pelo software que suporta o arquivo DXF, pode ser facilmente descartado.

Os conjuntos de dados são divididos em seção Corpo, seção Tabelas, seção Blocos e seção Entidades. Esta estrutura pode ser facilmente entendida observando a figura 15.

A seção Corpo contém o conjunto de variáveis associadas ao desenho. Estas variáveis definem, entre outras coisas, a direção do ângulo, arredondamento para distâncias, data de criação, etc..

Na seção Tabelas estão oito tabelas que se encontram zeradas. As tabelas determinam o tipo de linha, o layout (cor, tipo de linha, etc.), a fonte do texto, o tipo de visualização, os estilos de dimensões, qual o sistema de coordenada e um índice único que mantém o nome da aplicação.

Seção Bloco possui a definição dos blocos, através da delimitação pelo grupo BLOCK e ENDBLK.

Seção Entidades mantém a descrição das primitivas do desenho. As seguintes primitivas são suportadas: *line, point, circle, arc, trace, solid, text, shape, insert, attdef, attrib, polyline, vertex, seqend, 3Dface, viewport, dimension, extended entity data*.

Figura 15 - Estrutura do arquivo DXF.

```
Listagem da estrutura de um arquivo ".DXF"

          0                (início seção HEADER)
        SECTION
          2
        HEADER
                     <<<<variáveis de cabeçalho>>>>
          0
        ENDSEC             (fim seção HEADER)
          0                (início seção TABLES)
        SECTION
          2
        TABLES
          0
        TABLES
          2
        VPORT
          70
        (itens que determinam valores de visulização)
                   <<<<itens da tabela mostrados>>>>
          0
        ENDTAB
          0
        TABLE
          2
        LTYPE, LAYER, STYLE, VIEW, UCS, or DWGMGR
          70
        (número máximo dos itens na seção Table)
                   <<<<variáveis da tabela>>>>
          0
        ENDTAB
          0
        ENDSEC             (fim seção TABLES)
          0                (início seção BLOCKS)
        SECTION
          2
        BLOCKS
              <<<<definição dos blocos das entidades>>>>
          0
        ENDSEC             (fim seção BLOCKS)
          0                (início seção ENTITIES)
        SECTION
          2
        ENTITIES
                   <<<<Drawing entities go here>>>>
          0
        ENDSEC             (fim seção ENTITIES)
          0
        EOF                (fim do arquivo)
```

5 DESENVOLVIMENTO DO PROTÓTIPO

Este capítulo descreve a especificação e o desenvolvimento do protótipo de software, que é o resultado da pesquisa realizada nas seções anteriores, mostrados através dos diagramas de contexto e hierárquico funcional. Aqui também são descritos alguns componentes do ambiente de desenvolvimento Delphi que possibilitaram a visualização da superfície tridimensional gerada.

5.1 ESPECIFICAÇÃO DO PROTÓTIPO

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

5.1.1 DIAGRAMA DE CONTEXTO E HIERÁRQUICO FUNCIONAL

Nesta seção tem-se a visão macro do funcionamento do protótipo e suas interações com o meio, (figura 16). A figura 17 mostra todas as funções permitidas pelo protótipo.

Figura 16 - Diagrama de contexto.

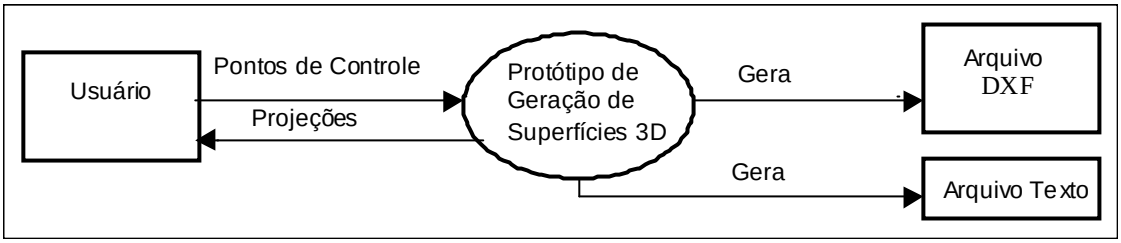
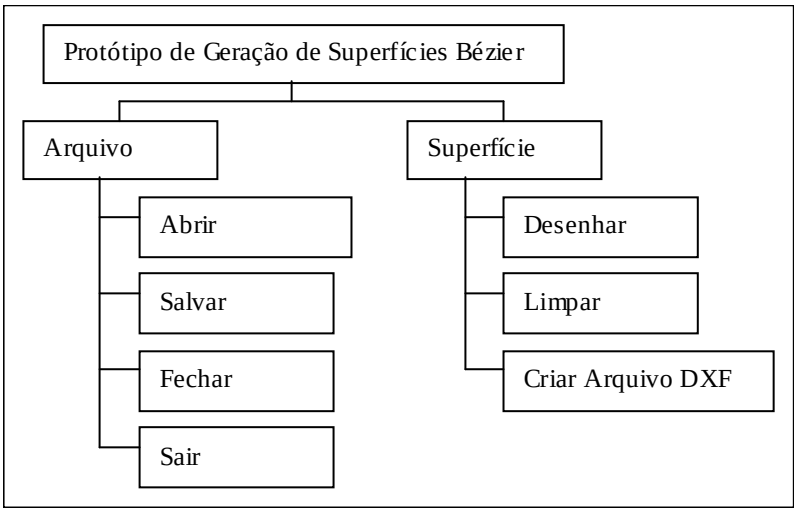


Figura 17 – Diagrama Hieráquico Funcional.



5.2 IMPLEMENTAÇÃO DO PROTÓTIPO

Utilizando a ferramenta de desenvolvimento Delphi versão 5.0 da Borland, foi escrito o código de geração de superfícies com base na fórmula de *Bézier* (ver seção 2.2.1 - $\boldsymbol{p}(u) = (1-u)^3 \boldsymbol{p}_0 + 3u(1-u)^2 \boldsymbol{p}_1 + 3u^2(1-u)\boldsymbol{p}_2 + u^3 \boldsymbol{p}_3$). O Delphi é uma ferramenta de desenvolvimento que utiliza o modo de programação visual usando componentes. Este sistema vem com vários componentes prontos para o uso. O próprio sistema operacional Windows possui alguns componentes, conhecidos por controles. Um controle é tecnicamente uma janela predefinida com um comportamento específico, algumas propriedades e alguns métodos ([CAN1997]).

Além dos controles básicos que foram feitos como os controles do Windows, o Delphi oferece muitos recursos que permitem criar facilmente aplicações gráficas. Os componentes para desenhos do Delphi obedecem sistema de coordenadas dado pela regra da Mão-Direita[5]. Um dos recursos do Delphi é a classe `TCanvas` que possui duas características distintas: manter a coleção de ferramentas de desenho (caneta, pincel e fonte) e um grande número de métodos de desenho. Esta classe possui funções para desenhar figuras geométricas em uma área de desenho dos formulários e de outros componentes ([LEA1998]).

Os componentes do Delphi que possuem o componente `Canvas`, herdam as características da classe `TCanvas`, ou seja, para que um componente possa utilizar os recursos do `TCanvas`, este deverá possuir o componente `Canvas` incorporado. O componente `Image` é um dos componentes que possui o componente `Canvas`. O `Canvas` do `Image` apenas possui métodos de manipulação de desenhos em duas dimensões. Um dos métodos em destaque é o método de suavização de curvas com a técnica *Bézier* chamado `PolyBezier`, entretanto, preferiu-se implementá-la.

Outro componente que também possui o `Canvas` inerente ao seu componente é o `Chart`. Este é geralmente utilizado para o desenvolvimento de gráficos estatísticos, porém o `Canvas` utilizado pelo `Chart` herda características da classe `TCanvas3D` que por sua vez é

---

[5] Sistema que obedece a seguinte ordem: o eixo X cresce para a direita, a eixo Y cresce para baixo e o eixo Z perpendicular ao papel e se afastando do observador.

filho da classe `TCanvas`. Os métodos do `Canvas` do `Chart` para desenhos são similares aos do `Image`, porém em três dimensões ([CAN1997]).

Para o desenvolvimento do protótipo foram utilizados, para a representação da superfície, o componente `Chart`, da biblioteca `TeeChart`, em três dimensões, ou seja, em perspectiva, e o componente `Image` para as projeções ortográficas, descritas no capítulo 3.

Além desses componentes de visualização, foi utilizado componentes como `RadioGroup` para associar os pontos de controle de cada *spline* que compõem a superfície, caixas `Edit` para visualizar e alterar os valores das coordenadas desses pontos, `OpenDialog` para possibilitar a abertura de arquivos, `SaveDialog` para o salvamento da superfície e a criação do arquivo DXF.

Utilizou-se para a demonstração da superfície uma estrutura de 16 pontos de controles, com três coordenadas, dispostos em uma matriz 4x4. A estrutura da matriz pode ser observada no Quadro 1.

Quadro 1 – Estrutura da Matriz de Pontos de Controle

```
Type
  Tpontos = record
    x,y,z : Integer;
  end;
Tmatriz = array [1..4,1..4] of Tpontos;
```

Cada linha e coluna da matriz determinará uma *spline*, formando assim uma superfície formada por 4 *splines* verticais e 4 *splines* horizontais. Esta matriz é inicializada com valores predefinidos que poderão ser alterados. A escolha pela superfície com 16 pontos de controle foi devido ao fato de que a equação de *Bézier* seria de grau 3, ou seja, projetar uma superfície com um nível de complexidade satisfatório.

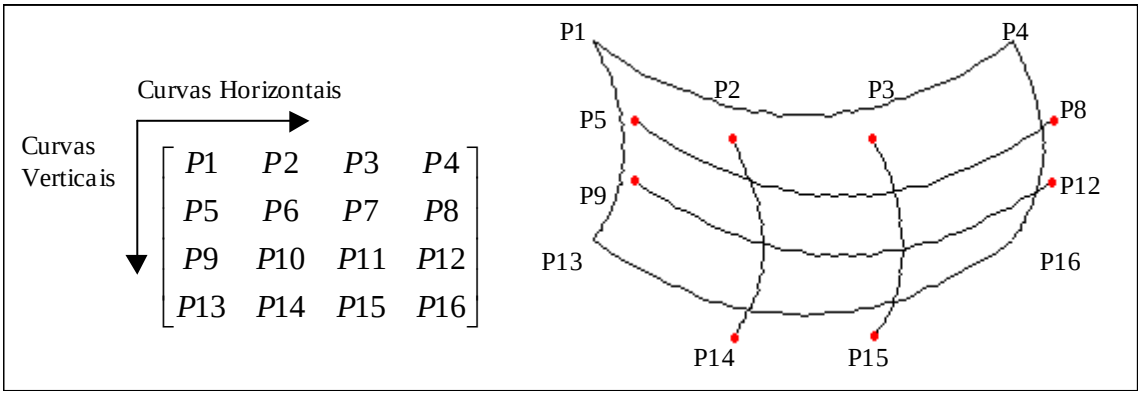
No desenvolvimento, o algoritmo da superfície de *Bézier* (seção 2.2.2) foi dividido em procedimentos separados, onde cada um destes processos possui suas próprias estruturas, não possuindo nenhuma rotina genérica, proporcionando uma análise personalizada de cada método [MEL1990].

Ao longo do trabalho de implementação observou-se algumas particularidades da aplicação da fórmula de *Bézier* para superfícies, que serão apresentadas a seguir no detalhamento da implementação.

5.2.1 DETALHAMENTO DA IMPLEMENTAÇÃO

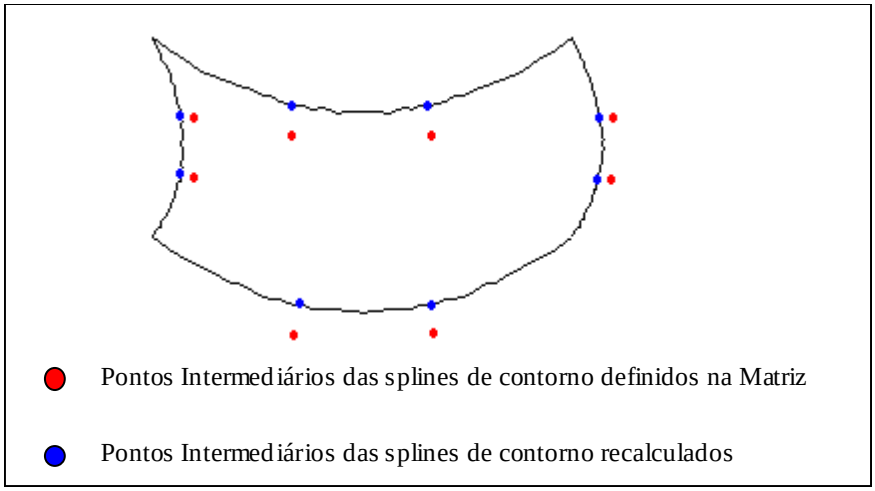
Observou-se pela equação de *Bézier* para superfícies, que os pontos inicial e final das *splines* internas formadas pelos pontos de controles da segunda linha, terceira linha, segunda coluna e terceira coluna da matriz, figura 19, não iriam fazer parte das *splines* cujos pontos de controle correspondem à primeira linha, quarta linha, primeira coluna e quarta coluna da matriz, caso a superfície fosse formada por *splines* curvas. Isto acontece devido a suavização da curva nos pontos de controle intermediários.

Figura 18 – Superfície gerada pela fórmula de *Bézier*.



A solução para esta observação foi encontrada gerando primeiramente o contorno da superfície, *splines* externas. E para cada *spline* gerada, calcular e guardar as coordenadas equivalentes aos pontos de controles intermediários, ver figura 20.

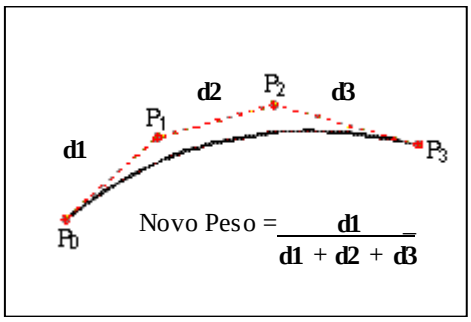
Figura 19 – Contorno da Superfície com pontos de controle.



O cálculo foi feito a partir de duas hipóteses: a primeira foi baseada na obtenção de um ponto pertencente à *spline* equivalente ao ponto de controle e a segunda na teoria de que cada ponto de controle influencia um ponto específico da *spline*.

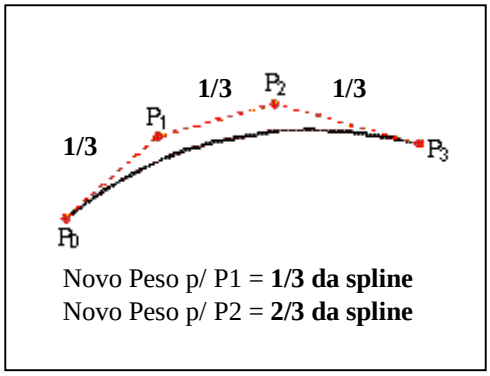
A primeira hipótese foi estudada através do cálculo da proporção das distâncias entre os pontos de controle, desta forma, acha-se um ponto (coordenada) pertencente à *spline*, dividindo a distância do primeiro ponto de controle até o ponto de controle que se quer achar a proporção, com a somatória das distâncias entre os quatro pontos de controle. Na figura 21 tem-se um exemplo do cálculo referente ao ponto P1.

Figura 20 – Representação das distâncias entre os pontos de controles.



A outra hipótese é obtida através da segmentação do peso da *spline* em três partes, onde, o ponto de controle P1 corresponde a um terço do peso da *spline*, assim como o ponto de controle P2 corresponde a dois terços do peso da *spline*, figura 22. Estes valores constantes, peso1 = 1/3 e peso2 = 2/3, são aplicados à fórmula de *Bézier* e se obtém as novas coordenadas.

Figura 21 – Pesos da *Spline* segmentada em três partes.



Observou-se que a solução pela segmentação dos pesos das *splines* em três, ocorria uma pequena distorção de visualização das *splines* pertencentes à superfície. A figura 23 exemplifica a situação. A superfície reta, sem curvatura, e plana foi submetida a uma situação em que duas *splines* verticais (Curva 3 e Curva 4) fossem colocadas com as mesmas coordenadas, estas não apareciam sobrepostas, e sim uma como foi determinada e a outra com um desvio nos pontos de controles inicial e final. Compare o resultado das duas hipóteses, observando a figura 23 o método da segmentação do peso e a figura 24 o método do cálculo da proporção.

Figura 22 – Hipótese da segmentação dos pesos.

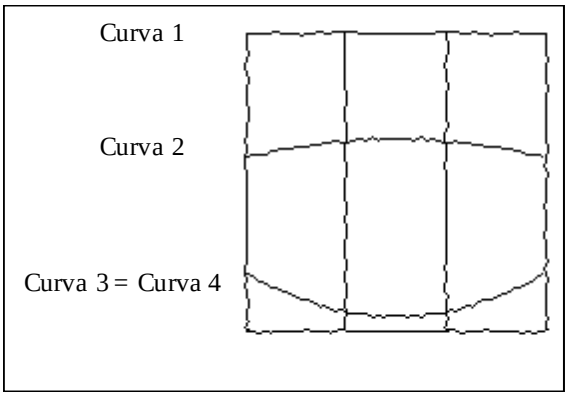
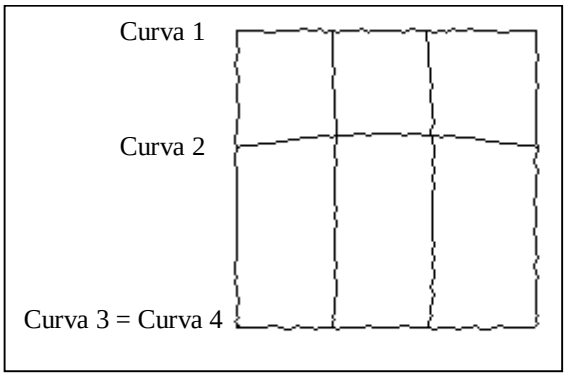


Figura 23 – Hipótese da proporção.



O protótipo oferece a opção de abrir, salvar e fechar uma superfície, assim como gerar uma nova a partir da entrada dos pontos de controle e a criação de arquivo no formato DXF para ser importado por um ambiente como AutoCad ou Corel Draw para possibilitar a visualização tridimensional com mais recursos.

Os componentes usados para salvar e abrir arquivos, `OpenDialog` e `SaveDialog`, ao serem executados, abrem uma caixa de diálogo pedindo o nome do arquivo, com sua respectiva extensão, o diretório onde se encontra o arquivo e espera uma confirmação ou cancelamento do processo.

O ambiente Delphi é baseado no desenho sobre o formulário, ou seja, se algum processo realizado altera o layout do formulário, o usuário do Delphi deve se preocupar em atualizar a tela novamente. Foi o que ocorreu com os componentes `OpenDialog` e `SaveDialog`, ao confirmar/cancelar o processo desses componentes, o Delphi manteve a sombra das caixas por cima do layout.

Para redesenhar a tela, foi usado o componente `Timer`, pois se a atualização fosse realizada no final do evento de abrir e salvar arquivo, a atualização ocorreria antes do Delphi desenhar a sombra da caixa.

O componente `Timer` é disparado quando habilitado e executa suas linhas de código depois do tempo determinado na propriedade `Interval`. Neste caso, é executado depois de 50 milisegundos, este tempo foi determinado para que o Delphi desenhe a sombra e a atualização seja imperceptível pelo usuário. A `procedure Timer` possui a chamada de duas `procedures` para atualização da tela. Veja o Quadro 2.

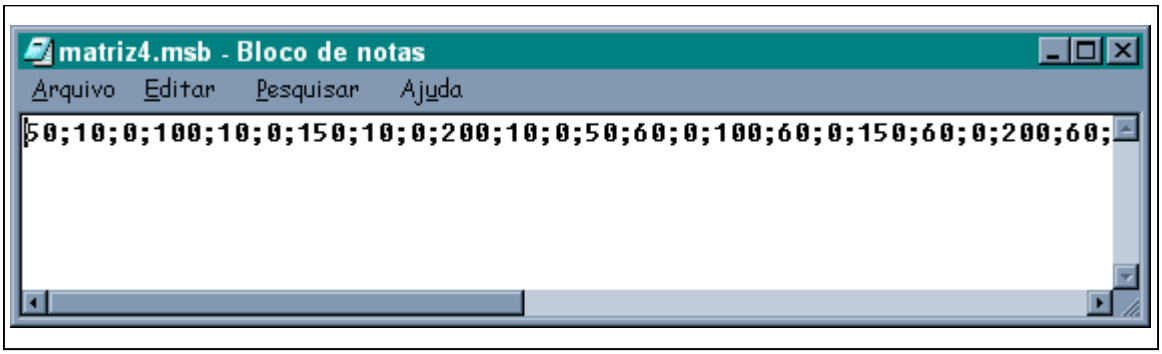
Quadro 2 – `Procedure Timer`

```
Procedure TFBezier.Timer1Timer(Sender: Tobject);
Begin
  Timer1.Enabled := false;
  LimpaTela;
  DesenhaSuperficie;
  DesenhaTitulos;
End;
```

A função Salvar e Criar Arquivo DXF geram dois arquivos textos contendo os pontos de controle da superfície e a estrutura para representação do objeto o qual poderá ser lido pelo AutoCad, respectivamente.

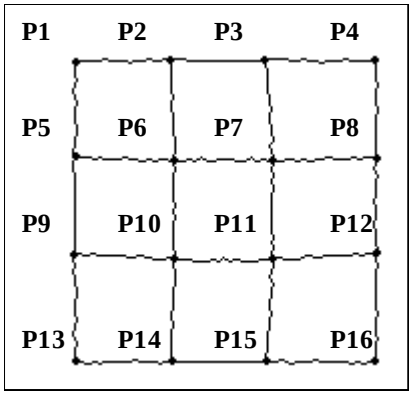
O arquivo texto contendo as coordenadas dos pontos de controle pode ser observado na figura 25.

Figura 24 - Arquivo Texto que contém as coordenadas dos pontos de controle.



O arquivo foi montado de tal forma, que os valores das coordenadas dos pontos de controle, de cada *spline* que compõem a superfície, ficassem dispostos sequencialmente e separados por ponto e vírgula. A cada grupo de três coordenadas equivale a um ponto de controle. A matriz 4x4 foi lida na direção horizontal começando pela primeira linha. Para melhor entendimento exemplificou-se uma associação entre as coordenadas do arquivo texto (figura 25) com os pontos de controle de uma superfície plana e reta, ilustrada na figura 26.

Figura 25 – Superfície com destaque os pontos de controle.



Assim, o ponto de controle **P1** (figura 26) possui coordenadas $x = 50$, $y = 10$ e $z = 0$ (figura 25); o ponto de controle **P2** possui coordenadas $x = 100$, $y = 10$ e $z = 0$, e desta forma sucessivamente até o ponto de controle **P4**. O próximo ponto de controle lido é o **P5** que possui coordenadas $x = 50$, $y = 60$ e $z = 0$ até o ponto **P8**, ou seja o arquivo possui um padrão de leitura e disposição dos pontos.

O arquivo DXF (ver seção 4.2), foi montado a partir da observação de um outro arquivo DXF já existente, gerado pelo AutoCad, que possuia uma face tridimensional formada pela opção 3DFACE do AutoCad possibilitando assim exportar a representação da superfície através de um conjunto de 9 faces.

5.2.2 ARQUIVO QUE SALVA ESTRUTURA DA SUPERFÍCIE

O arquivo que salva a estrutura da superfície no protótipo, é um arquivo de formato texto de extensão "*.msb". Esta extensão é determinada pela propriedade `Filter` do componente `SaveDialogMatriz`.

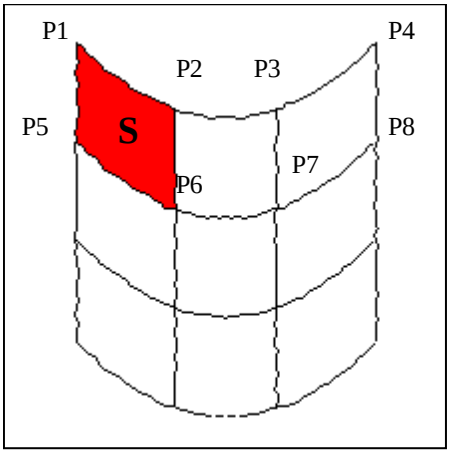
Os elementos que compõem o arquivo correspondem às coordenadas x, y e z dos pontos de controle da superfície e são separados por ";" (ponto e vírgula). Para maiores detalhes, o algortimo pode ser visto no Anexo A.

5.2.3 ARQUIVO DXF

O arquivo DXF é criado a partir do componente `SaveDialogDXF` que determina a extensão "*.dxf".

O arquivo é montado por seções. A primeira seção equivale ao cabeçalho, a segunda a topologia e dentro da topologia se encontram os valores das coordenadas correspondentes a cada face, ou seja, no protótipo visualiza-se uma superfície em aramado. A montagem do arquivo pode ser visto no Anexo B. Cada pedaço S da superfície corresponde a uma face, figura 27.

Figura 26 – Representação de cada pedaço da superfície que corresponde a uma face.



A figura 28 e 29 mostram o cabeçalho e a topologia do arquivo DXF, respectivamente.

Figura 27 – Cabeçalho do arquivo DXF.

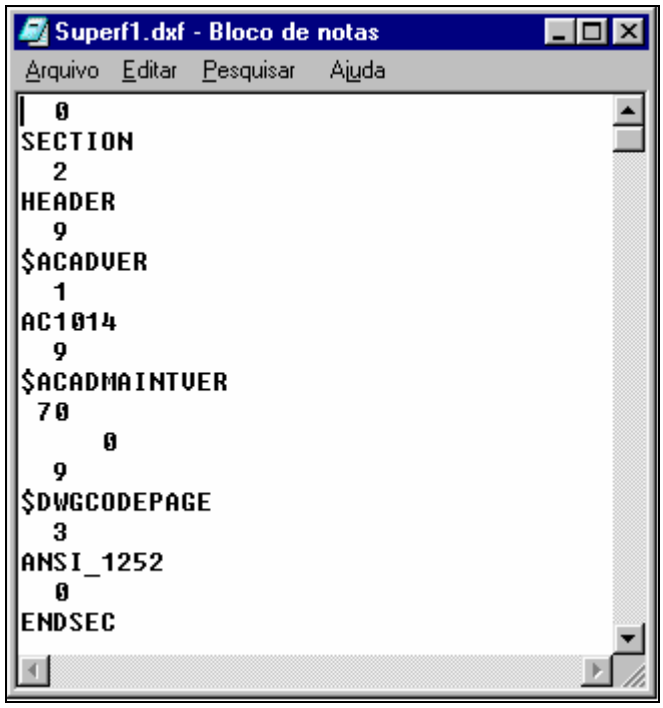
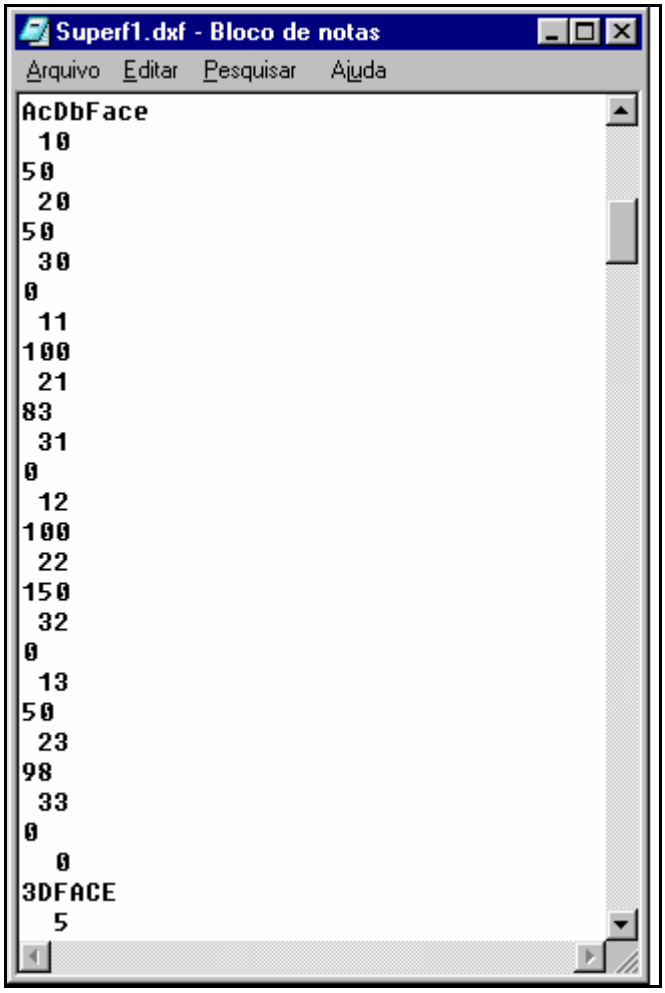


Figura 28 - Topologia do arquivo DXF.



Associando-se a figura 27 e 29 pode-se visualizar as coordenadas de cada ponto se encontram dentro do arquivo. O arquivo DXF representa a coordenada x, y e z pelas dezenas 10, 20, 30. Para representar mais de uma coordenada soma-se uma unidade a essas dezenas. Desta forma P6 (figura 27) possui as coordenadas (figura 29) x = 50, y = 50 e z = 0; P5 com x = 100, y = 83 e z = 0; P1 com x = 100, y = 150 e z = 0 e P2 com x = 50, y = 98 e z = 0.

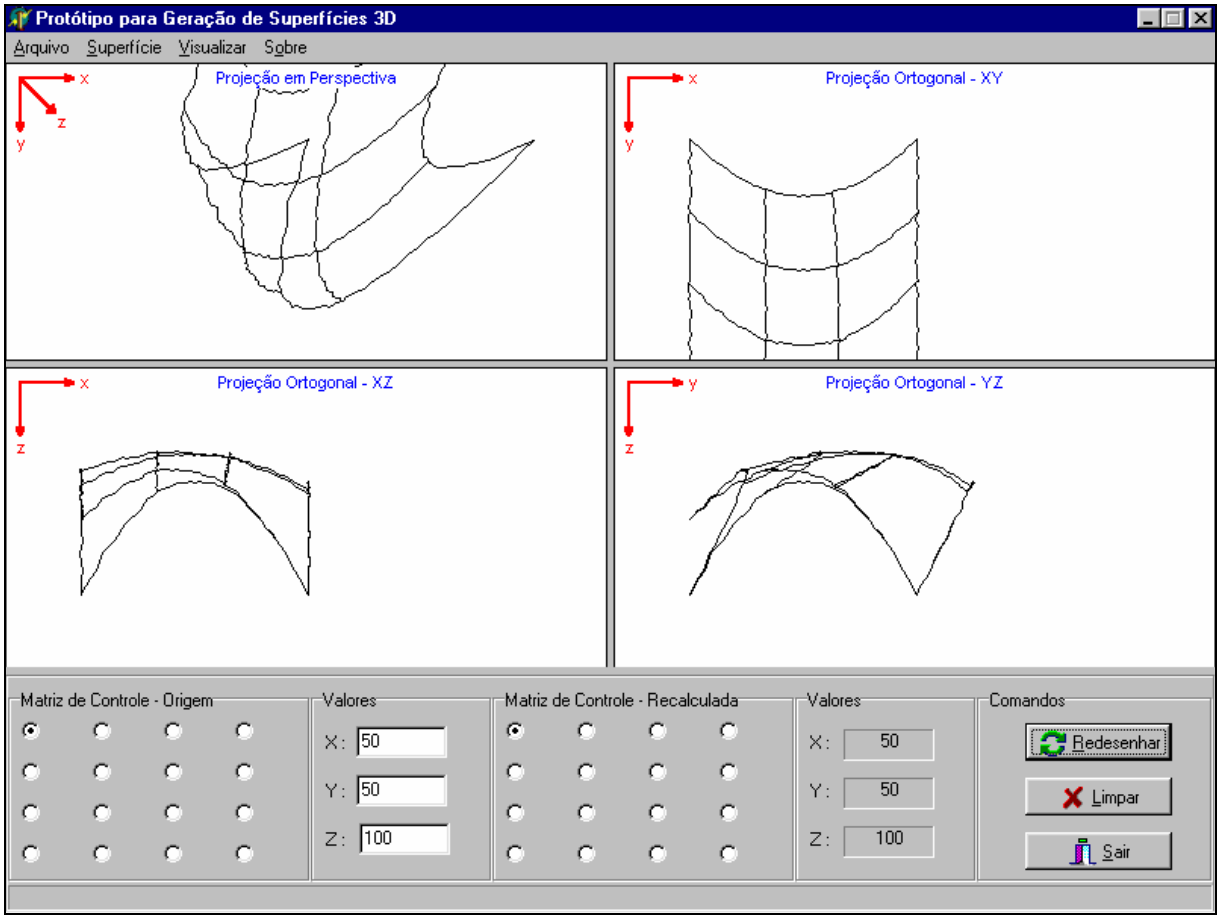
5.3 FUNCIONAMENTO DO PROTÓTIPO

A seção explica todas as funções permitidas pelo protótipo para a criação, manipulação, salvamento e geração de arquivos para serem lidos por outro ambiente.

Ao executar o protótipo, encontra-se o layout geral contendo o menu principal, quatro seções destinadas as projeções, botões de atalho equivalentes a algumas opções do menu, a

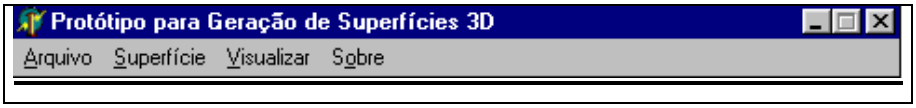
Matriz de Controle com os pontos de controle correspondentes, os valores das coordenadas de cada ponto de controle e a opção de recálculo da superfície (ver Figura 30).

Figura 29 – Layout geral.



No menu principal encontram-se quatro submenus: o submenu Arquivo, o submenu Superfície, o submenu Visualizar e o submenu Sobre, veja figura 31.

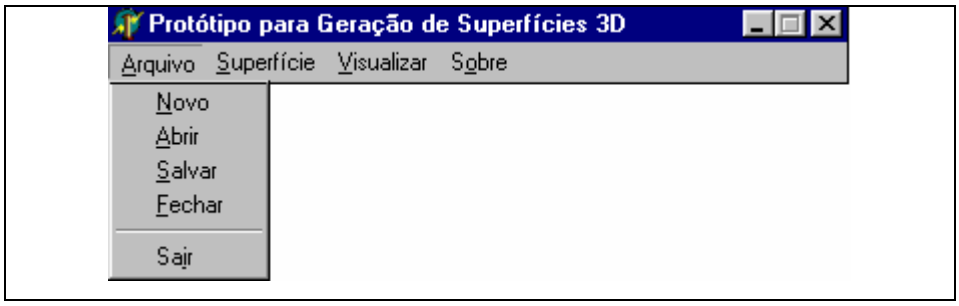
Figura 30 – Menu Principal.



5.3.1 SUBMENU ARQUIVO

O submenu arquivo possui as opções **Novo**, **Abrir**, **Salvar**, **Fechar** e **Sair**, mostradas na figura 32. Estas opções geralmente ficam neste submenu, segundo o padrão Windows.

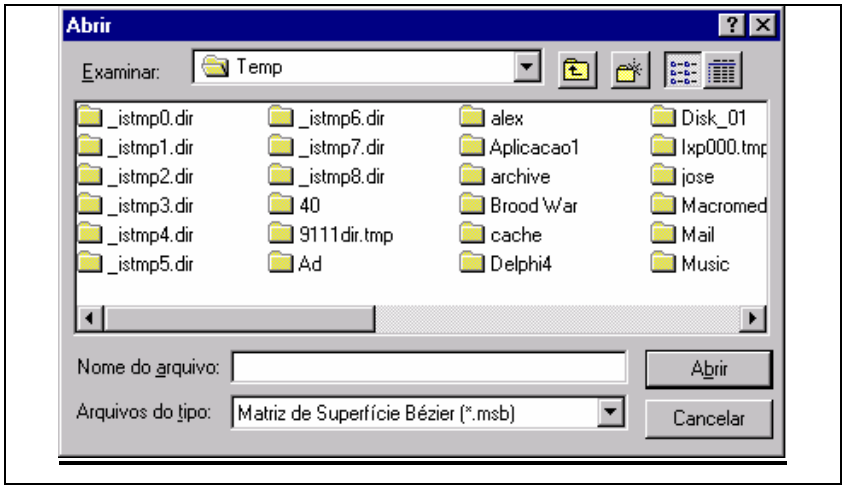
Figura 31 – Submenu Arquivo.



A opção **Novo** permite criar um novo arquivo, permitindo assim a entrada de valores para as coordenadas dos pontos de controle.

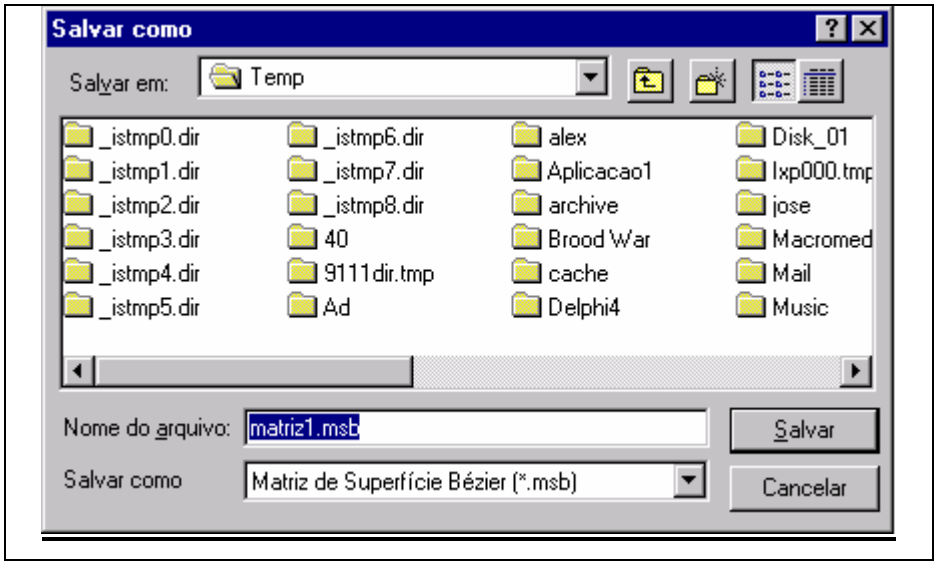
A opção **Abrir** permite abrir um arquivo de extensão ".msb" como foi explicado na seção 5.2.1. Ao executar esta opção, uma caixa de diálogo é apresentada para a localização do arquivo, ver figura 33.

Figura 32 – Caixa de diálogo da Opção Abrir.



A opção **Salvar** salva um arquivo de extensão ".msb" contendo as coordenadas da superfície, ou seja, esta opção salva a superfície propriamente dita (figura 34). Esta opção mostra uma caixa diálogo parecida com a da opção **Abrir** para a escolha do diretório onde se quer salvar.

Figura 33 – Caixa de diálogo da opção Salvar.

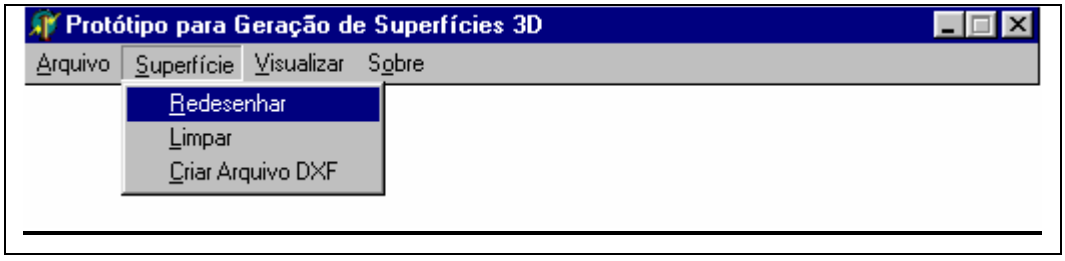


A opção **Fechar**, mostra uma mensagem perguntando se o usuário confirma ou não salvamento, caso a superfície tenha sido alterada. Sua função principal é fechar a estrutura da superfície para que outra seja mostrada. E por último a opção **Sair** que finaliza o protótipo.

5.3.2 SUBMENU SUPERFÍCIE

No submenu Superfície possui as seguintes opções: **Redesenhar**, **Limpar**, **Criar Arquivo DXF**, como mostra a figura 35.

Figura 34 – Submenu Superfície.



A opção **Redesenhar** é utilizada quando alguma coordenada de um ponto de controle é modificado, ou quando entra-se com novos pontos de controle, compare as superfícies observando as figuras 36 e 37. Esta opção limpa as projeções e executa o algoritmo que

desenha a superfície de *Bézier*, além de salvar o valor da coordenada modificada, segundo Anexo C e D e redesenha a superfície na seção tridimensional e nas seções bidimensionais.

Figura 35 – Superfície antes de ser alterada.

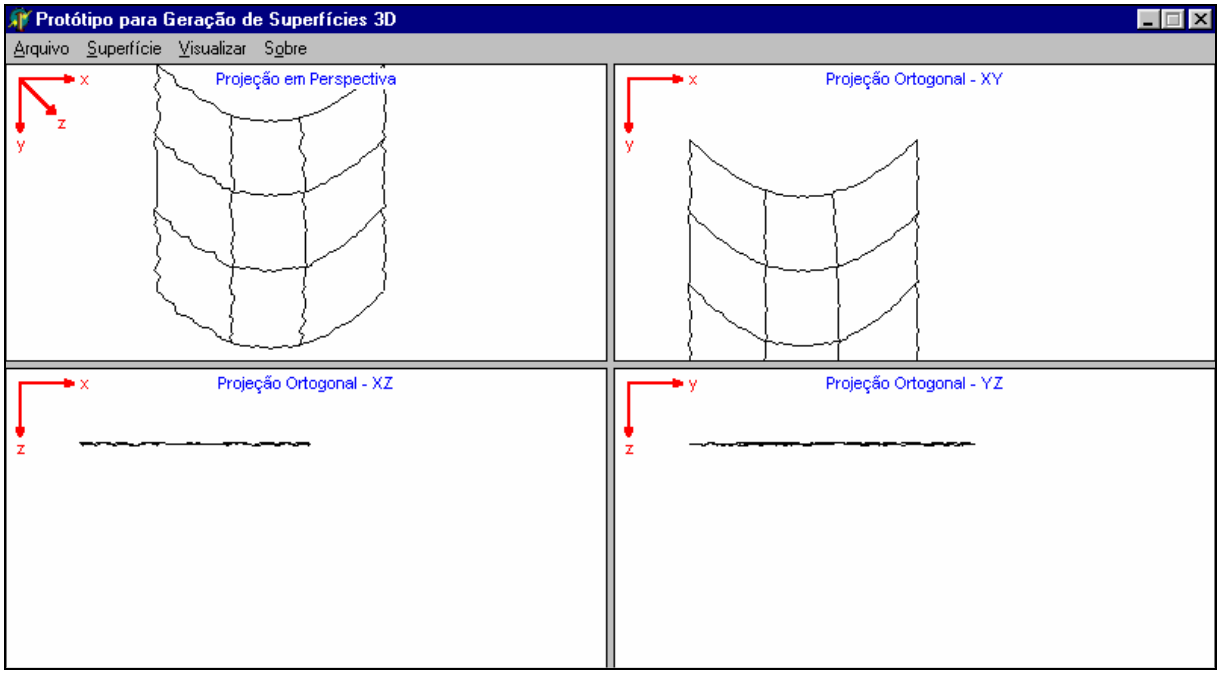
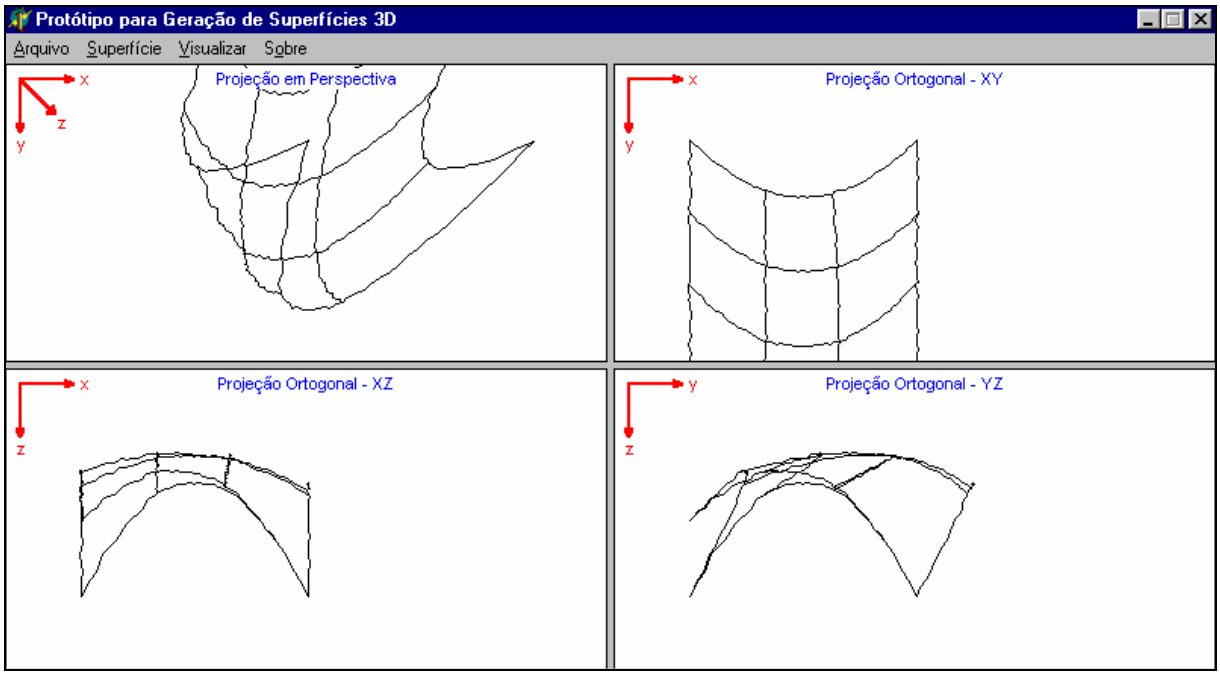


Figura 36 – Superfície depois de alterar algumas de suas coordenadas.

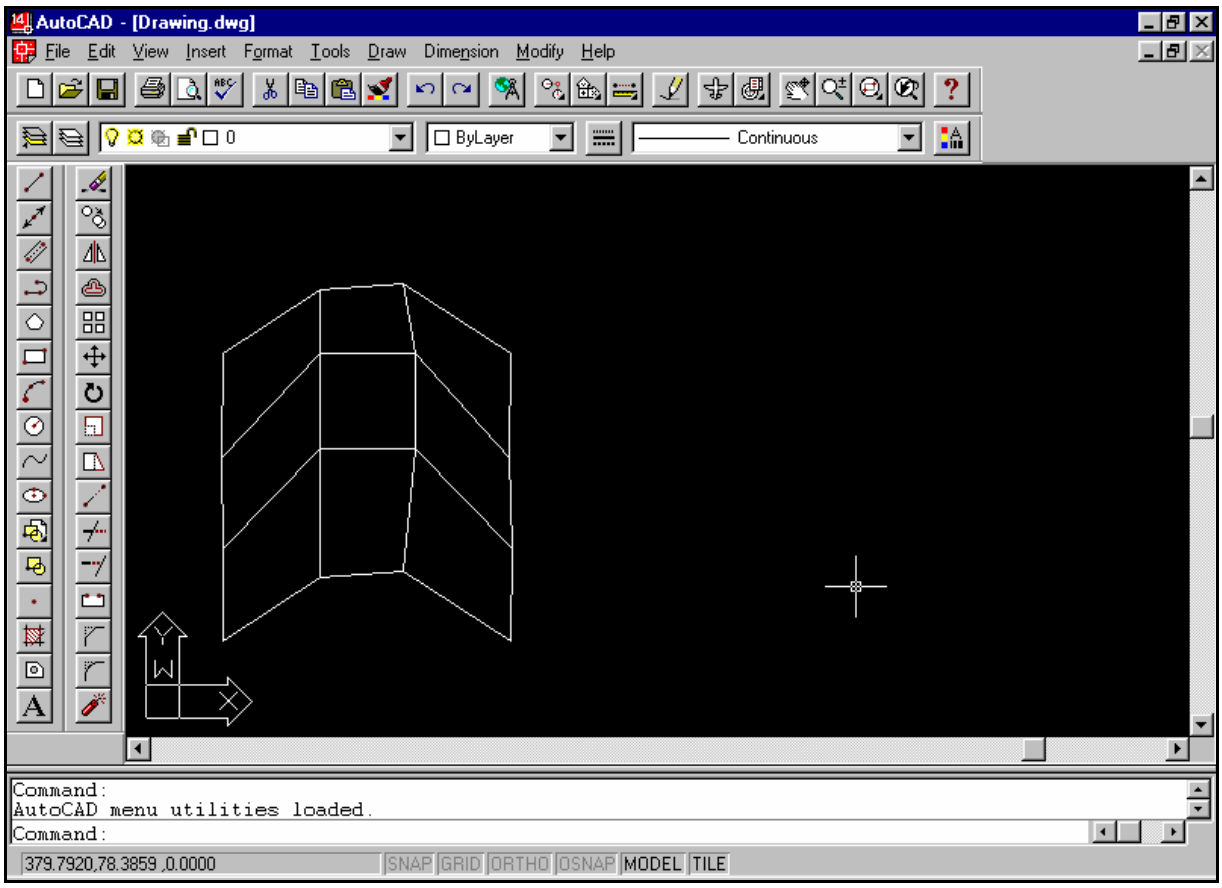


A opção **Limpar** apenas limpa todas as projeções, ou seja, se alguma coordenada for alterada e esta opção for executada apenas as projeções da superfície antiga são apagadas.

Caso o usuário escolha a opção redesenhar, a nova superfície será desenhada com as alterações.

A opção **Criar Arquivo DXF**, executa o algoritmo de montagem de um arquivo com extensão “.dxf” que poderá ser lido por outro ambiente, que consiga importar este tipo de arquivo, para aumentar as opções de visualização, limitadas pelo protótipo. A seguir a figura 38 mostra a superfície lida pelo Autocad, importando o arquivo .DXF gerada pelo protótipo.

Figura 37 – Arquivo DXF importada pelo AutoCad.

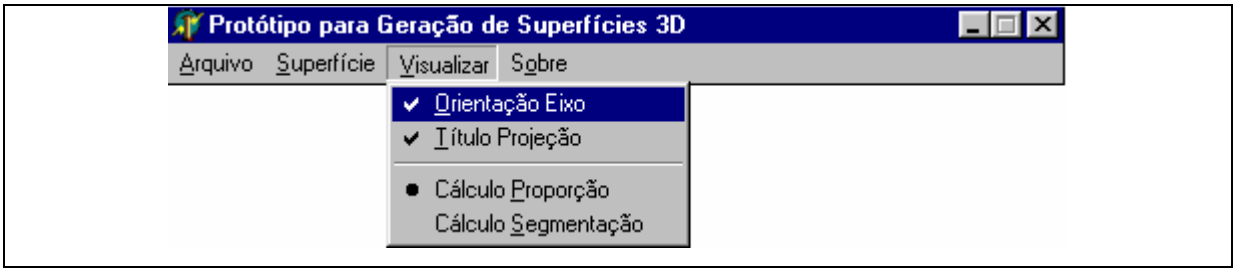


A figura 38 está invertida em relação ao protótipo, pois como mostra a figura, a coordenada *y* cresce para cima. Percebe-se também que a curvatura da superfície é menor do que na criação do arquivo DXF apenas são informados os pontos limites de cada face, ver seção 5.2.3.

5.3.3 SUBMENU VISUALIZAR

O submenu Visualizar possui as seguintes opções: **Orientação Eixo**, **Título Projeção**, **Cálculo Proporção**, **Cálculo Segmentação**, como mostra a figura 39.

Figura 38 - Submenu Visualizar



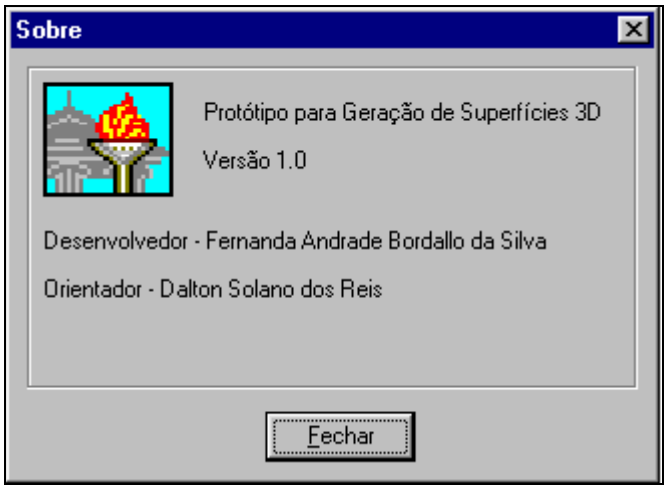
As opções Orientação Eixo e Título Projeção habilitam e desabilitam o sistema de referência usado pelo protótipo e o tipo de projeção cada imagem está usando, respectivamente.

As opções Cálculo Proporção e Cálculo Segmentação desenham a superfície de acordo com as hipóteses de recálculo da superfície explicadas na seção 5.2.1.

5.3.4 SUBMENU SOBRE

Este submenu mostra uma janela com informações gerais sobre o desenvolvimento do protótipo, ver figura 40.

Figura 39 – Submenu Sobre



6 CONCLUSÕES

A pesquisa dos conceitos acima viabilizou a geração de superfícies em três dimensões, que foram construídas por *splines Bézier* formando uma malha, através da implementação de um protótipo de software.

O ambiente de desenvolvimento Delphi mostrou-se eficiente no que diz respeito a oferecer recursos para a visualização de desenhos, em duas e três dimensões, e de possuir opções para geração de *splines* utilizando esta técnica.

Desta forma o trabalho presente obteve êxito no seu propósito, levando à aprendizagem do ambiente de desenvolvimento, da técnica para a geração de superfícies tridimensionais e tipos de projeções existentes.

O levantamento das duas hipóteses para cálculo de fechamento da borda da malha de *Bézier* levou a melhor compreensão da teoria de construção da superfície, pois requeriu uma lógica que correspondesse ao assunto proposto, fazendo também com que o resultado obtido fosse satisfatório.

A seguir apresentam-se as dificuldades encontradas no decorrer do trabalho e sugestões para trabalhos futuros.

6.1 LIMITAÇÕES

Ao gerar a superfície tridimensional, é perceptível o *aliasing* (distorção no desenho). Isto deve-se ao fato de que os componente de desenho do Delphi apenas aceitam números inteiros para serem lidos como coordenadas, devido ao fato de que as coordenadas do Delphi correspondem a pixels do computador. Desta forma o cálculo feito para geração das splines que retornam números com casas decimais, obriga o uso de comandos como `round` ou `trunc`.

Outro obstáculo encontrado foi a escassez de documentação sobre o cálculos das coordenadas dos pontos de controles intermediários, para esta técnica, com o intuito de se obter uma malha fechada, já que apenas aplicando-se a fórmula de *Bézier* esta retornaria uma malha aberta.

6.2 EXTENSÕES

Pode-se aplicar um algoritmo de *antialiasing* no algoritmo de geração da superfície, ou conseguir informações com a Borland, de como o fazer para tirar estas distorções (*aliasing*), já que o seu método de desenhar curvas *Bézier* não possui tais deformidades.

Trabalhos futuros podem estender este trabalho seguindo algumas sugestões como: explorar melhores técnicas de visualização, como as utilizadas pelo Autocad; utilizar outros tipos de técnicas para geração de superfícies, como por exemplo *Casteljau*, *Hermit*, *B-Spline* e usar os cálculos, levantados neste trabalho, para o fechamento da borda da malha da superfície.

Para melhor visualização da superfície pelo ambiente 3D, que importa o arquivo DXF, pode-se subdividir a superfície em mais pedaços de quatro pontos.

ANEXO A: ALGORITMO QUE SALVA A MATRIZ

```
Procedure TFBezier.Salvar1Click(Sender: TObject);
Var
   Arquivo: TextFile;
   linha, coluna : byte;
begin
  if SaveDialogMatriz.Execute then
  begin
     if (not FileExists(SaveDialogMatriz.FileName))
     or (MessageDlg('Arquivo já existente !!!'+#13+#10+'Substituir ?',
                     mtConfirmation,mbYesNoCancel,0) = mrYes ) Then
     begin
       try
          AssignFile(Arquivo, SaveDialogMatriz.FileName);
          rewrite(Arquivo);
          // Salva as coordenadas da matriz no arquivo
          for linha := 1 to 4 Do
             for coluna := 1 to 4 Do
                 write(arquivo,InttoStr(MOrigem[linha,coluna].x)+';'+
                               InttoStr(MOrigem[linha,coluna].y)+';'+
                               InttoStr(MOrigem[linha,coluna].z)+';');
          CloseFile(Arquivo);
       Except
          ShowMessage('Não foi possível salvar o arquivo');
       end;
     end;
  end;
  Timer1.Enabled := true;
end;
```

ANEXO B: MONTA ARQUIVO DXF

```
Procedure TFBezier.CriarArqDxf1Click(Sender: TObject);
var Arquivo :TextFile;

    procedure MontaCabecalho;
    const l = #13+#10;
    begin
        writeln(arquivo,'  0');
        writeln(arquivo,'SECTION'+l+'  2');
        writeln(arquivo,'HEADER'+l+'  9');
        writeln(arquivo,'$ACADVER'+l+'  1');
        writeln(arquivo,'AC1014'+l+'  9');
        writeln(arquivo,'$ACADMAINTVER'+ l + ' 70');
        writeln(arquivo,'     0'+l+'  9');
        writeln(arquivo,'$DWGCODEPAGE'+l+'  3');
        writeln(arquivo,'ANSI_1252'+l+'  0');
        writeln(arquivo,'ENDSEC');
    end;

    procedure MontaCorpo;
    const l = #13+#10;
    var indeixo, contador, face, indface :byte;

       procedure Cabecalho3DFace;
       begin
          writeln(arquivo,'  0'+l+'3DFACE');
          writeln(arquivo,'  5'+l+'4'+IntToStr(contador));
          writeln(arquivo,'100'+l+'AcDbEntity');
          writeln(arquivo,'  8'+l+'0');
          writeln(arquivo,'100'+l+'AcDbFace');
          inc(contador);
       end;

    begin
        contador := 0;
        writeln(arquivo,'  0'+l+'SECTION');
        writeln(arquivo,'  2'+l+'ENTITIES');
        for face := 1 to 3 do
        begin
           for indface := 1 to 3 do
           begin
             indeixo := 0;
             cabecalho3DFace;
             writeln(arquivo,'
1'+IntToStr(indeixo)+l+IntToStr(MTemp[face,indface].x));
             writeln(arquivo,'
2'+IntToStr(indeixo)+l+IntToStr(MTemp[face,indface].y));
             writeln(arquivo,'
3'+IntToStr(indeixo)+l+IntToStr(MTemp[face,indface].z));
             inc(indeixo);
             writeln(arquivo,'
1'+IntToStr(indeixo)+l+IntToStr(MTemp[face,indface+1].x));
             writeln(arquivo,'
2'+IntToStr(indeixo)+l+IntToStr(MTemp[face,indface+1].y));
             writeln(arquivo,'
```

```
3'+IntToStr(indeixo)+l+IntToStr(MTemp[face,indface+1].z));
             inc(indeixo);
             writeln(arquivo,'
1'+IntToStr(indeixo)+l+IntToStr(MTemp[face+1,indface+1].x));
             writeln(arquivo,'
2'+IntToStr(indeixo)+l+IntToStr(MTemp[face+1,indface+1].y));
             writeln(arquivo,'
3'+IntToStr(indeixo)+l+IntToStr(MTemp[face+1,indface+1].z));
             inc(indeixo);
             writeln(arquivo,'
1'+IntToStr(indeixo)+l+IntToStr(MTemp[face+1,indface].x));
             writeln(arquivo,'
2'+IntToStr(indeixo)+l+IntToStr(MTemp[face+1,indface].y));
             writeln(arquivo,'
3'+IntToStr(indeixo)+l+IntToStr(MTemp[face+1,indface].z));
           end;
        end;
        writeln(arquivo,'  0'+l+'ENDSEC');
        writeln(arquivo,'  0'+l+'EOF');
    end;

begin
  if SaveDialogDXF.Execute then
  begin
     if (not FileExists(SaveDialogDXF.FileName))
     or (MessageDlg('Arquivo já existente !!!'+#13+#10+'Substituir ?',
                    mtConfirmation,mbYesNoCancel,0) = mrYes ) Then
     begin
       try
          AssignFile(Arquivo, SaveDialogDXF.FileName);
          rewrite(Arquivo);
          MontaCabecalho;
          MontaCorpo;
          CloseFile(Arquivo);
       Except
          ShowMessage('Não foi possível salvar o arquivo');
       end;
     end;
  end;
  Timer1.Enabled := true;
end;
```

ANEXO C : ALGORITMO DE CONTRUÇÃO DA SPLINE

```
Procedure TFBezier.GeraSpline(p1,p2,p3,p4 :TPontos;
                    IndPtoRecalculado2, IndPtoRecalculado3:Integer;
                    IdentSpline: byte);
var PtoPeso, peso, aux, k0, k1, k2, ponto2, ponto3 :real;
    p1x, p1y, p1z, p2x, p2y, p2z, p3x, p3y, p3z,p4x, p4y, p4z : real;
    Cx1, Cy1, Cz1, Cx2, Cy2, Cz2, Cx3, Cy3, Cz3, Cx4, Cy4, Cz4 : integer;
    contador : integer;
    Ponto : Tpontos;

// p1,p2,p3,p4 são os quatro pontos de controle que compõem uma spline
    procedure CalculaCoordenada;
    begin
      aux := 1 – PtoPeso;
      ponto2 := exp(PtoPeso, 2);
      ponto3 := exp(PtoPeso, 3);
      k0 := exp(aux, 3);
      k1 := exp(aux, 2);
      k2 := exp(aux, 1);

      p1x := k0 * p1.x;
      p1y := k0 * p1.y;
      p1z := k0 * p1.z;
      p2x := k1 * p2.x * 3 * PtoPeso;
      p2y := k1 * p2.y * 3 * PtoPeso;
      p2z := k1 * p2.z * 3 * PtoPeso;
      p3x := k2 * p3.x * 3 * ponto2;
      p3y := k2 * p3.y * 3 * ponto2;
      p3z := k2 * p3.z * 3 * ponto2;
      p4x := p4.x * ponto3;
      p4y := p4.y * ponto3;
      p4z := p4.z * ponto3;

      Cx1 := round(p1x);
      Cy1 := round(p1y);
      Cz1 := round(p1z);
      Cx2 := round(p2x);
      Cy2 := round(p2y);
      Cz2 := round(p2z);
      Cx3 := round(p3x);
      Cy3 := round(p3y);
      Cz3 := round(p3z);
      Cx4 := round(p4x);
      Cy4 := round(p4y);
      Cz4 := round(p4z);
      Ponto.x := Cx1 + Cx2 + Cx3 + Cx4;
      Ponto.y := Cy1 + Cy2 + Cy3 + Cy4;
      Ponto.z := Cz1 + Cz2 + Cz3 + Cz4;
    end;
```

```
begin
   // seta ferramentas para desenhar a superfície em preto
   Image.Canvas.Pen.Color:= clBlack;
   Image.Canvas.Pen.Width:= 1;
   ProjXY.Canvas.Pen.Color:= clBlack;
   ProjXY.Canvas.Pen.Width:= 1;
   ProjXZ.Canvas.Pen.Color:= clBlack;
   ProjXZ.Canvas.Pen.Width:= 1;
   ProjYZ.Canvas.Pen.Color:= clBlack;
   ProjYZ.Canvas.Pen.Width:= 1;

   PtoPeso := 0;
   Peso := 1 / NroPontos;
   contador := 0;

   while (PtoPeso <= 1) do
   begin
      inc(contador);
      CalculaCoordenada;

      if PtoPeso = 0 then
      begin
      // move o primeiro ponto para a posição correta nas 3 seções
         Image.Canvas.MoveTo3D(Ponto.x,Ponto.y,Ponto.z);
         ProjXY.Canvas.MoveTo(Ponto.x,Ponto.y);
         ProjXZ.Canvas.MoveTo(Ponto.x,Ponto.z);
         ProjYZ.Canvas.MoveTo(Ponto.y,Ponto.z);
      end;
      // desenha linha nas 3 seções
      Image.Canvas.LineTo3D(Ponto.x,Ponto.y,Ponto.z);
      ProjXY.Canvas.LineTo(Ponto.x,Ponto.y);
      ProjXZ.Canvas.LineTo(Ponto.x,Ponto.z);
      ProjYZ.Canvas.LineTo(Ponto.y,Ponto.z);

// altera pontos de controle recalculados na matriz de trabalho
      if (contador = IndPtoRecalculado2) then
      begin
         Case IdentSpline of
            1: begin
                  MTemp[1,2]:= Ponto;
               end;
            2: begin
                  MTemp[4,2]:= Ponto;
               end;
            3: begin
                  MTemp[2,1]:= Ponto;
               end;
            4: begin
                  MTemp[2,4]:= Ponto;
               end;
         end;
      end
      else if (contador = IndPtoRecalculado3) then
      begin
         Case IdentSpline of
            1: begin
                  MTemp[1,3]:= Ponto;
               end;
            2: begin
```

```
                  MTemp[4,3]:= Ponto;
               end;
            3: begin
                  MTemp[3,1]:= Ponto;
               end;
            4: begin
                  MTemp[3,4]:= Ponto;
               end;
         end;
      end;
      PtoPeso := PtoPeso + peso;
   end;
   Image.Canvas.LineTo3D(P4.x,P4.y,P4.z);
   ProjXY.Canvas.LineTo(P4.x,P4.y);
   ProjXZ.Canvas.LineTo(P4.x,P4.z);
   ProjYZ.Canvas.LineTo(P4.y,P4.z);
End;
```

ANEXO D: ALGORITMO PARA RECALCULAR DOS PONTOS DE CONTROLE

```
      procedure  TfBezier.RecalculaPonto(var  IndPtoRecalculado2:  integer;
var IndPtoRecalculado3: Integer);
      var
        d1, d2, d3: real;
      begin

        if rgCalculo.ItemIndex = 0 Then //cálculo da proporção
        begin
           d1 := sqrt(
                    exp((PontosCntrl[2].x - PontosCntrl[1].x),2)+
                    exp((PontosCntrl[2].y - PontosCntrl[1].y),2) );
           d2 := sqrt(
                    exp((PontosCntrl[3].x - PontosCntrl[2].x),2)+
                    exp((PontosCntrl[3].y - PontosCntrl[2].y),2) );
           d3 := sqrt(
                    exp((PontosCntrl[4].x - PontosCntrl[3].x),2)+
                    exp((PontosCntrl[4].y - PontosCntrl[3].y),2) );
           IndPtoRecalculado2 := round(d1 / (d1+d2+d3) * NroPontos);
           IndPtoRecalculado3 := round((d1+d2) / (d1+d2+d3) * NroPontos);
        End
        else  //cálculo da segmentação
        begin
           IndPtoRecalculado2:= NroPontos Div 3;
           IndPtoRecalculado3:= (2 * NroPontos) Div 3;
        end;

end;
```

REFERÊNCIAS BIBLIOGRÁFICAS

[BAR1992] BARTELS, Richard H. and Beatty, John C.. ***An Introduction to splines for use in computer graphics and geometric modelling***. California : Morgan Kaufmann Publishers, Inc., 1992.

[BOR1989] BORGES, José Antônio dos Santos. **Introdução às técnicas de computação gráfica 3D.** Rio de Janeiro : Núcleo de Computação Eletrônica, 1989.

[BRO1995] BROWN, C. Wayne; SHEPHERD, Barry J.. ***Graphics file formats: reference and guide***. Greenwich : Manning Publications, 1995. p. 314

[CAN1997] CANTÚ, Marco. **Dominando o Delphi 3 "a bíblia"**. São Paulo : MAKRON *Books*, 1997.

[COR1994] CORRIGAN, John. **Computação gráfica: segredos e soluções**. Rio de Janeiro : Ciência Moderna, 1994.

[FOL1990] FOLEY, James D.; DAM, Andreis Van; FEINER, Steven K.; HUGHES, John F. ***Computer graphics: priciples and practice***. 2 ed.- [S.l.] : Addison Wesley, 1990.

[LEA1998] LEÃO, Marcelo. **Borland Delphi 4 curso completo**. Rio de Janeiro : Axcel Books, 1998.

[HILL1990] HILL, Francis S.. ***Computer graphics***. New York : Macmillan Publishing Company, 1990.

[MEL1990] MELENDEZ FILHO, Rubem. **Prototipação de sistemas de informação**: fundamentos, técnicas e metodologias**.** Rio de Janeiro : Livros Técnicos e Científicos, 1990.

[MOR1985] MORTENSON, Manuel. ***Geometric modeling*.** New York : John Wiley, 1985. p. 151 – 215.

[NEW1991] NEWMAN, Willian M. and Sproull, Robert F.. ***Principles of interative computer graphics.*** New York : McGraw-Hill Book Company, 1991.

[PER1989] PERSIANO, Ronaldo César Marinho; OLIVEIRA, Antônio Alberto Fernandes de. **Introdução à computação gráfica**. Rio de Janeiro : Livros Técnicos e Científicos Editora, 1989. p. 162.

[REI1995] REIS, Dalton Solano dos. **GG3D, ambiente para geração de gráficos 3D.** Porto Alegre, 1995. Trabalho Individual: (Mestrado em Ciências da Computação) CPGCC, UFRGS.

[ROG1990] Rogers, David F.; ADAMS, Alan J. ***Mathematical elements for computer graphics.*** 2. Ed. USA : McGraw-Hill, 1990.

[ZIN1993] ZINDARS, Gerson Paulo. **Implementação de uma câmera sintética para visualização de objetos tridimensionais.** Blumenau, 1993. Monografia (Bacharelado em Ciências da Computação) Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau.