**UNIVERSIDADE REGIONAL DE BLUMENAU**

**CENTRO DE CIÊNCIAS EXATAS E NATURAIS**

**CURSO DE CIÊNCIAS DA COMPUTAÇÃO**

(Bacharelado)

# ESTUDO E AVALIAÇÃO DE ALGUNS MÉTODOS DE TRIANGULARIZAÇÕES DE PONTOS DISPERSOS EM UMA SUPERFÍCIE 3D

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

**LUCIANO RAITZ**

BLUMENAU, JUNHO/2001

2001/1-46

# ESTUDO E AVALIAÇÃO DE ALGUNS MÉTODOS DE TRIANGULARIZAÇÕES DE PONTOS DISPERSOS EM UMA SUPERFÍCIE 3D

**LUCIANO RAITZ**

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. Antônio Carlos Tavares

_____
Prof. Paulo Cesar Rodacki Gomes

# DEDICATÓRIA

Dedico este trabalho a minha mãe Delorme, minha avó Andolina e aos meus tios Odair e Denise, Sérgio e Dulcenéia. Que apesar da distância física estiveram presentes em todos os momentos da minha formação acadêmica.

# AGRADECIMENTOS

A toda minha família,  por sempre terem acreditado e me apoiado durante estes anos em que me dediquei a formação profissional.

Ao meu orientador, prof. Dalton Solano dos Reis, pelo auxílio para que este trabalho pudesse ser realizado com sucesso.

Aos amigos e colegas, tanto aqueles que ficaram no decorrer do curso, como aos que conseguiram chegar ao fim de mais uma etapa de nossas vidas.

# SUMÁRIO

# LISTA DE FIGURAS

# LISTA DE QUADROS

# LISTA DE TABELAS

# RESUMO

Este trabalho, através de uma interface gráfica permite, ao usuário, criar pontos aleatórios para se triangularizar e gerar a otimização destes triângulos. A superfície triangularizada pode ser vista de vários ângulos e tamanhos, pois a imagem é projetada em perspectiva numa Câmera Sintética. A plataforma de desenvolvimento e o ambiente de execução são, respectivamente, o Borland Delphi 5.0 e o Windows 98, além da biblioteca gráfica OpenGL.

# ABSTRACT

This work, through graphic interface it is allowed, to the user, to create aleatory points to triangulate and to optimize these triangles. The surface triangulateed can be seen of several angles and sizes, because the image is projected in perspective in a Synthetic Camera. The development platform and the enviroment of execution are, respectively, Borland Delphi 5.0 and Windows 98, besides the graphic library OpenGL.

# 1 INTRODUÇÃO

A grande maioria dos problemas de engenharia requer a construção prévia de um mapa topográfico que represente adequadamente a superfície do terreno, e que se constitui assim num ponto de apoio fundamental para o projeto e execução das obras (Paredes, 1994).

O objetivo da representação do relevo topográfico é registrar e permitir a visualização da forma do terreno, e ao mesmo tempo fornecer os dados numéricos necessários para a solução dos problemas da engenharia: distâncias, e principalmente cotas de pontos desejados, com a precisão suficiente (Cintra,1984).

Como a superfície do terreno é constituída por infinitos pontos, impõem-se uma escolha de valores discretos, que sejam representativos e permitam uma representação confiável. Os critérios que norteiam essa escolha devem ser técnicos e econômicos, procurando conciliar o pouco dispêndio (reduzido número de pontos) com a qualidade do mapa (mais pontos, fidelidade do terreno) (Cintra, 1984).

Através destes pontos, pode-se gerar redes triangulares que segundo Cintra (1984), fornecidos alguns pontos distribuídos de maneira aleatória, trata-se de uní-los convenientemente para formar uma rede de triângulos. A malha triangular deve ter como vértices somente os pontos fornecidos, isto é, não deve haver cruzamentos de linhas que unem dois pontos, o que daria origem a um vértice fictício. Não deve haver pontos isolados ou não totalmente conectados. Por exemplo, cada vértice de fronteira deve estar unido a pelo menos duas arestas e cada vértice interno a pelo menos três. Caso contrário seria necessário completar a triangulação.

Triangularizações tem um papel importante na teoria da aproximação, métodos de elementos finitos, análise numérica e projeto geométrico auxiliado por computador entre inúmeras aplicações, onde freqüentemente são usadas para definir um domínio particionado que permite aproximar ou modelar superfícies (Reis, 1997).

Entre as possíveis formas de se gerar os triângulos, a triangularização *Delaunay* permite obter um conjunto de triângulos mais eqüi-angulares. Permite melhorar a representação das superfícies geradas de pontos interpolados a partir dos vértices destes

triângulos, pois asseguram que cada ponto sobre a superfície está o mais próximo possível de um vértice (Spatial, 1994; Goodchild, 1995).

A cobertura convexa (*convex hull*) de um conjunto planar de pontos é um polígono convexo que contém todo o conjunto planar com a área mínima (Eddy 1977; Goodchild, 1995).

Alguns algoritmos de triangularização necessitam da cobertura convexa, já que os seus segmentos seguramente serão lados dos triângulos, ou mais, dos triângulos otimizados (triangularização *Delaunay*), como também seguramente os pontos mínimos e máximo em ambos os eixos de coordenadas pertencerão à cobertura convexa. O cálculo da cobertura convexa (particularmente no plano) também é extensivamente estudado e aplicado em reconhecimento de padrões e processamento de imagens (Preparata, 1985).

A triangularização *Delaunay* possui interessantes propriedades, tais como: ela é única entre as diversas topologias que uma triangularização pode ter, maximiza os mínimos ângulos internos de cada triângulo, os pontos de um triângulo não podem ser colineares, a cobertura convexa sempre faz parte da grade *Delaunay* e triângulos *Delaunay* não são hierárquicos (eles não podem ser agregados ou particionados) (Goodchild, 1995; Lee, 1980; Preparata, 1985; Spatial, 1994).

O estudo da cobertura convexa é necessário devido a utilização destes em alguns dos métodos de triangularizações que posteriormente deverão ser otimizados para, desta forma, se chegar a um resultado para a devida avaliação de cada um dos algoritmos estudados através do protótipo.

## 1.1 MOTIVAÇÃO

A dedicação para a realização deste trabalho, vem da necessidade de um grande número de cálculos matemáticos para a geração de Grades Irregulares Triangularizadas.

Torna-se importante o estudo de representações de superfícies aliada à ciências exatas (matemática); superfícies estas que serão representadas através de recursos computacionais.

## 1.2 OBJETIVOS

O principal objetivo do trabalho é estudar, implementar e avaliar alguns dos diversos algoritmos, que permitam a triangularização de superfícies 3D através de pontos dispersos.

Os objetivos específicos do trabalho são:

a) implementação de um algoritmo de cobertura convexa, no mínimo três de triangularização e um de otimização;
b) efetuar avaliação dos algoritmos mencionados acima.

## 1.3 RELEVÂNCIA

Segundo Reis (1997), existe um emergente crescimento da utilização de Sistemas de Informação Geográfica em diferentes áreas, especialmente em aplicações de geoprocessamento dedicadas às necessidades específicas de Modelos Numéricos de Terreno (com o uso de Grades Irregulares Triangularizadas).

Então é importante o estudo de alguns dos algorítmos para geração de superfícies 3D, pois, a triangularização é uma das principais formas de representação gráfica destas superfícies.

## 1.4 ORGANIZAÇÃO DO TEXTO

O texto a seguir organiza-se nos capítulos abaixo:

a) Capítulo 1- Introdução: o presente capítulo, descreve o trabalho de forma geral;
b) Capítulo 2 – Ambiente de Modelagem: descreve os conceitos básicos de projeções que podem ser utilizadas na visualização de objetos tridimensionais num plano. Este capítulo fornece um conhecimento introdutório para se entender Câmera Sintética, além de apresentar as transformações geométricas em imagens Tridimensionais (3D);
c) Capítulo 3 – O Processo de Triangularização: é apresentado o algoritmo de cobertura convexa, os algoritmos de triangularização e por fim possui a otimização dos triângulos;

d) Capítulo 4 – Neste capítulo descrevem-se as principais características e os princípios de funcionamento do protótipo. Descreve-se nesta etapa o funcionamento do software incluindo detalhes de implementação e a apresentação de suas telas;

e) Capítulo 5 – Considerações Finais: relata-se aqui a análise dos resultados, conclusões, as limitações do software e por fim, sugestões para futuros trabalhos.

# 2 AMBIENTE DE MODELAGEM

Orientar-se no espaço terrestre, do ponto de vista geográfico e astronômico, sempre foi uma das preocupações básicas do ser humano. Nos primórdios da humanidade, justificava-se pela necessidade de localização de alimento e abrigo. Com o passar do tempo, veio a necessidade de traçar rotas comerciais, rotas de navegação, evoluções no campo de batalha, localização de recursos no subsolo, etc. E, cada vez mais, as necessidades iam se multiplicando. É por isso que, desde o homem paleolítico, passando pelos egípcios, babilônicos, chineses, gregos, árabes, pelos navegadores europeus, até a época atual, marcada pela existência de grandes grupos econômicos e poderosos Estados, a localização dos fenômenos geográficos sempre foi, muito mais que uma curiosidade, uma verdadeira necessidade (Sene, 1999).

A necessidade de se orientar na superfície do planeta levou os homens, ao longo da história, a elaborar vários tipos de mapas, desde as rústicas representações babilônicas até as mais modernas, feita a partir da coleta de informações obtidas por sensoriamento remoto e processadas pela informática (Sene, 1999).

Com as diversas mudanças tecnológicas em todo o mundo, principalmente com a criação dos computadores, as informações começaram a ser digitalizadas, desta forma, foi possível tratar muito mais informação em menos tempo.

A automação da cartografia vem ocorrendo em todas as fases da elaboração dos documentos cartográficos, desde a coleta de informações no campo até a impressão final dos documentos. A substituição das informações analógicas pelas digitais implica em conceitos como Distância Eletrônica, Restituição Numérica, Caderneta Eletrônica de Dados, Imagens Digitais, Plano de Informação e outros, que dão origem à Cartografia Digital (Paredes, 1994).

Ressalta-se que Sistemas de Informações Geográficas (SIG) significam muito mais que uma simples codificação, armazenamento e recuperação de dados espaciais. Geralmente, estes dados representam um modelo do mundo real, que permitem realizar simulações com situações específicas, alguns dos quais não seriam possíveis no mundo real. Por isso é importante a capacidade de realidade e a capacidade de transformação do sistema. É esta

característica do SIG que o diferencia da cartografia digital e do sensoriamento remoto (Paredes, 1994).

O SIG é um sistema que subsidia o processo de observação do mundo real em atividades de definição, mensuração, classificação, enumeração. Subsidia a atuação sobre o mundo real em atividades de operação, manutenção, gerenciamento, construção, etc. Subsidia ainda, nos processos de análise do mundo real (Paredes, 1994).

## 2.1 MODELAGEM GEOMÉTRICA

Essa elaboração pode ser dividida em três etapas. A primeira é constituída pelo levantamento: o trabalho de campo em que se obtém os dados. A segunda consiste nos cálculos: tratamento e transformação adequada dos dados para que forneçam os elementos desejados. A terceira é o desenho ou representação gráfica do terreno (Cintra,1984).

Como a superfície do terreno é constituída por infinitos pontos, impõe-se uma escolha de valores discretos, que sejam representativos e permitam uma representação confiável (Cintra,1984).

Dessa forma, os pontos levantados na topografia não são amostras escolhidas ao acaso. Há uma nítida preferência por pontos notáveis: pontos mais altos (picos), mudanças de declive, linha de cumeada ou de vale, linha d'água, etc. O profissional de campo costuma escolher os pontos de tal forma que uma interpolação linear na direção do gradiente seja uma aproximação razoável (Cintra,1984).

A distribuição espacial dos pontos também não é aleatória pois os métodos de levantamento procuram cobrir toda a região com uma distribuição relativamente uniforme dos pontos sobre a superfície a ser mapeada. As estações da polinomial são cuidadosamente escolhidas, bem como os pontos detalhes, de forma a não deixar regiões com baixa densidade de pontos (Cintra,1984).

Para a representação de superfícies em computadores, realização de um Modelo Numérico do Terreno (MNT), que é a base para a solução numérica de uma série de problemas geográficos, são utilizados um conjunto de pontos significativos. Estes pontos são registrados em coordenadas x, y e z para representarem superfícies tridimensionais (3D).

Um Modelo Numérico de Terreno é uma representação matemática, tratável computacionalmente, que representa a distribuição espacial de uma determinada característica vinculada a uma superfície real (Felgueiras, 1987). Compreende a distribuição espacial de variáveis físicas tais como: topografia, aeromagnetismo, temperatura, relevo, população, vegetação, hidrografia, minerologia, tipo de solo, etc. (Câmara, 1992). Podendo-se gerar a plotagem de contornos, geração de mapas de declividade e aspecto, determinação de volume e perfis, e visualização 3D (em aramado ou superposta a imagens ou mapas temáticos) (Felgueiras, 1987).

Os dados do Modelo Numérico de Terreno podem ser combinados com outros tipos de informações digitais como mapas topográficos, imagens de satélite, cartas pontuais e dados não gráficos na forma tabular. Permite obter assim informações de caráter qualitativo e quantitativo relevantes à superfície sem a necessidade de se trabalhar diretamente com a superfície real (Reis, 1997).

As variáveis físicas de um MNT são distribuídas espacialmente, sendo que as coordenadas x e y dos pontos estão relacionadas com as posições dos pontos na superfície. E a coordenada z está relacionada com a característica da superfície que se quer modelar. Esta distribuição pode ser de formas regular ou irregular onde os pontos (variáveis físicas) possuem espaçamentos regulares (grade regular retangular ou triangular), ou irregulares (grade irregular triangularizada) (Pfeifle, 1996).

## 2.2 REPRESENTAÇÃO 3D

Segundo Persiano (1989), transformações 3D em processamento gráfico estão divididas nos dois grupos abaixo:

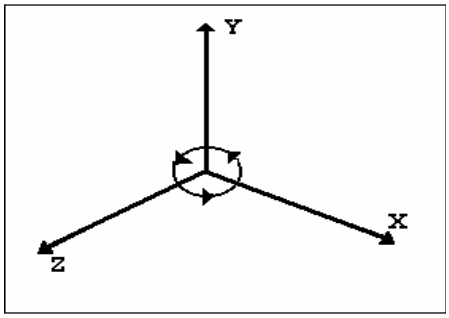
a) Transformações Geométricas. Aquelas que modificam a posição, dimensões e a forma de objetos tridimensionais: translações, rotações e escalas 3D;
b) Projeções, que visam obter representações bidimensionais de objetos 3D. Destas transformações são tratadas apenas as chamadas projeções planares, ou seja, aquelas em que a projeção é feita sobre um plano.

## 2.2.1 TRANSFORMAÇÕES GEOMÉTRICAS

A representação e visualização de um objeto tridimensional é fundamental para a percepção de sua forma. Porém, em muitas situações o objeto poderá ser movimentado através de translações, escala e rotações.

O sistema de coordenadas 3D utilizado será o da Regra da Mão Direita, com o eixo Z perpendicular ao papel e saindo em direção ao observador, como poder ser visto na FIGURA 1 (Zindars, 1993).

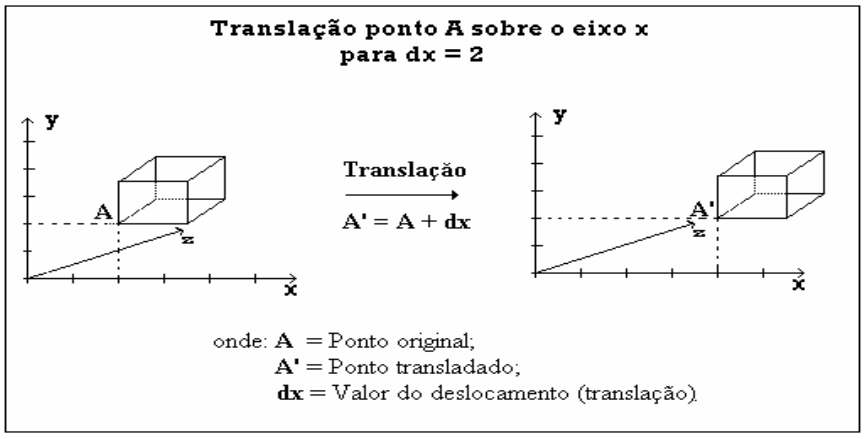
FIGURA 1 - Sistema de coordenadas dado pela regra da mão direita.




### 2.2.1.1 TRANSLAÇÃO

É a movimentação de um ponto qualquer localizado no sistema de referência do universo, para uma nova posição neste mesmo sistema. Como mostra a FIGURA 2.

FIGURA 2 - Translação do ponto A para A'.




A matriz para translação tridimensional em coordenadas homogêneas, é (Park, 1985):

$$\begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Portanto, a translação é obtida escrevendo-se:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

Multiplicando-se as matrizes, tem-se:

- x’ = x + dx;
- y’ = y + dy;
- z’ = z + dz.

Onde:

- x’, y’ e z’ = correspondem às novas coordenadas após translação;
- x, y e z = correspondem às coordenadas originais;
- dx, dy e dz = correspondem aos valores da translação.

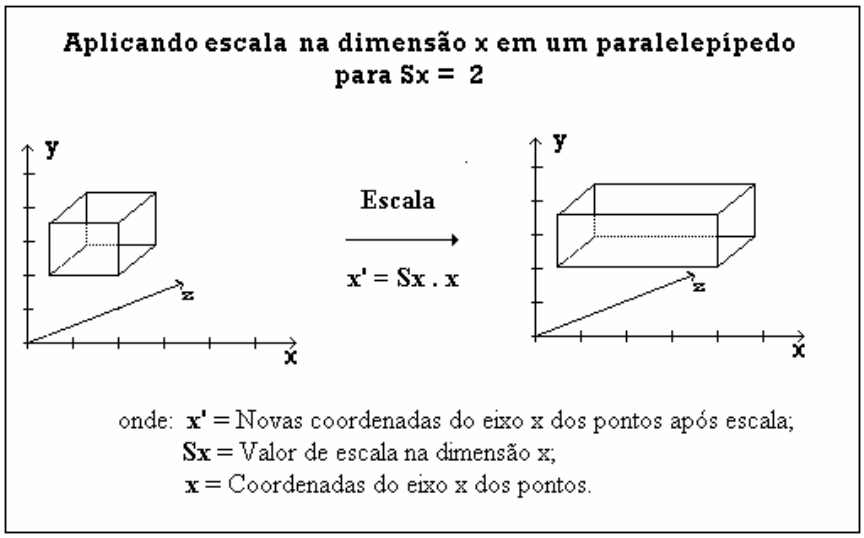
Por exemplo, para o caso apresentado na FIGURA 2, tem-se:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

#### 2.2.1.2 ESCALA

Aplica-se escala a um ponto, multiplicando-se este por um número, que representa o fator de escala. Como mostra a FIGURA 3.

FIGURA 3 - Aplicação da escala ao vértice *x* do paralelepípedo




A matriz para escala tridimensional em coordenadas homogêneas, é (Park, 1985):

$$\begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Portanto, escala é obtida escrevendo-se:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

Multiplicando-se as matrizes, tem-se que:

- x’ = x + Sx;
- y’ = y + Sy;
- z’ = z + Sz.

Onde:

- x’, y’ e z’ = correspondem às novas coordenadas após escala;
- x, y e z = correspondem às coordenadas originais;

- Sx, Sy e Sz = correspondem aos valores da escala.

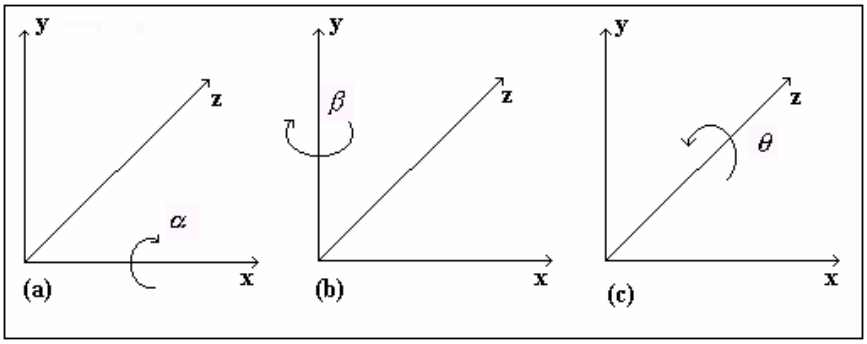
Por exemplo, para o caso apresentado na FIGURA 3, tem-se:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

### 2.2.1.3 ROTAÇÃO

A rotação de um ponto no universo pode ser realizada sobre qualquer um dos eixos do sistema de referência do universo, como pode ser observado na FIGURA 4.

FIGURA 4 - Visualização de rotação sobre os eixos *x*, *y* e *z*, conforme (a), (b) e (c)




Fonte: Newman, 1979.

Para rotacionar sobre o eixo *x*, a coordenada *x* de um vetor não muda. A rotação ocorre em plano perpendicular ao eixo x. Similarmente, as rotações sobre os eixos *y* e *z* ocorrem em planos perpendiculares aos eixos *y* e *z*, respectivamente.

As matrizes para rotação sobre cada um dos eixos são:

a) Rotação sobre o eixo *x*:

A matriz para calcular a rotação tridimensional em coordenadas homogêneas sobre o eixo x, conforme FIGURA 4 (a), é (Rogers, 1990):

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\operatorname{sen}\alpha & 0 \\ 0 & \operatorname{sen}\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Portanto, a rotação sobre o eixo *x* é obtida, escrevendo-se:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\operatorname{sen}\alpha & 0 \\ 0 & \operatorname{sen}\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

Multiplicando-se as matrizes, tem-se que:

- $x' = x;$
- $y' = y.cos\propto - z.sen\propto;$
- $z' = y.cos\propto - z.sen\propto.$

Onde:

- *x'*, *y'* e *z'*, correspondem às novas coordenadas após a rotação;
- *x*, *y* e *z*, correspondem às coordenadas após a rotação;
- $\propto$, corresponde ao ângulo de rotação.

b) Rotação sobre o eixo *y*:

A matriz para calcular a rotação tridimensional em coordenadas homogêneas sobre o eixo *y*, conforme FIGURA 4 (b), é (Rogers, 1990):

$$\begin{bmatrix} \cos\beta & 0 & \operatorname{sen}\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\operatorname{sen}\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Portanto, a rotação sobre o eixo *x* é obtida, escrevendo-se:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\beta & 0 & \operatorname{sen}\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\operatorname{sen}\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

Multiplicando-se as matrizes, tem-se que:

- $x' = x.cos\beta + z.sen\beta;$
- $y' = y;$
- $z' = -x.sen\beta + z.cos\beta.$

Onde:

- *x'*, *y'* e *z'*, correspondem às novas coordenadas após a rotação;
- *x*, *y* e *z*, correspondem às coordenadas após a rotação;
- $\beta$, corresponde ao ângulo de rotação.

c) Rotação sobre o eixo *z*:

A matriz para calcular a rotação tridimensional em coordenadas homogêneas sobre o eixo z, conforme FIGURA 4 (c), é (Rogers, 1990):

$$\begin{bmatrix} \cos\theta & -\text{sen}\,\theta & 0 & 0 \\ \text{sen}\,\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Portanto, a rotação sobre o eixo *z* é obtida, escrevendo-se:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\text{sen}\,\theta & 1 & 0 \\ \text{sen}\,\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

Multiplicando-se as matrizes, tem-se que:

- $x' = x.\cos\theta - y.\text{sen}\,\theta$;
- $y' = x.\text{sen}\,\theta + y.\cos\theta$;
- $z' = z$.

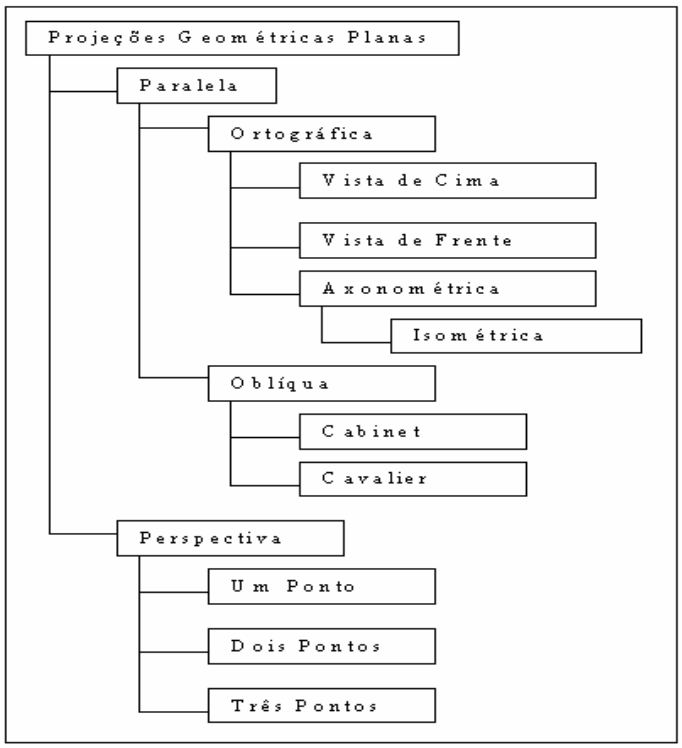
Onde:

- *x'*, *y'* e *z'*, correspondem às novas coordenadas após a rotação;
- *x*, *y* e *z*, correspondem às coordenadas após a rotação;
- $\theta$, corresponde ao ângulo de rotação.

## 2.2.2 PROJEÇÕES

Projeção é a forma de representação de figuras em três dimensões numa superfície plana, como por exemplo, a tela do computador. A projeção de figuras em 3D sobre o plano pode pertencer a duas classes distintas, paralela e a perspectiva (Borges, 1988). Estas classes estão subdivididas conforme FIGURA 5.

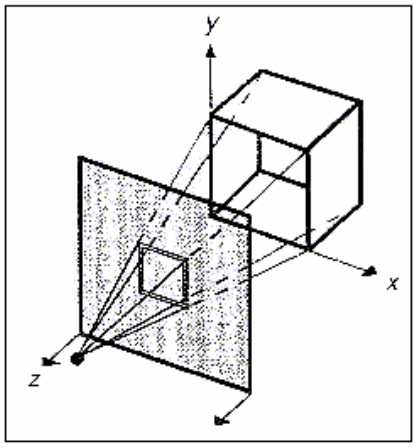
FIGURA 5 - Hierarquia das projeções




Neste trabalho foi utilizada a projeção em perspectiva com um ponto de fuga, para se representar as superfícies triangularizadas que foi facilitado com a utilização da biblioteca gráfica OpenGL (ver capítulo 3).

A Projeção em perspectiva cria um efeito semelhante a visão humana. As dimensões relativas não são preservadas. Quanto mais distante estiver um objeto do centro de projeção, menor será sua projeção perspectiva (Borges, 1988). Apenas as linhas que são paralelas ao plano de projeção serão projetadas como paralelas, mostrado na FIGURA 6.

FIGURA 6 - Projeção perspectiva do cubo partindo do ponto paralelo ao eixo z




Fonte: Foley, 1990

Esta forma de projeção é especificada definindo-se o centro de projeção, para onde convergem as linhas de projeção. Existe o ponto de fuga, onde será convergido qualquer conjunto de linhas paralelas de imagens, que não sejam paralelas ao plano de projeção. 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".

## 2.3 CÂMERA SINTÉTICA

Câmera sintética foi utilizada para uma melhor visualização dos objetos em 3D que serão representados no protótipo. A câmera sintética permite um grande número de transformações geométricas em objetos tridimensionais, visualizados no sistema de referência do universo, que serão representados em um plano (monitor), simulando desta forma a imagem feita através de uma máquina fotográfica.

A visualização de objetos tridimensionais num plano consiste em exibir os pontos dos objetos após o cálculo da projeção perspectiva dos mesmos (Zindars, 1993).

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

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 parte do cenário disponível (Zindars, 1993).

## 2.4 SUPERFÍCIES

Para a representação de uma superfície real no computador é indispensável a elaboração e criação de um modelo digital, que pode estar representado por equações analíticas ou uma rede (grade) de pontos, de modo a transmitir ao usuário as características espaciais do terreno ou qualquer outro objeto.

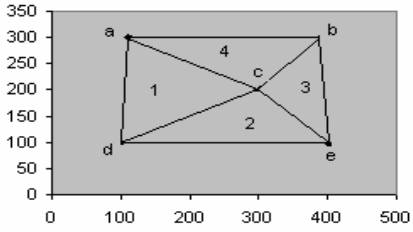
Os dados do Modelo Numérico de Terreno (MNT) podem ser combinados com outros tipos de informações digitais como mapas topográficos, imagens de satélite, cartas pontuais e dados não gráficos na forma tabular. Permite obter assim informações de caráter qualitativo e quantitativo relevantes à superfície sem a necessidade de se trabalhar diretamente com a superfície real. O Modelo Numérico de Terreno aparece referenciado na literatura de várias formas, tais como: *Digital Elevation Data* (DED), *Digital Terrain Data* (DTD), *Digital Terrain Model* (DTM), *Digital Elevation Model* (DEM), *Digital Terrain Elevation Data* (DTED) e *Triangulated Irregular Network* (TIN).

### 2.4.1 TOPOLOGIA DE ARQUIVOS

Existem duas maneiras básicas de armazenar uma grade triangularizada: triângulo por triângulo ou ponto e seus vizinhos (FIGURA 7). Em triângulo por triângulo, o registro usualmente contém: um número para referenciar o triângulo, as coordenadas $(x, y, z)$ dos três vértices e o número de referência dos três triângulos vizinhos. Em ponto e seus vizinhos, têm-se um número de identificação e as coordenadas $(x, y, z)$ para referenciar seus vértices vizinhos (ordem horário ou anti-horário). Esta foi a estrutura original dos modelos de Grades Irregulares Triangularizadas sendo uma alternativa para armazenar muitos vértices (Spatial, 1994; Goodchild, 1995).

FIGURA 7 – Topologia de Arquivos

**Topologia de Arquivos**




| Índices | Coordenadas | | |
|---|---|---|---|
| | X | Y | Z |
| a | 110 | 300 | 100 |
| b | 390 | 300 | 100 |
| c | 300 | 200 | 100 |
| d | 100 | 100 | 100 |
| e | 400 | 100 | 100 |

Triângulo por Triângulo

| Índices | Ponto | | | Ponto | | | Ponto | | | Indicação | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X | Y | Z | X | Y | Z | X | Y | Z | Vizinho | | |
| 1 | 300 | 200 | 100 | 110 | 300 | 100 | 100 | 100 | 100 | - | 2 | 4 |
| 2 | 300 | 200 | 100 | 100 | 100 | 100 | 400 | 100 | 100 | - | 3 | 1 |
| 3 | 300 | 200 | 100 | 400 | 100 | 100 | 390 | 300 | 100 | - | 4 | 2 |
| 4 | 300 | 200 | 100 | 390 | 300 | 100 | 110 | 300 | 100 | - | 1 | 3 |

Ponto e seus Vizinhos

| Índices | Ponto | | | Ponto | | | Ponto | | | Ponto | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X | Y | Z | X | Y | Z | X | Y | Z | X | Y | Z |
| a | 100 | 100 | 100 | 300 | 200 | 100 | 390 | 300 | 100 | - | - | - |
| b | 110 | 300 | 100 | 300 | 200 | 100 | 400 | 100 | 100 | - | - | - |
| c | 110 | 300 | 100 | 100 | 100 | 100 | 400 | 100 | 100 | 390 | 300 | 100 |
| d | 400 | 100 | 100 | 300 | 200 | 100 | 110 | 300 | 100 | - | - | - |
| e | 390 | 300 | 100 | 300 | 200 | 100 | 100 | 100 | 100 | - | - | - |

Ambas as estruturas são recomendáveis dependendo do propósito a que se destinam. Análise de declividade, aspecto, visualização e cálculos de volume necessitam da primeira enquanto que geração de contorno e outros procedimentos de trajetória trabalham melhor com a segunda estrutura. Pode-se converter de um formato para o outro através de uma pesquisa exaustiva ponto por ponto (Reis, 1997).

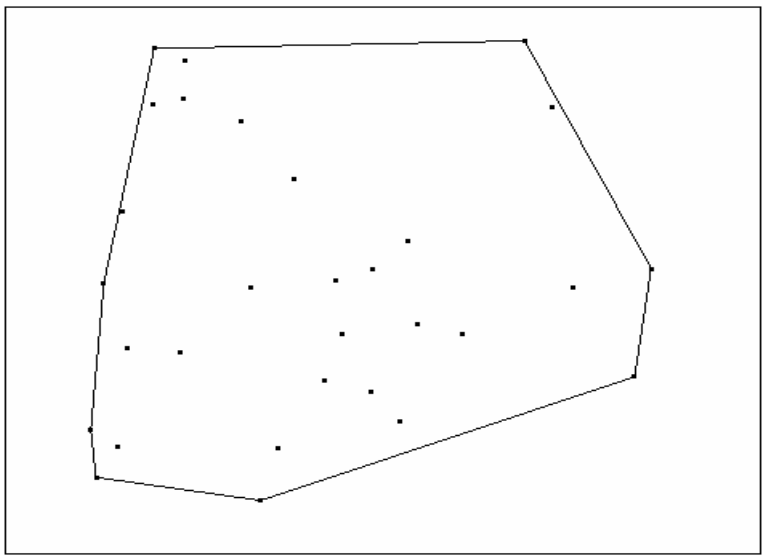
# 3 O PROCESSO DE TRIANGULARIZAÇÃO

## 3.1 COBERTURA CONVEXA

Os pontos que formam a cobertura convexa podem ser obtidos de várias maneiras, mas em qualquer conjunto de pontos distribuídos sempre haverá uma única seqüência de pontos da cobertura convexa. Esta seqüência pode ter um sentido horário ou anti-horário. No caso optou-se pelo anti-horário (Reis, 1997).

A cobertura convexa (*convex hull*) de um conjunto planar de pontos é um polígono convexo que contém todo o conjunto planar com a área mínima (Eddy 1977; Goodchild, 1995) ( FIGURA 8).

FIGURA 8 - Cobertura convexa com 30 pontos



Fonte: Reis, 1997.

Existem diversos algorítmos para geração da cobertura convexa, já que estes algorítmos são necessários em alguns métodos de triangularização. O algoritmo conhecido como Pesquisa de Cantos, foi estudado e implementado para a realização deste trabalho.

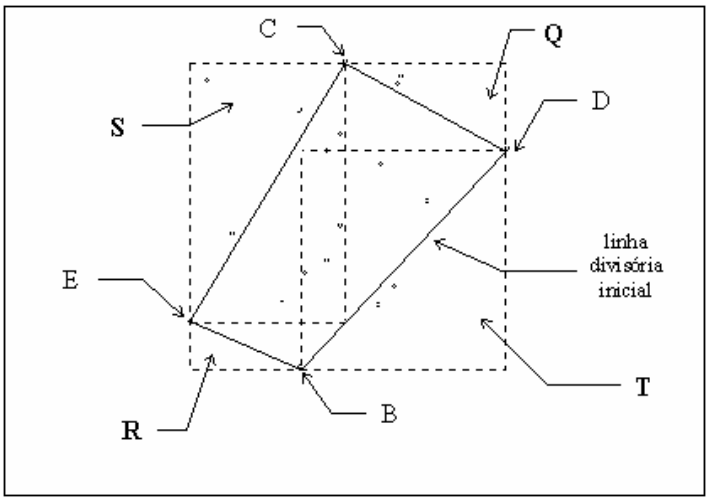
### 3.1.1 ALGORITMO DE COBERTURA CONVEXA

Segundo Reis (1997), o algoritmo de pesquisa de cantos obtém os pontos da cobertura convexa através de uma série de passos que começam com o particionamento do conjunto ***P*** constituído de n pontos $P_i$ coplanares em quatro subconjuntos ***Q, S, R*** e ***T***. Estes subconjuntos são definidos por uma pesquisa ordenada nos pontos $P_i$ do conjunto ***P*** testando suas coordenadas em relação aos pontos D, C, E e B, que estão, respectivamente, mais a direita, mais acima mais à esquerda e mais abaixo.

As definições dos quatro subconjuntos iniciais são obtidas por uma pesquisa ordenada nos pontos $P_i$ do conjunto ***P*** da seguinte forma (FIGURA 9):

a) no subconjunto ***Q***, em relação ao valor da coordenada x, $P_i$ está à direita de C, e, em relação ao valor da coordenada y, $P_i$ está acima de D;
b) no subconjunto ***S***, em relação ao valor da coordenada x, $P_i$ está à esquerda de C, e, em relação ao valor da coordenada y, $P_i$ está acima de E;
c) no subconjunto ***R***, em relação ao valor da coordenada x, $P_i$ está à esquerda de B, e, em relação ao valor da coordenada y, $P_i$ está abaixo de E;
d) no subconjunto ***T***, em relação ao valor da coordenada x, $P_i$ está à direita de B, e, em relação ao valor da coordenada y, $P_i$ está abaixo de D.

FIGURA 9 - Subconjuntos do Conjunto *P* e linhas divisórias iniciais



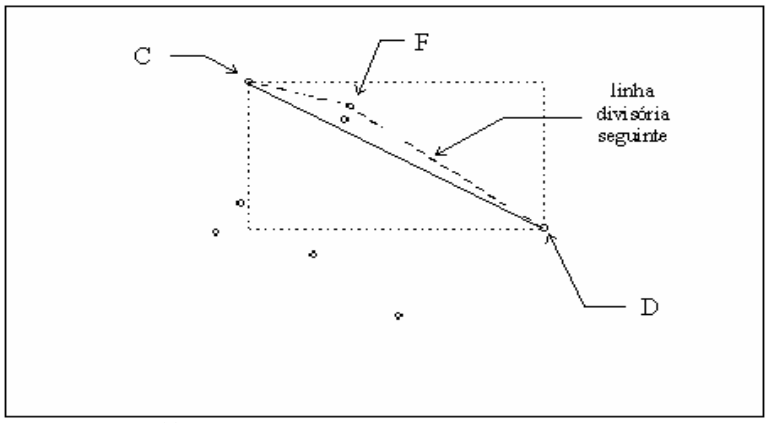

Fonte: Reis, 1997.

Após definir os quatro subconjuntos (***Q, S, R*** e ***T***) se particiona cada um destes subconjuntos de acordo com as linhas divisórias iniciais (FIGURA 9) definidas por dois pontos cada como:

a) no subconjunto ***Q***, pelos pontos D e C;
b) no subconjunto ***S***, pelos pontos C e E;
c) no subconjunto ***R***, pelos pontos E e B;
d) no subconjunto ***T***, pelos pontos B e D.

Com o particionamento deste subconjuntos, verifica-se entre seus pontos o ponto mais afastado F de uma perpendicular em relação à linha divisória inicial. Assim, no caso do subconjunto ***Q*** determina-se a linha divisória seguinte através do ponto F e um dos pontos (D e C) usados para determinar a linha divisória inicial. Esta linha divisória seguinte particiona o conjunto ***Q*** em dois subconjuntos seus, e assim sucessivamente para todos os subconjuntos (FIGURA 10).

FIGURA 10 - Detalhe da área de pesquisa




Fonte: Reis, 1997.

O término destes particionamentos sucessivos ocorrerá quando o conjunto contiver um ou nenhum ponto. O conjunto (lista) de pontos pertencentes à cobertura convexa é formado pelos quatro pontos (D, C, E e B) mais afastados entre si em relação aos eixos, e pelos pontos F mais afastados perpendicularmente em relação às linhas divisórias em cada repetição.

Um caso que pode ocorrer é quando D, C, E e B, que definem as quatro linhas divisórias iniciais, serem pontos coincidentes entre si. Neste caso não se necessita fazer o particionamento inicial do conjunto de pontos que contém estes dois pontos coincidentes como linha divisória e um destes dois pontos pertencerá à cobertura convexa.

Para definir os subconjuntos iniciais do conjunto ***P***, os pontos D, C, E e B são determinados por uma pesquisa ordenada dos seus pontos **P**. Desta forma, determinam-se quatro pontos D, C, E e B tais que $x_D = \max(x_P)$, $y_C = \min(y_P)$, $x_E = \min(x_P)$ e $y_B = \max(y_P)$ onde $x_P$ é o valor da coordenada x e $y_P$ é o valor da coordenada y, do ponto qualquer $\mathbf{P}(x_{P,}\, y_P)$ de ***P***.

A determinação se o ponto $P_i$ está à esquerda ou à direita da linha divisória é obtida, pelo cálculo do produto vetorial, obtendo-se um valor real para vereificar a posição do ponto, de acordo com o procedimento encontrado no anexo 1. O retorno da função `PosPonto` será um valor numérico, se este valor for maior que zero (0), é possível que este ponto que foi calculado pertença a cobertura convexa.

## 3.2 TRIANGULARIZAÇÃO

A triangularização dos pontos dispersos pode se dar de muitas formas levando às mais diversas combinações de triângulos, mas em qualquer um destes conjuntos têm-se sempre os mesmos números de triângulos e de lados (Lawson,1977). Na definição dos algorítmos de triangularização, convencionou-se um único sentido, o anti-horário e, o algorítmo de cobertura convexa estudado na seção 3.1 foi utilizado em um dos métodos de triangularização.

### 3.2.1 ALGORITMO DE INTERSECÇÃO

Segundo Reis (1997), a essência deste algoritmo está numa série de passos que se inicia com a geração de todas as possíveis combinações de segmentos com os elementos do conjunto ***P***, sem que estes segmentos interseccionem entre si. O conjunto ***P*** é constituído de n pontos $P_i$ coplanares.

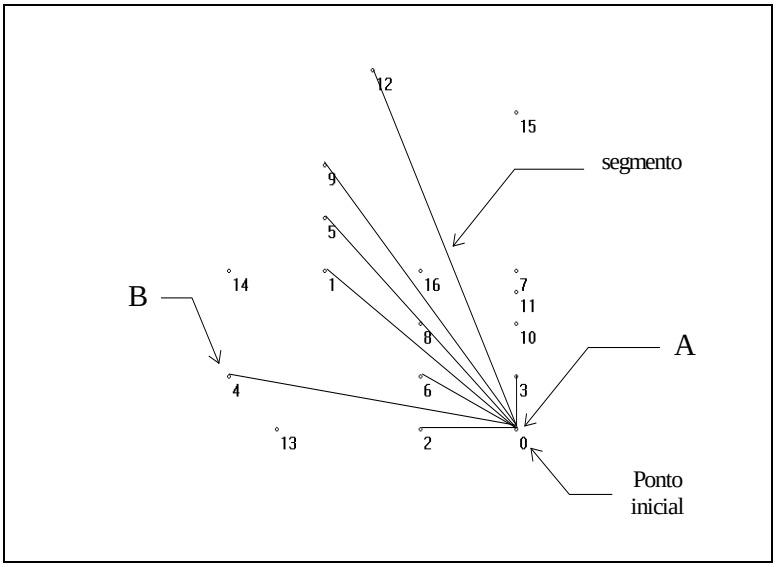
Após esta etapa, geram-se os triângulos por meio de uma pesquisa ordenada destes segmentos, verificando-se quais segmentos possuem índices de vértices em comum. Desta forma, permite-se gerar o conjunto ***T*** de combinações de índices de vértices que definem uma triangularização aos pontos $P_i$ do conjunto ***P***. O conjunto ***T*** é constituído de n combinações de índices de vértices os quais representam cada triângulo numa triangularização.

Outra informação contida no conjunto ***T*** é a indicação da vizinhança de cada triângulo, ou seja, uma combinação de índices que representa um triângulo sempre possuirá três indicações de triângulos adjacentes aos referidos segmentos. Um caso que pode ocorrer é quando um triângulo possuir um ou dois segmentos sem a definição de vizinhança, indicando que estes segmentos pertencem ao conjunto de pontos da cobertura convexa.

Para formar os segmentos da triangularização, deve-se pesquisar seqüencialmente os pontos $P_i$ do conjunto ***P*** a fim de gerar o conjunto ***S*** que contém todas as combinações de segmentos. Estes segmentos são formados pelos pares de índices A e B, onde A se constitui de todos os pontos $P_i$ do conjunto ***P*** e B são todos os pontos do conjunto ***P*** que numa iteração anterior de pesquisa ainda não tenham sido um ponto A (FIGURA 11). A cada segmento novo

de pesquisa, deve-se verificar se este não intersecciona um segmento já existente no conjunto ***S***.

FIGURA 11 - Segmentos de pesquisa

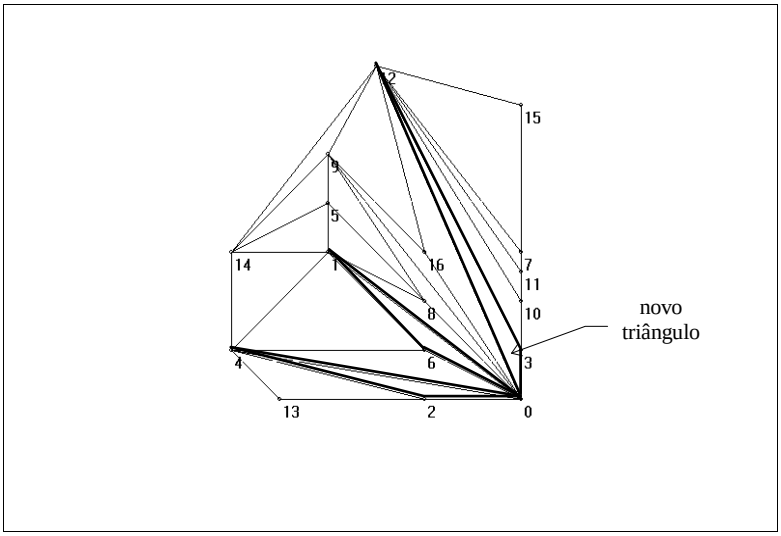



Fonte: Reis, 1997.

Com o conjunto ***S*** definido, deve-se pesquisar seqüencialmente todos os seus segmentos para gerar o conjunto ***T***, que contém combinações de índices de vértices que definem uma triangularização. Esta pesquisa é feita da seguinte forma:

a) obtém-se o primeiro segmento $\bar{AB}$ (índices A e B) (índices 0 e 1 na FIGURA 12);
b) pesquisa-se o próximo segmento $\bar{CD}$ (índices C e D) após o segmento $\bar{AB}$ (índices 1 e 6 na FIGURA 12) para encontrar todos os segmentos que tenham o índice B igual ao de C, até que B seja menor que C;
c) pesquisa-se o terceiro segmento $\bar{EF}$ para formar um triângulo, sendo este o próximo segmento após $\bar{AB}$ (índices 0 e 6 na FIGURA 12) que possua o índice D igual ao de F, até que D seja menor que F.

FIGURA 12 - Segmentos dos triângulos (lados)




Fonte: Reis, 1997.

Encontrando-se estes três segmentos ( $\overline{AB}$, $\overline{CD}$ e $\overline{EF}$ ), deve-se então testar se E é igual a A, podendo assim formar um novo triângulo (C, D e E) com estes três segmentos. Ao gerar este novo triângulo, deve-se testar se todos os pontos P do conjunto *P* (exceto os pontos C, D e E) são externos ao novo triângulo.

Para garantir que a triangularização possua um sentido anti-horário, deve-se testar a cada novo triângulo o sentido de seus vértices pelo cálculo do determinante dos seus pontos projetados num plano 2D (anexo 2). Deve-se também gerar os índices de vizinhança para cada novo triângulo, os quais podem ser obtidos por uma pesquisa ordenada no conjunto *T*, para encontrar os triângulos que possuam os segmentos adjacentes aos segmentos do novo triângulo.

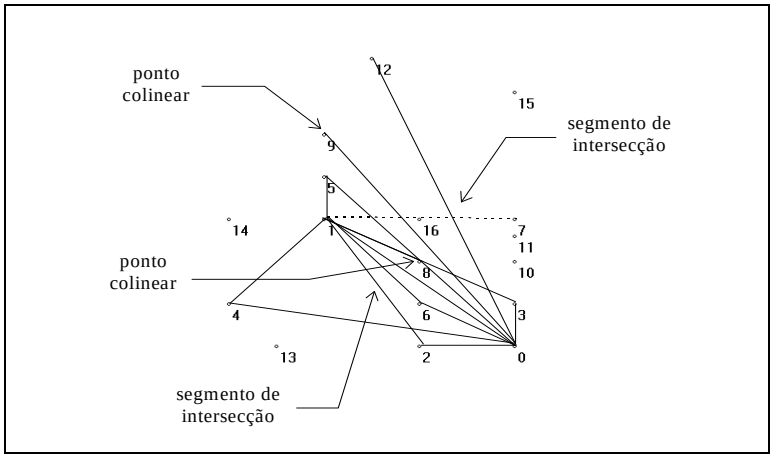
Com uma pesquisa ordenada do conjunto *T* que contém os índices dos triângulos, gera-se o conjunto de pontos pertencentes à cobertura convexa. Estes pontos são definidos pelos triângulos que não possuem um ou dois índices de vizinhança.

O teste de intersecção entre dois segmentos utiliza as definições da equação da reta, para determinar o ponto de intersecção **I** entre as duas retas suportes dos segmentos de

pesquisa. O ponto de intersecção **I**, encontrando-se dentro do intervalo dos dois segmentos de pesquisa, indica que este ponto representa um ponto de intersecção entre estes dois segmentos. Este procedimento pode ser verificado no anexo 3.

Caso este ponto **I** esteja dentro do intervalo (segmentos 1,2 e 0,4 - 1,3 e 0,5 - 1,7 e 0,5 na FIGURA 13), não se pode formar um novo segmento e deve-se testar a intersecção no próximo segmento de pesquisa, mas, caso contrário, continua-se o teste de intersecção. Se o segmento de pesquisa não interseccionar nenhum segmento já existente no conjunto ***S*** (conjunto de segmentos), o segmento de pesquisa passa a ser um novo segmento no conjunto ***S*** e deve-se testar a intersecção no próximo segmento de pesquisa.

FIGURA 13 - Segmentos de pesquisa



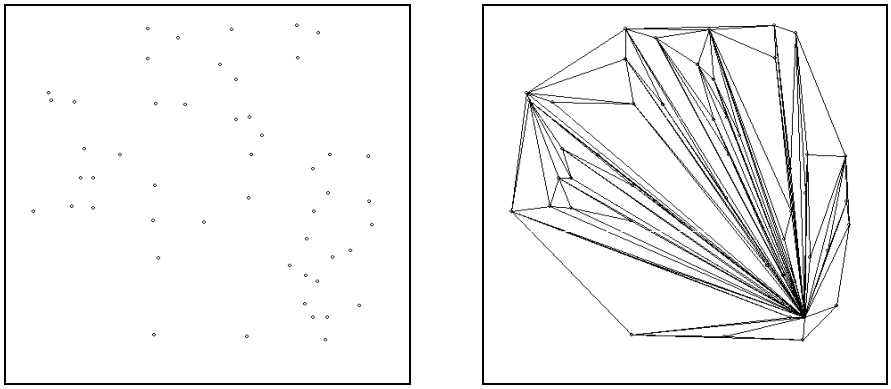

Fonte: Reis, 1997.

Outra consistência que deve ser feita ao se calcular o ponto de intersecção **I** entre os dois segmentos é de verificar se estes são paralelos entre si, podendo assim formar segmentos colineares (anexo 2). Se são segmentos colineares e não estão sobrepostos (ou contíguos), continua-se o teste de intersecção com o segmento de pesquisa (segmentos 5,9 e 1,5 - 5,8 e 0,8 na FIGURA 13). Mas, se os segmentos estão sobrepostos e se o sobreposto for o segmento de pesquisa (segmentos 0,7 e 0,3 - 1,9 e 1,5 na FIGURA 13), deve-se testar a intersecção com o próximo segmento de pesquisa. Se o sobreposto for um segmento já existente no conjunto *S*, apaga-se este segmento e o segmento de pesquisa (segmentos 1,8 e 1,3 na FIGURA 13) torna-se um novo segmento no conjunto ***S***, devendo-se testar a intersecção para o próximo segmento de pesquisa.

Para verificar se um ponto é externo a um triângulo, pesquisa-se saber se este está à esquerda ou à direita das três linhas formadas pelos três vértices deste triângulo. Este procedimento que se utiliza do produto vetorial pode ser encontrado no anexo 1.

A FIGURA 14 possui 50 pontos, intervalo fechado de coordenadas de 0 a 400, 10 pontos na cobertura convexa, 88 triângulos e 137 lados.

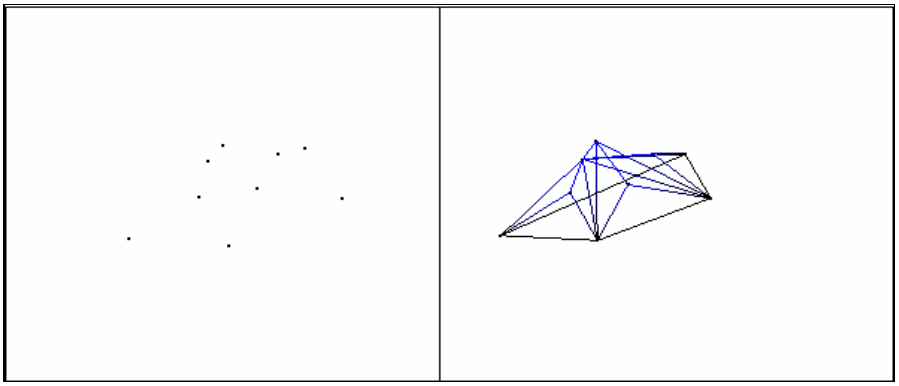
FIGURA 14 - Pontos dispersos 2D– Triangularização



Fonte: Reis, 1997.

A Figura 15 possui 9 pontos, intervalo fechado de coordenadas de 100 a 900, 4 pontos na cobertura convexa, 12 triângulos e 20 lados.

FIGURA 15 - Pontos dispersos 3D– Triangularização

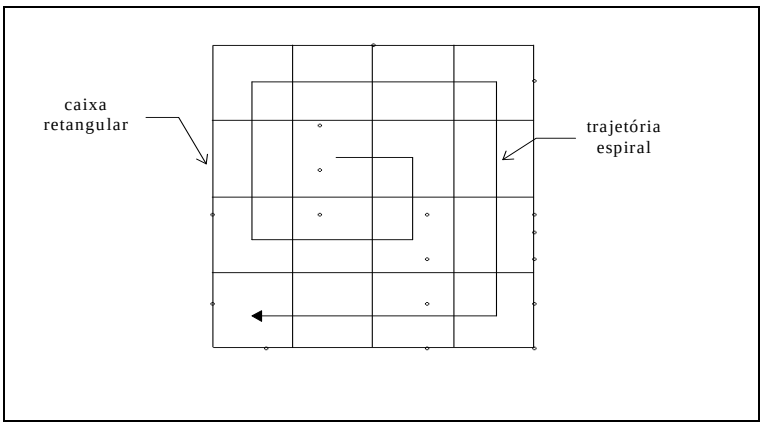


## 3.2.2 TRIANGULARIZAÇÃO ESPIRAL

Segundo Reis (1997), este algoritmo inicia pela definição do conjunto ***V*** de pontos da cobertura convexa através do algoritmo do capítulo 3.1.

Após esta definição os pontos do conjunto ***V*** são excluídos do conjunto ***P*** de pontos dispersos. Os pontos $P_i$ do conjunto ***P*** são particionados em caixas retangulares (Lee, 1980), em uma grade, definida pelo envelope que abrange todos os pontos $P_i$ (FIGURA 16). Esta grade contém tanto na horizontal como na vertical um número de caixas retangulares igual a raiz quadrada da quantidade de pontos P do conjunto ***P***, resultando num total de caixas aproximadamente igual ao número de pontos P do conjunto ***P*** (Lee, 1980).

FIGURA 16 - Caixas retangulares

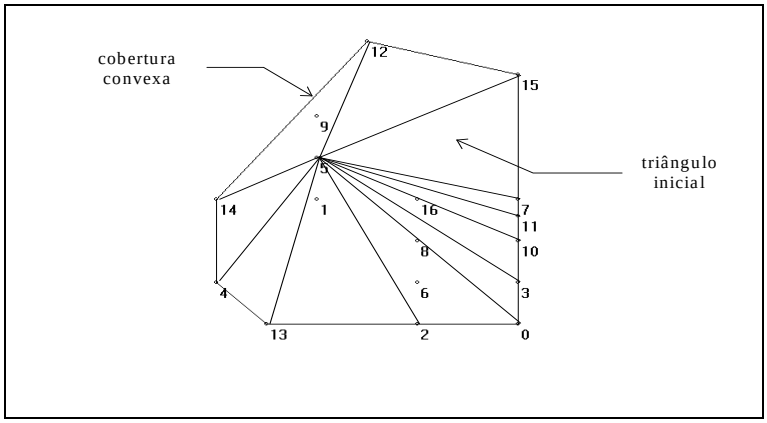



Fonte: Reis, 1997.

Segundo Reis (1997), a definição de qual ponto $P_i$ pertence à qual caixa retangular é feita por uma pesquisa ordenada de todos os pontos do conjunto ***P***. Esta definição compara as coordenadas do ponto $P_i$ com os limites de cada caixa retangular. Dependendo de quanto os pontos $P_i$ estão dispersos uns dos outros, pode ocorrer de mais de um ponto localizar-se em uma mesma caixa.

Tendo-se a definição dos pontos do conjunto ***V*** da cobertura convexa e o particionamento dos pontos $P_i$ na grade, inicia-se o processo de triangularização. Os triângulos iniciais do conjunto ***T*** são definidos por um ponto da caixa mais central à grade e dois pontos sucessivos do conjunto ***V*** (FIGURA 17).

FIGURA 17 - Triangularização parcial



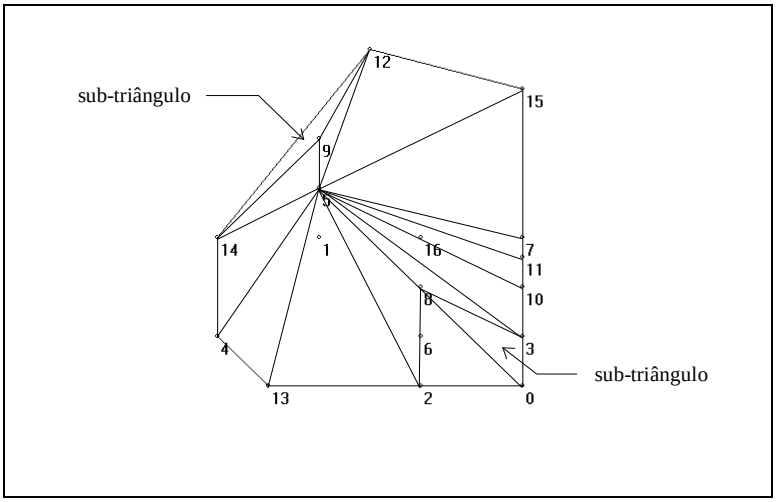

Fonte: Reis, 1997.

Com a triangularização inicial definida, deve-se triangularizar os próximos pontos $P_i$ do conjunto ***P*** internos à cobertura convexa. Este ponto $P_i$ é obtido de uma forma espiral iniciando-se pela caixa mais central à grade, exceto pelo ponto que definiu a triangularização inicial (FIGURA 17).

Pode-se verificar claramente que estes pontos $P_i$ são seguramente internos a algum triângulo da triangularização inicial. Esta pesquisa é feita através do cálculo do produto vetorial (1) a todos os triângulos do conjunto ***T***, verificando se o ponto $P_i$ está sobre ou é interno a um destes triângulos (anexo 4).

Tendo-se o triângulo que abrange o ponto $P_i$, deve-se testar se este ponto é colinear com algum lado do referido triângulo (anexo 2). Caso não seja colinear, geram-se três triângulos novos com o ponto $P_i$ e os pontos do triângulo de abrangência (pontos 9,12,14; 9,14,5 e 9,5,12 na FIGURA 17). Caso o ponto $P_i$ seja colinear, geram-se quatro triângulos novos, já que este ponto nunca deverá ser colinear a um dos lados da cobertura convexa. Estes quatro triângulos novos serão formados pelo ponto $P_i$, um ponto do lado colinear e um ponto não pertencente a este lado (pontos 8,3,5; 8,0,3; 8,5,2 e 8,2,0 na FIGURA 18).

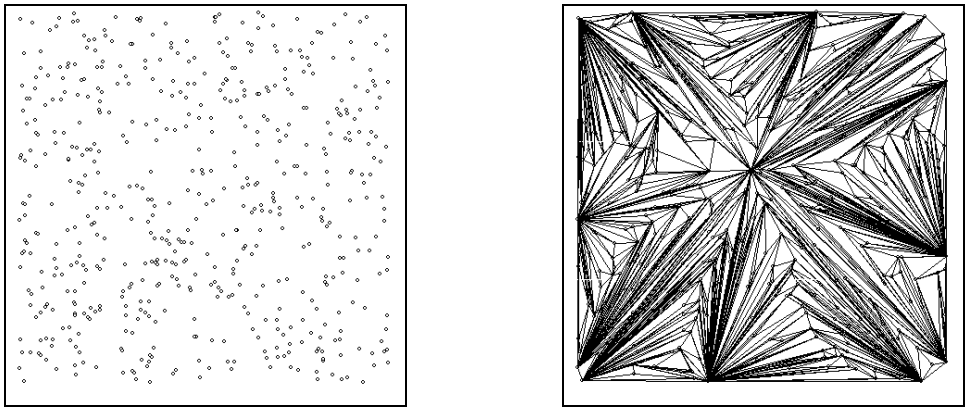
FIGURA 18 -Sub-triangularização




Fonte: Reis, 1997.

A FIGURA 19 possui 550 pontos, intervalo fechado de coordenadas de 0 a 400, cobertura convexa de 15 pontos, 1083 triângulos, 1632 lados e uma grade de 23 por 23 caixas.

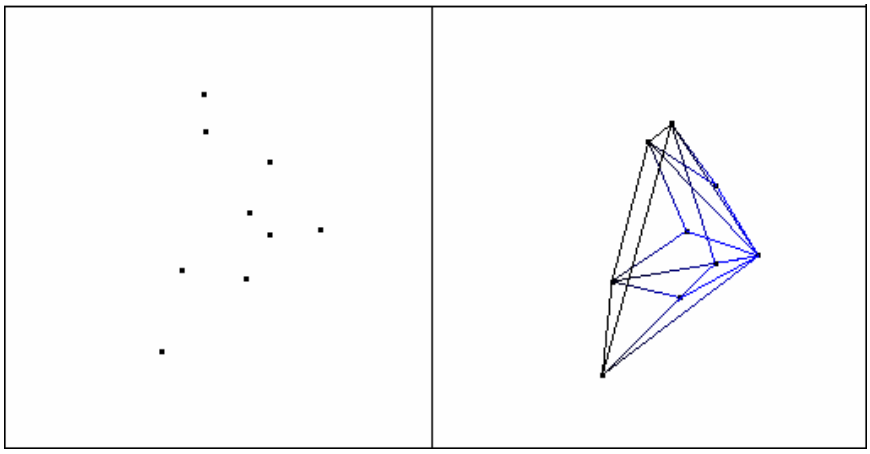
FIGURA 19 - Pontos dispersos 2D – Triangularização Espiral



Fonte: Reis, 1997.

A Figura 20 possui 9 pontos, intervalo fechado de coordenadas de 100 a 900, 4 pontos na cobertura convexa, 12 triângulos e 20 lados.

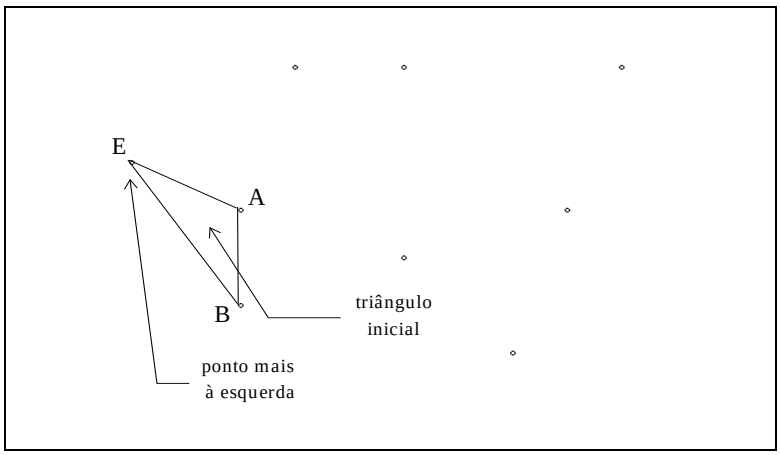
FIGURA 20 - Pontos dispersos 3D– Triangularização Espiral



### 3.2.3 PESQUISA DO VIZINHO OPOSTO

Sendo um conjunto ***P*** constituído de n pontos $P_i$ coplanares, este algoritmo tem como essência uma série de passos, iniciando-se pela definição do primeiro triângulo. Este triângulo é formado pelo ponto E mais à esquerda (FIGURA 21) e pelos dois pontos A e B mais próximos deste. Este triângulo inicial não poderá ser formado se estes três pontos forem três pontos colineares (anexo 2) (Reis, 1997).

FIGURA 21 - Triângulo inicial (E,B e A)




Fonte: Reis, 1997.

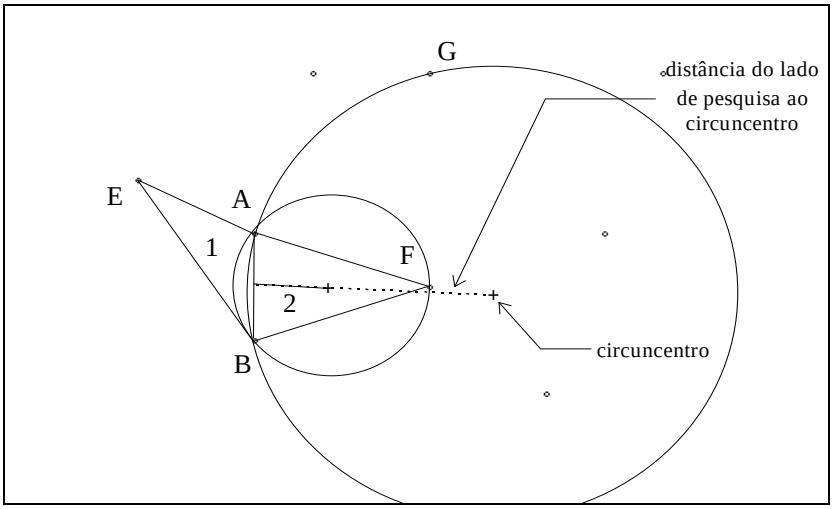
Segundo Reis (1997), com este triângulo inicial, pesquisam-se todos os pontos do conjunto ***P*** para verificar a possibilidade de se gerar novos triângulos adjacentes a este triângulo. A fim de garantir que a triangularização possua um sentido anti-horário, deve-se testar a cada novo triângulo o sentido de seus vértices pelo cálculo do determinante dos seus pontos projetados num plano 2D (anexo 2).

Como a pesquisa dos novos triângulos está relacionada com o fator de adjacência, fica muito clara a importância de se garantir a atribuição dos índices de vizinhança a cada novo triângulo e aos seus respectivos lados adjacentes.

O triângulo inicial (E, B e A na FIGURA 21) não possui nenhum vizinho adjacente, desta forma se faz uma pesquisa para os seus três lados. Para determinar um novo triângulo adjacente, deve-se pesquisar um ponto $P_i$ do conjunto ***P*** que possa definir um triângulo com dois pontos do triângulo inicial (seu lado).

Segundo McLain (1976), esta pesquisa pode ser restringida a um conjunto ***Q*** de pontos $P_i$ que estão do lado oposto ao vértice que não pertence ao lado pesquisado. Após esta restrição, os pontos pertencentes ao conjunto ***Q*** são ordenados pela distância entre o lado de pesquisa ao circuncentro de uma circunferência, que é formada pelos pontos do lado em pesquisa (A e B) e um ponto do conjunto ***P*** (FIGURA 22).

FIGURA 22 - Ponto F mais próximo ao lado de pesquisa (A e B)



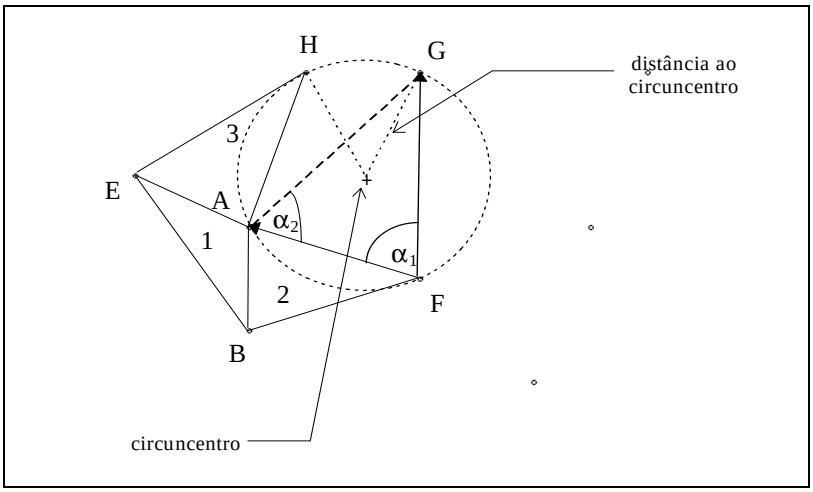

Fonte: Reis, 1997.

O ponto no conjunto ***Q*** mais próximo do lado de pesquisa formará um novo triângulo (A, B e F na FIGURA 22) pertencente ao conjunto ***T***, devendo-se também definir os índices dos seus triângulos vizinhos. Para os lados adjacentes entre o triângulo de pesquisa e o novo triângulo gerado fica implícita a definição dos índices de vizinhança. Mas para obter-se os índices de vizinhança para os outros lados, deve-se fazer uma pesquisa ordenada no conjunto ***T***, para encontrar os triângulos que possuam os segmentos adjacentes aos segmentos do novo triângulo.

Terminada a pesquisa de todos os lados de um triângulo, inicia-se novamente a pesquisa de vizinhança do próximo triângulo do conjunto ***T***. Esta pesquisa deve ser feita somente aos lados do triângulo que já não possuem um índice de vizinhança. Quando a pesquisa dos pontos $P_i$ referentes a um lado do triângulo resultar em nenhum ponto em ***Q***, faz-se com que os pontos deste lado pertençam ao conjunto ***J*** de pontos da cobertura convexa do conjunto ***P***.

Um caso que pode ocorrer é a existência de mais de um ponto com a mesma distância ao circuncentro (FIGURA 23). Estes pontos podem ser coincidentes em suas coordenadas ou não. Se eles tiverem as mesmas coordenadas, deve-se deixar somente um destes pontos no conjunto ***P***. Porém, se existir mais um ponto com coordenadas diferentes e mesma distância ao circuncentro, deve-se ordená-los pela soma de dois ângulos $\alpha_1$ e $\alpha_2$ , já que o ponto com menor soma estará mais próximo ao lado (A e F) (FIGURA 23).

FIGURA 23 - Pontos G e H com a mesma distância ao circuncentro

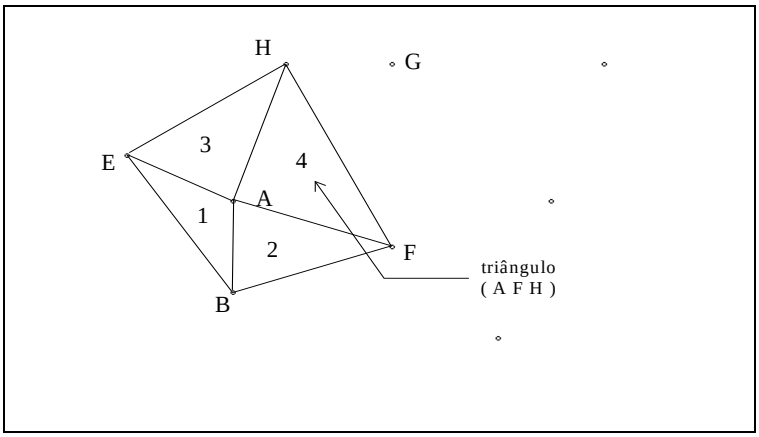



Fonte: Reis, 1997.

Os ângulos $\alpha_1$ e $\alpha_2$ são obtidos pelo cálculo da divisão do produto escalar de dois vetores pelos seus módulos. Estes dois vetores são formados pelos pontos do lado de pesquisa A e F, e por um dos pontos com a mesma distância ao circuncentro. Na FIGURA 23, ilustram-se os ângulos $\alpha_1$ e $\alpha_2$ para o ponto G associado ao lado $\overline{AF}$ em pesquisa, onde os vetores para $\alpha_1$ são $\overrightarrow{FA}$ e $\overrightarrow{FG}$ e para $\alpha_2$ são $\overrightarrow{AF}$ e $\overrightarrow{AG}$.

Desta forma, no exemplo, o novo triângulo é definido pelos dois pontos do lado de pesquisa A e F, e pelo ponto H pois é o que tem a menor soma dos dois ângulos $\alpha_1$ e $\alpha_2$ (FIGURA 24). Maiores detalhes sobre este ordenamento angular podem ser encontrados em McLain (1976).

FIGURA 24 - Triângulo gerado pela coordena angular

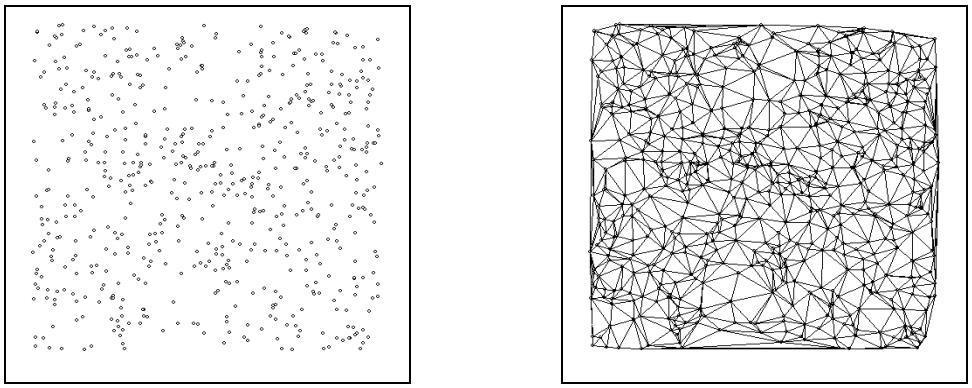



Fonte: Reis, 1997.

Este procedimento termina após pesquisarem-se todos os lados dos triângulos do conjunto ***T***, resultando em uma triangularização Delaunay (ver capítulo 4.3) dos pontos do conjunto ***P*** (Reis, 1997).

O exemplo da FIGURA 25 tem 600 pontos, intervalo fechado de coordenadas de 0 a 400, com 22 pontos na cobertura convexa, 1176 triângulos e 1775 lados (Reis, 1997).

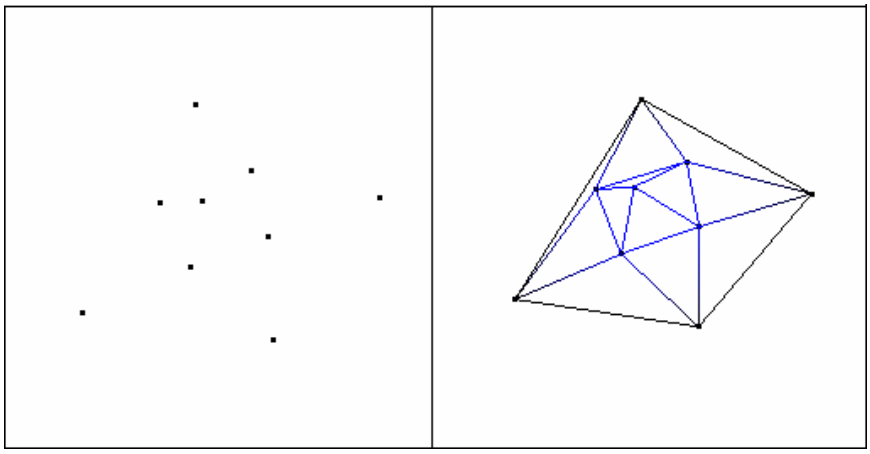
FIGURA 25 - Pontos dispersos 2D - Triangularização otimizada



Fonte: Reis, 1997.

A Figura 26 possui 9 pontos, intervalo fechado de coordenadas de 100 a 900, 4 pontos na cobertura convexa, 12 triângulos e 20 lados.

FIGURA 26 - Pontos dispersos 3D– Triangularização Otimizada

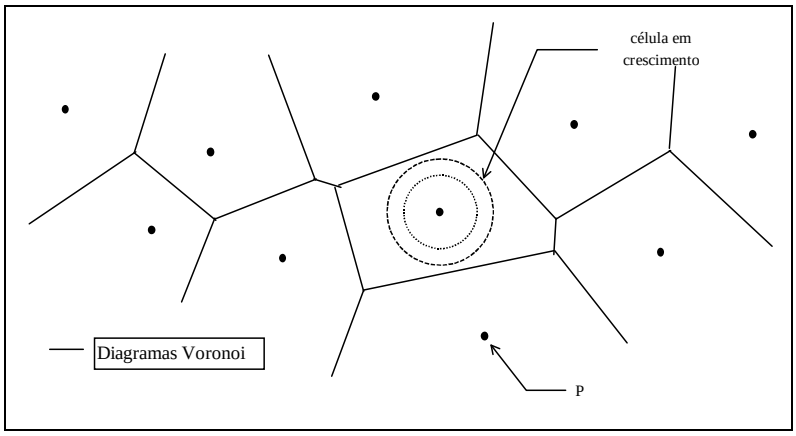


## 3.3 TRIANGULARIZAÇÃO DELAUNAY

Um conjunto de pontos dispersos pode ser particionado de várias formas, permitindo assim gerar inúmeras combinações de diagramas. Proveniente disto surge a questão de qual combinação pode se aproximar mais da representação real. Este problema foi teoricamente resolvido no século passado pelos diagramas de Voronoi (Agishtein, 1991).

Lee (1980) explica a definição de diagramas Voronoi através de um processo de células em crescimento, no qual assume-se que cada ponto P de um conjunto ***P*** é o núcleo de uma célula em crescimento. Estas células se propagam para o exterior dos seus núcleos, simultânea e uniformemente. As bordas das células em crescimento param de se propagar ao entrar em contato com as bordas das outras células em crescimento, exceto pelas células que estão sobre a cobertura convexa do conjunto ***P***, as quais se expandem as células indefinidamente (FIGURA 27).

FIGURA 27 - Diagramas Voronoi do conjunto ***P***



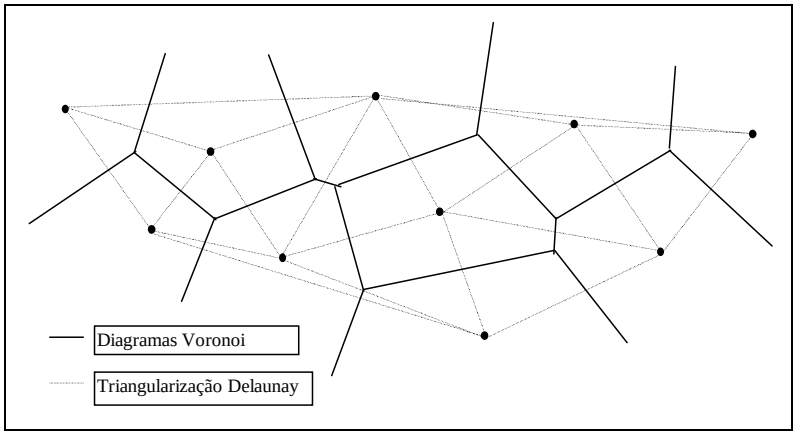

Fonte: Reis, 1997.

Desta forma as células externas definem poligonais sobre a cobertura convexa e as células internas formam um conjunto de polígonos convexos não sobrepostos (um polígono sobre cada núcleo). Estes polígonos fechados representam os diagramas Voronoi do plano que contém os pontos do conjunto ***P*** (Reis, 1997).

Como no particionamento por diagramas Voronoi, uma triangularização também pode gerar inúmeras combinações de relações topológicas entre os pontos dispersos (Choi, 1988). Em particular, entre uma dessas combinações, têm-se uma triangularização otimizada, conhecida como triangularização Delaunay. Este tipo de triangularização é grandemente utilizado para definir os modelos de grades irregulares triangularizadas utilizados em *software* comercial (Felgueiras, 1995).

Pode-se obter uma triangularização Delaunay dos diagramas Voronoi quando a triangularização é executada sobre um domínio plano, já que existe um comprometimento gráfico entre estas definições (Choi, 1988; Felgueiras, 1995). Este comprometimento pode ser observado na FIGURA 28, onde pode-se ver que os lados Delaunay (tracejado) são ortogonais a seus correspondentes lados Voronoi (sólidos), mas não necessariamente se interseccionam (Felgueiras, 1995). Observa-se também que cada ponto Voronoi corresponde a um triângulo e cada lado Voronoi a um lado Delaunay (Lee, 1980).

FIGURA 28- Diagramas Voronoi e triângulos Delaunay




Fonte: Reis, 1997.

Mesmo podendo se obter uma triangularização Delaunay dos diagramas Voronoi (Lee, 1980), optou-se em não utilizar este procedimento e sim o de otimizar os possíveis triângulos delgados obtidos com os algoritmos de triangularização anteriormente citados (ver capítulo 4.2) (Reis, 1997). Não há necessidade desta forma de um procedimento com duas etapas, a de construir os diagramas Voronoi e a de convertê-los em triângulos, com um tempo adicional de O(n) (Preparata, 1985).

Entre as possíveis formas de se gerar os triângulos, a triangularização Delaunay permite obter um conjunto de triângulos mais eqüi-angulares. Permite melhorar a representação das superfícies geradas de pontos interpolados a partir dos vértices destes triângulos, pois asseguram que cada ponto sobre a superfície está o mais próximo possível de um vértice (Spatial, 1994; Goodchild, 1995).

A triangularização Delaunay possui interessantes propriedades, tais como: ela é única entre as diversas topologias que uma triangularização pode ter, maximiza os mínimos ângulos internos de cada triângulo, os pontos de um triângulo não podem ser colineares, a cobertura convexa sempre faz parte da grade Delaunay e triângulos Delaunay não são hierárquicos (eles não podem ser agregados ou particionados) (Spatial, 1994; Goodchild, 1995; Lee, 1980; Preparata, 1985).

Entre os algoritmos de triangularização acima demonstrados, apenas um resulta em uma triangularização Delaunay. Mesmo que alguns se aproximem bastante, tem-se ainda a formação de muitos triângulos delgados, necessitando-se desta forma de um método de otimização (critério mínimo-máximo, critério do círculo, critério da menor diagonal interna e critério esférico) a fim de ter-se uma triangularização Delaunay (Choi, 1988; Lawson, 1977; Lee, 1980).

### 3.3.1 GERAR DELAUNAY

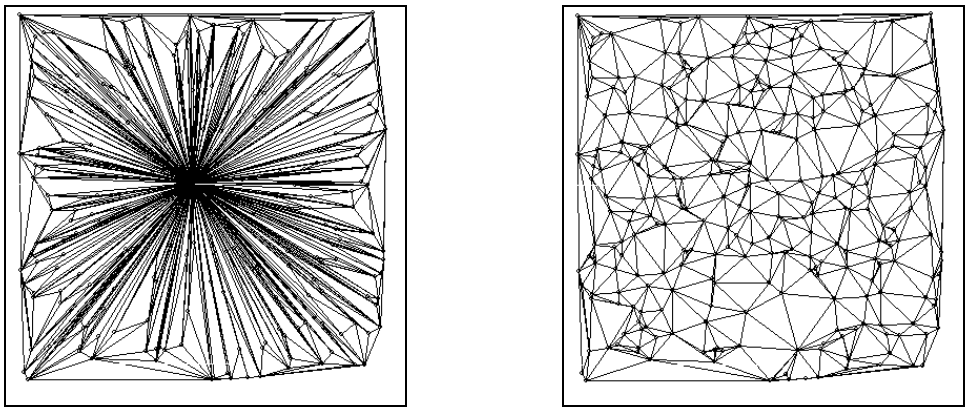
O método de otimização que foi estudado é o de menor diagonal. Pois em Mirante (1982), tem-se um melhoramento da triangularização inicial pelo critério da menor diagonal interna para poder gerar uma aproximação da triangularização Delaunay desejada.

Segundo Reis (1997), nesta otimização utilizam-se polígonos exatamente convexos obtidos por uma pesquisa ordenada em todos os triângulos do conjunto ***T*** (triangularização inicial). Desta forma, calculando-se apenas as duas distâncias das diagonais internas.

Caso a diagonal interna dos triângulos propostos (diferentes da atual triangularização) seja menor, deve-se alterar o polígono de pesquisa a fim de formar a outra configuração possível de triângulos e ajustar os seus referidos indicadores de vizinhança.

Na FIGURA 29-b tem-se o resultado de uma triangularização otimizada de um exemplo (FIGURA 29-a) com 300 pontos, intervalo fechado de coordenadas de 0 a 350, 14 pontos na cobertura convexa, 584 triângulos, 883 lados e 6 iterações no procedimento de otimização ao conjunto ***T*** de triângulos.

FIGURA 29 - a) Triangularização - b) Triangularização otimizada



Fonte: Reis, 1997.

# 4 DESENVOLVIMENTO DO PROTÓTIPO

Com base nos conceitos apresentados nas seções anteriores, tornou-se possível o desenvolvimento do protótipo de *software* que permite gerar a triangularização dos pontos. Neste capítulo será abordado a especificação, implementação e o funcionamento do protótipo que foi desenvolvido.

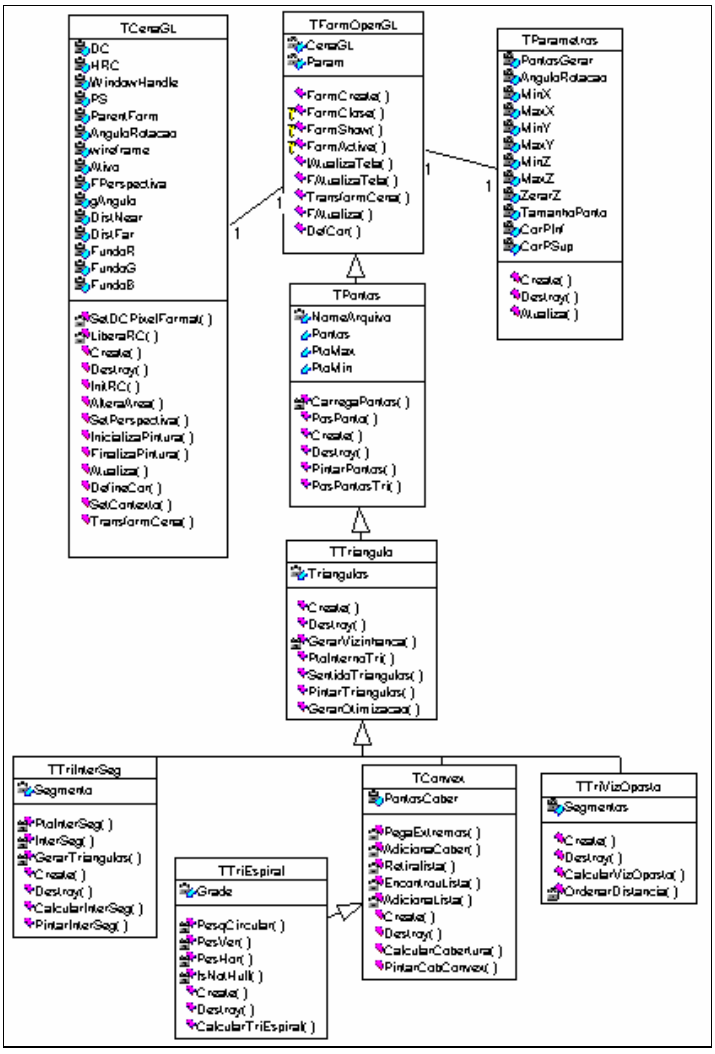
## 4.1 ESPECIFICAÇÃO DO PROTÓTIPO

Neste tópico poderá ser observado o diagrama de classe e alguns dos diagramas de seqüência. Estes diagramas possibilitam uma visão geral do funcionamento do protótipo, utilizando a representação gráfica.

### 4.1.1 DIAGRAMA DE CLASSE

Neste diagrama, visualiza-se individualmente as classes, representadas por retângulo, com três compartimentos. O primeiro é para o nome da classe que é obrigatório, o segundo e o terceiro compartimentos são opcionais e podem ser usados para listar, respectivamente, os atributos e as operações definidas para a classe. Conforme FIGURA 30, no diagrama de classe, são visualizados os atributos e as operações das classes utilizadas no desenvolvimento do protótipo.

FIGURA 30 – Diagrama de classe




A principal classe do sistema é a TFormOpenGL, é esta classe que tem a base para a geração das imagens; ligadas a esta classe tem-se a TCenaGL e TParametros que, respectivamente, possui os comandos para a utilização do OpenGL e parêmetros para a geração das imagens.

A classe TPontos possui a lista de todos os pontos que serão triangularizados, além de funções como: pintar pontos e posição de um ponto a um segmento.

Os triângulos gerados são armazenados e pintados na classe Ttriangulo, a otimização de triangulos também se encontra nesta classe.

A classe Tconvex, possui funções expecíficas referente a geração da cobertura convexa. As outras três classes: TtriInterSeg, TtriEspiral e TtriVizOposto; são as classes com os métodos de triangularização.

### 4.1.2 DIAGRAMA DE SEQÜÊNCIA

Os diagramas de seqüência que foram modelados, tem como intuito de visualizar algumas das operações realizadas no protótipo de software. Os diagramas de seqüência representados são a geração da Cobertura Convexa (FIGURA 31) e a Geração da Otimização(FIGURA 32).

FIGURA 31 – Seqüência para geração da Cobertura Convexa

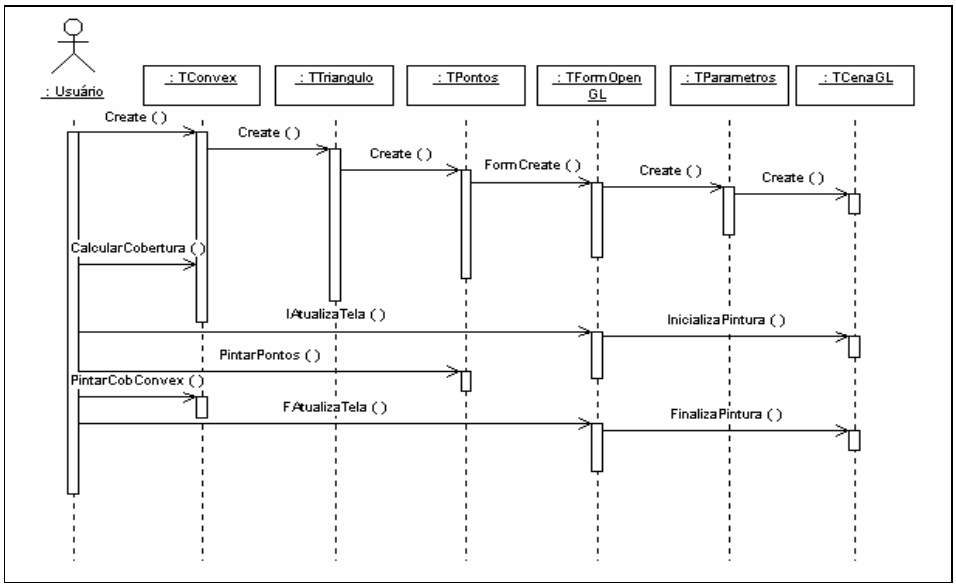
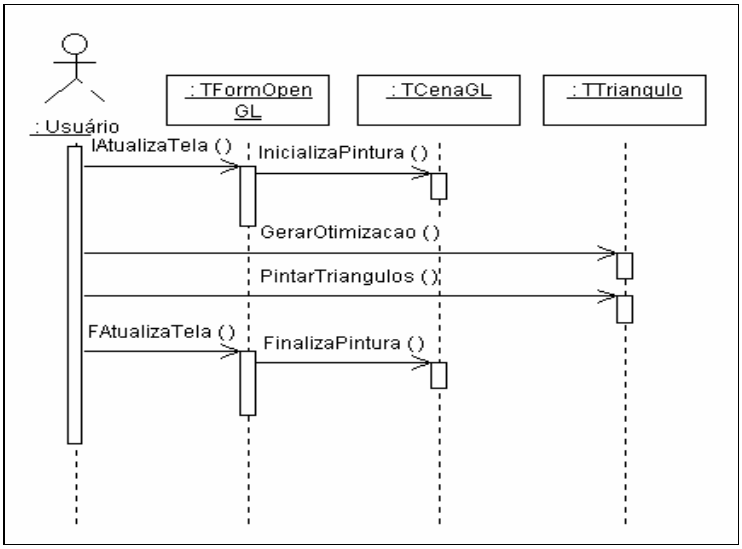


FIGURA 32 – Seqüência para Geração da Otimização




## 4.2 IMPLEMENTAÇÃO DO PROTÓTIPO

Este protótipo foi implementado no ambiente *Delphi* 5.0, sendo que utilizou-se uma programação orientada a objeto, composta de *unit´s* distintas.

OpenGL é uma biblioteca de funções gráficas que estabelece para o programador uma interface uniforme com o hardware gráfico, independente da plataforma. OpenGL permite a criação de programas interativos com imagens coloridas e animação 3D (Rieder, 2000).

Segundo Rieder (2000) a construção de imagens na tela com o OpenGL segue, resumidamente, os seguintes passos principais:

a) construção de formas a partir de primitivas geométricas (descrição matemática de objetos);
b) Posicionamento dos objetos no espaço tridimensional e escolha da perspectiva de visualização da cena;
c) Cálculo das cores de todos os objetos (informação obtida através das cores especificadas para cada objeto e de suas condições de iluminação e textura);

d) Conversão da descrição matemática dos objetos e de suas respectivas cores em pixels na tela. Esse processo é chamado de rasterização.

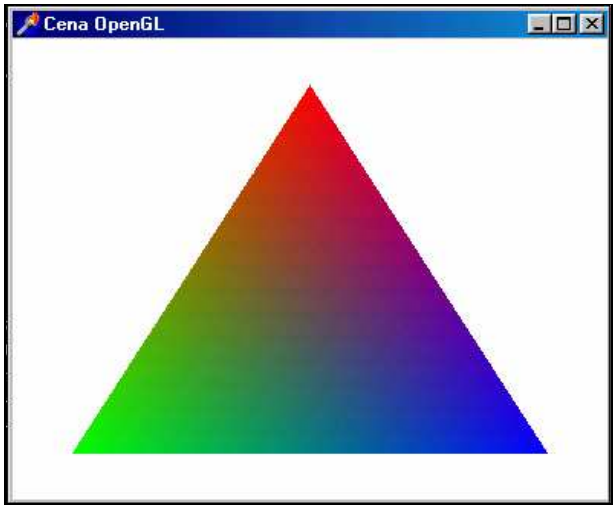
A interface OpenGL consiste de aproximadamente 250 comandos de descrição de objetos e de operações necessárias para produzir aplicações interativas tridimensionais. Sendo uma interface independente de hardware, pode ser implementada em diversas plataformas. O OpenGL permite construir imagens com um grau de complexidade bastante elevado. Isso inclui desde a definição da geometria do objeto, aparência (cor, textura, material), transformações sobre o objeto (escala, rotação e translação), definição de comportamento, animações, interação com o usuário, luzes, etc.

OpenGL é um API (*Aplication Programming Interface*) e, como tal, não provê comandos de alto nível para descrever objetos. Ao invés disso você deve construir o seu modelo a partir de primitivas geométricas (pontos, linhas e polígonos).

Parece difícil pensar como fazer para descrever algo tão complexo com apenas pontos, linhas e polígonos. Assim como um círculo pode ser aproximado por um polígono de muitos lados (sem que ao menos percebamos que este círculo é, na verdade, um polígono), podemos também aproximar uma superfície por uma série de triângulos e quadrados muito pequenos, interconectados de modo a formar uma suave montanha (Rieder, 2000).

Junto com o ambiente Delphi 5.0 foi utilizado a biblioteca gráfica OpenGL. Seguindo basicamente alguns passos, permite-se gerar imagens na tela como a da FIGURA 33, exemplo de uma imagem com o OpenGL dentro do Delphi 5.0.

FIGURA 33 - Cena OpenGL




Primeiro deve-se inicializar um `Form` do ambiente de desenvolvimento. Isto pode ser realizado como no Quadro 1, onde foi colocado uma seqüência de comandos no momento da criação da janela.

Quadro 1 – Criação da cena em OpenGL

```
procedure TForm1.FormCreate(Sender: TObject);
var
  pfd: TPixelFormatDescriptor;
  FormatIndex: integer;
begin
  fillchar(pfd,SizeOf(pfd),0);
  with pfd do
  begin
    nSize      := SizeOf(pfd);
    nVersion   := 1; {versao do descritos}
    dwFlags    := PFD_DRAW_TO_WINDOW or PFD_SUPPORT_OPENGL;
    iPixelType := PFD_TYPE_RGBA; {Para suportar o padrão RGB}
    cColorBits := 24; {suporte 24-bit de cor}
    cDepthBits := 32; {profundidade do eixo z}
    iLayerType := PFD_MAIN_PLANE;
  end; {with}
  glDC := getDC(handle);
  FormatIndex := ChoosePixelFormat(glDC,@pfd);
  SetPixelFormat(glDC,FormatIndex,@pfd);
  GLContext := wglCreateContext(glDC); {funções para administrar
contextos.}
  wglMakeCurrent(glDC,GLContext);
end; {FormCreate}
```

Para o próximo passo será definida a cor de fundo e se desenhará o triângulo que está em degrade, também se fará teste para saber se ocorreu algum erro ao desenhar a cena; esta próxima etapa está exemplificada no Quadro 2.

Quadro 2 – Definição da cor e do triângulo

```
procedure TForm1.FormPaint(Sender: TObject);
begin
  {Cor do fundo}
  glClearColor(1.0,1.0,1.0,0.0);
  glClear(GL_COLOR_BUFFER_BIT);
 glBegin(GL_TRIANGLES);       // Inicia o desenho de Triângulos
  glColor3f(1.0,0.0,0.0);     // Ajusta a Cor para Vermelho
  glVertex3f( 0.0, 0.8, 0.0); // Move uma unidade do centro(Ponto de
Cima)
  glColor3f(0.0,1.0,0.0);     // Ajusta a Cor para Verde
  glVertex3f(-0.8,-0.8, 0.0); // Move uma unidade para a esquerda e
para baixo
  glColor3f(0.0,0.0,1.0);     // Ajusta a cor para Azul
  glVertex3f( 0.8,-0.8, 0.0); // Move uma unidade para direita e
para baixo
 glEnd();// Finaliza o desenho de Triângulos
  {Checagem de erro}
  errorCode := glGetError;
  if errorCode<>GL_NO_ERROR then
    raise Exception.Create('Erro ao Pintar '#13+
gluErrorString(errorCode));
end;
```

No Quadro 2 o código que está entre a função `glBegin()` e `glEnd()`, determina o desenho que será feito, no caso um triângulo, que foi passado como parâmetro utilizando a constante do OpenGL `GL_TRIANGLES.` Após este comando são passados os três pontos necessários para a geração de um triângulo através da função `glVertex3f(x, y, z)`. O degrade é feito de forma automática, bastando apenas passar a cor de cada ponto do triângulo `glColor3f(R,G,B)`.

A seguir, está a destruição e a liberação do contexto gerado pelo OpenGL, esta função será executada no momento em que a janela for fechada (Quadro 3).

Quadro 3 – Destruição da cena em OpenGL

```
procedure TForm1.FormDestroy(Sender: TObject);
begin
  wglMakeCurrent(glDC,0);
  wglDeleteContext(GLContext);
end;
```

A geração de pontos no protótipo é realizada através de uma função randômica onde as coordenadas geradas, respeitam os limites impostos nos parâmetros do sistema. Os pontos gerados não são ordenados, e não é gerado ponto com as mesmas coordenadas `x` e `y`. O arquivo de pontos gerados, é formado por três colunas e n linhas. As colunas representam, respectivamente, as coordenadas `x`, `y` e `z`, separadas pelo caracter ';', cada linha representa um ponto no espaço 3D. Um exemplo do arquivo de pontos encontra-se no Quadro 4.

Quadro 4 – Exemplo de arquivo com Pontos

```
0300;0500;0500;
0500;0500;0700;
0700;0500;0500;
0200;0800;0300;
0500;0700;0500;
0800;0800;0300;
0800;0200;0300;
0500;0300;0500;
0200;0200;0300;
```

Para a geração das cores nos objetos triangularizados é utilizada a função do Quadro 5, onde são passadas as informações dos pontos minimo e máximo e suas respectivas cores definidas nos parâmetros.

Quadro 5 – Cálculo da cor de um ponto

```
Procedure TCenaGL.DefineCor(CorPInf, CorPSup: TRegRGB; Altura, PMax, PMin:
Double);
Var
  R, G, B: glFloat;
Begin
  if   (Pmax-Pmin) = 0 then //quando a coordenada z estiver zerada
       begin
         R:=CorPInf.R;G:=CorPInf.G;B:=CorPInf.B;
       end
  else begin
         R:=ABS((((CorPSup.R-CorPInf.R)*(PMax-Altura))-(CorPSup.R*(PMax-
PMin)))/(PMax-PMin));
```

```
        G:=ABS((((CorPSup.G-CorPInf.G)*(PMax-Altura))-(CorPSup.G*(PMax-
PMin)))/(PMax-PMin));
        B:=ABS((((CorPSup.B-CorPInf.B)*(PMax-Altura))-(CorPSup.B*(PMax-
PMin)))/(PMax-PMin));
      end;
  glColor3f(R, G, B);
End;
```

As transformações geométricas são realizadas através da função do Quadro 6, são passadas como informação para a função; a transformação a ser realizada o ângulo de rotação e as coordenadas do ponto onde, serão aplicadas as modificações.

Quadro 6 – Transformações da Cena

```
Procedure TCenaGL.TransformCena(Tra, Sca, Rot: Boolean; Angulo, X, Y, Z:
GLdouble);
begin
  If   Tra Then
       glTranslate(X, Y, Z);
  If   Sca Then
       glScale(X, Y, Z);
  If   Rot Then
       Begin
         AnguloRotacao:=AnguloRotacao+Angulo;
         if   AnguloRotacao >= 360 Then AnguloRotacao := 0;
         glRotate(AnguloRotacao, X, Y, Z);
       End;
end;
```

Uma das principais funções do sistema é a que desenha os triângulos (Quadro 7), é feita uma repetição para informar todos os triângulos, e antes que cada ponto do triângulo seja informado, é calculada sua cor, para se formar a imagem. A cor do ponto é definida através da coordenada `Z` (altura).

Quadro 7 – Criação dos Triângulos

```
Procedure TTriangulo.PintarTriangulos;
var
  Ind: Cardinal;
begin
  glBegin(GL_TRIANGLES);
  for Ind:=0 to Length(Triangulos)-1 Do
      begin
        DefCor(Pontos[Triangulos[Ind].P1].Z, PtoMax, PtoMin);
        glVertex3f(Pontos[Triangulos[Ind].P1].X,
Pontos[Triangulos[Ind].P1].Y, Pontos[Triangulos[Ind].P1].Z);
        DefCor(Pontos[Triangulos[Ind].P2].Z, PtoMax, PtoMin);
        glVertex3f(Pontos[Triangulos[Ind].P2].X,
Pontos[Triangulos[Ind].P2].Y, Pontos[Triangulos[Ind].P2].Z);
        DefCor(Pontos[Triangulos[Ind].P3].Z, PtoMax, PtoMin);
        glVertex3f(Pontos[Triangulos[Ind].P3].X,
Pontos[Triangulos[Ind].P3].Y, Pontos[Triangulos[Ind].P3].Z);
```

```
      end;
  glEnd;
end;
```

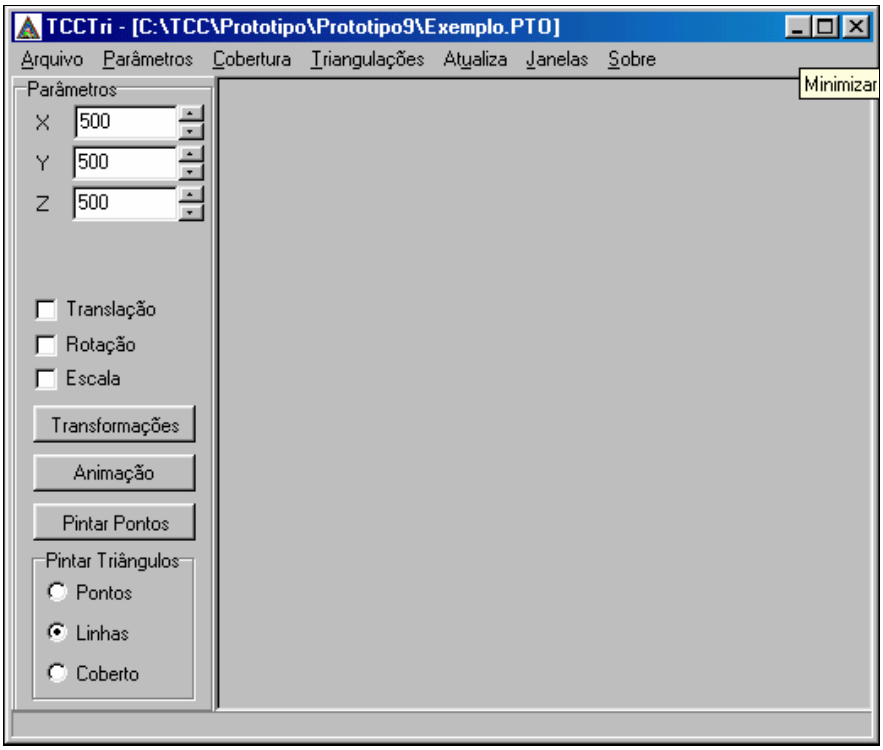
## 4.3 FUNCIONAMENTO DO PROTÓTIPO

A seguir apresenta-se o funcionamento do protótipo incluindo suas telas.

O protótipo, chamado de TCCTri, gera uma superfície triangularizada à partir de pontos dispersos num espaço tridimensional. Com a triangularização concluída, pode-se executar várias transformações sobre a superfície gerada e visualizá-la de vários ângulos.

A FIGURA 34 apresenta a tela principal e suas opções de menu, onde na parte superior da tela é mostrado um modelo de pontos que foi aberto de um arquivo chamado Exemplo.PTO de pontos que poderão ser triangularizados.

FIGURA 34 – Tela principal do TCCTri




As opções do menu do TCCTri são as seguintes: **Arquivo**, **Parâmetros**, **Cobertura**, **Triangulações**, **Atualiza**, **Janelas** e **Sobre**. O protótipo também exibe uma coluna a esquerda

com valores para **X**, **Y** e **Z**; que referem-se a um ponto onde ocorrerá as aplicações de transformações geométricas.

O botão **Transformações** irá executar a transformação geométrica selecionada, que pode ser: **Translação**, **Rotação** e **Escala**. O botão **Animação** simula o de Transformação, só que repetidas vezes até que seja clicado novamente.

O botão **Pintar Pontos** é utilizado para visualização dos pontos que serão triangularizados ou que já foram triangularizados.

As opções encontradas em Pintar Triângulos: **Pontos**, **Linhas** ou **Coberto**, são utilizados após a geração de umas das técnicas de triangularização, pois irão apresentar, respectivamente, os pontos, as linhas ou os triângulos totalmente pintados.O protótipo possui uma barra inferior para indicar que está processando alguma informação.

As opções do menu **Arquivo** estão relacionadas aos processos de gravação e leitura (FIGURA 35):

a) **Abrir** (Ctrl+A) prepara o TCCTri para que seja informado um novo arquivo de pontos para se triangularizar, o nome do arquivo é informado através de uma caixa de diálogo como mostra a FIGURA 36;

b) **Gerar Pontos**, gera o número de pontos contidos nos parâmetros, estes pontos são gerados de forma randômica, será solicitado um nome para o arquivo que conterá os pontos através de um a caixa semelhante a da FIGURA 36;

c) **Sair** (Ctrl+X) finaliza o TCCTri.

FIGURA 35 – Opção Arquivo

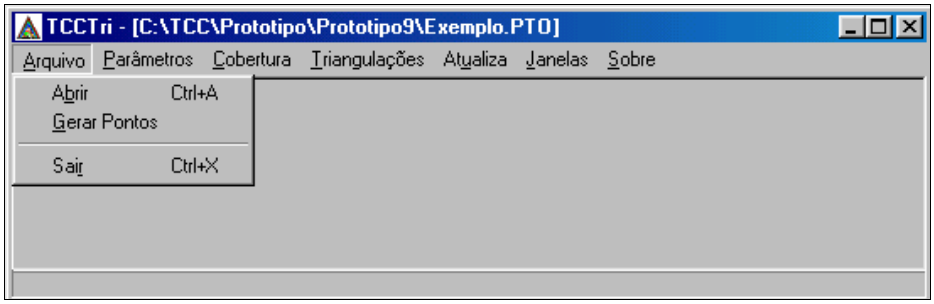
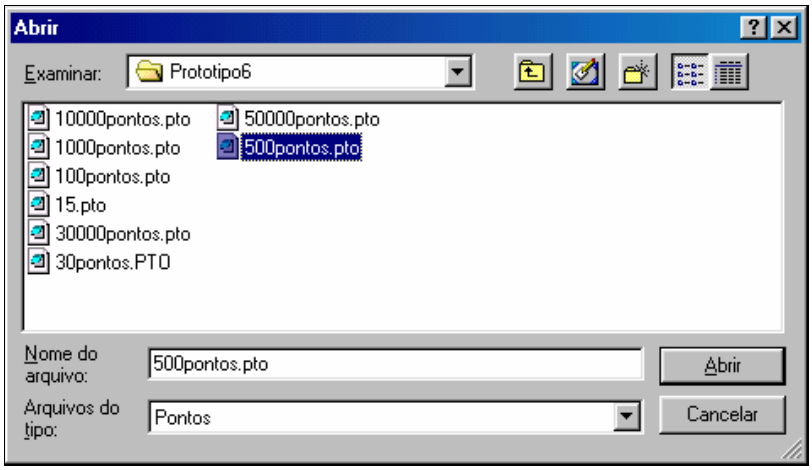
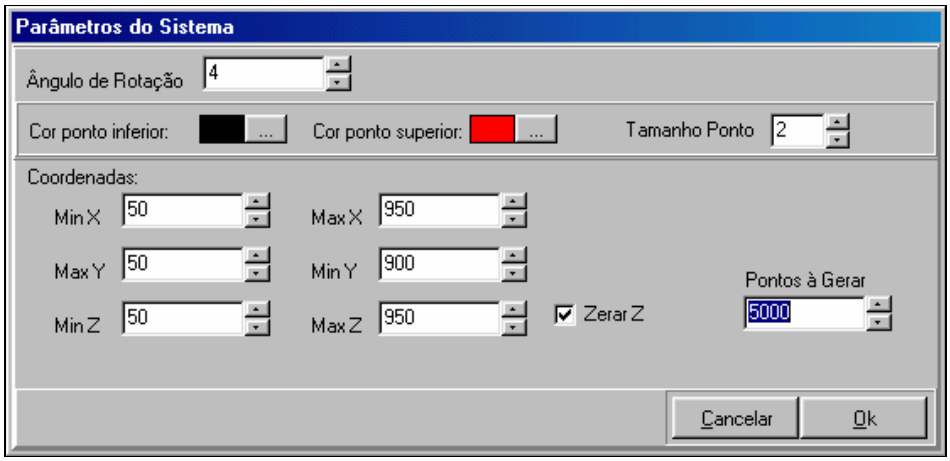


FIGURA 36 – Opção Abrir




As informações contidas em **Parâmetros** (FIGURA 37), estão relacionadas a geração de pontos, formação e transformação das imagens.

FIGURA 37 – Tela de parâmetros




O **Ângulo de Rotação** é utilizado no momento da realização da transformação geométrica, rotação.

As informações de **Cor ponto inferior** e **Cor ponto superior**, são utilizadas para a geração das imagens; como somente é informado a cor do menor e maior `z` (Altura)`,` os outros pontos (intermediários) são calculados, desta forma, gera-se uma imagem em degradê.

O **Tamanho Ponto** serve para definir a forma com que o ponto será desenhado.

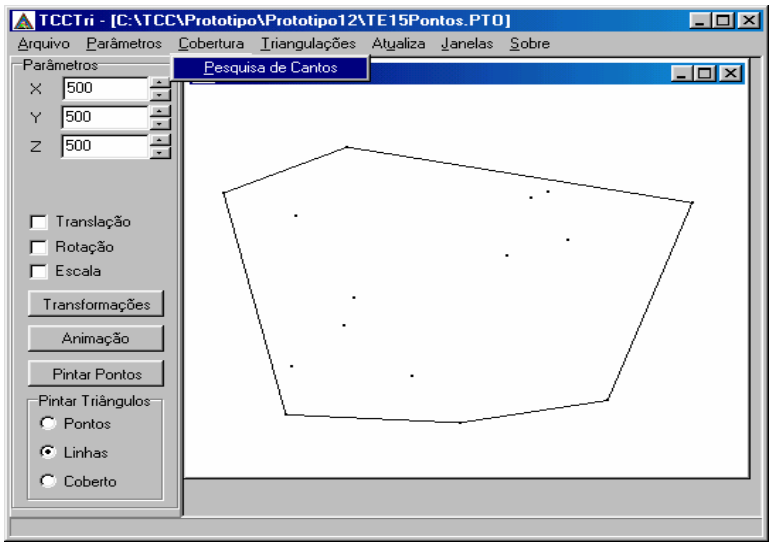
A informação **Pontos à Gerar** é utilizada no momento da criação de um arquivo de pontos, assim como as informações de limite inferior e superior de cada vértice das coordenadas **Min X**, **Max X**, **Min Y**, **Max Y**, **Min Z** e **Max Z**; já o campo **Zerar Z**, definirá se a coordenada de z de cada ponto gerado, terá algum valor diferente de 0 (zero) no momento de geração dos pontos.

Os botões **Cancelar** e **Ok**, irão respectivamente ignorar e salvar, as modificações feitas nos parâmetros.

Todas as telas que serão apresentadas deste ponto em diante estarão com as informações dos parâmetros que foram apresentadas na FIGURA 37. Enquanto não for aberto um arquivo de pontos, as opções **Cobertura**, **Triangulações**, **Atualiza** e **Janelas**; estarão desabilitadas.

Em **Cobertura** será apresentada a **Pesquisa de Cantos** (FIGURA 38), que ao clicar começará a calcular a cobertura convexa utilizando o algoritmo apresentado na seção 3.1.1. Após os cálculos dos pontos será apresentada a imagem da Cobertura Convexa

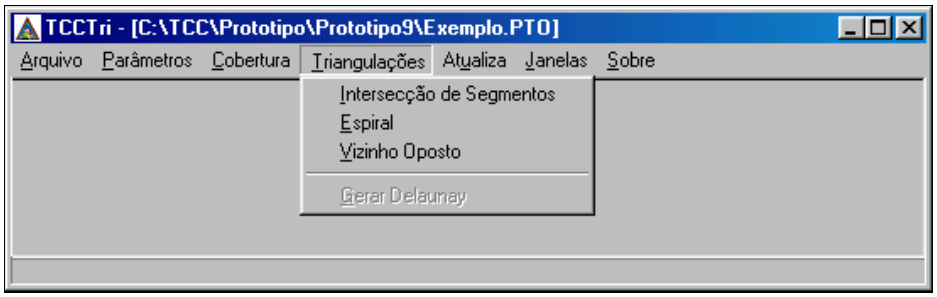
FIGURA 38 – Opção Cobertura




Em **Triangulações** (FIGURA 39) aparecerá: **Intersecção de Segmentos**, **Espiral** e **Vizinho Oposto** que são os métodos de triangularizações estudados na seção 3.2. Após clicar

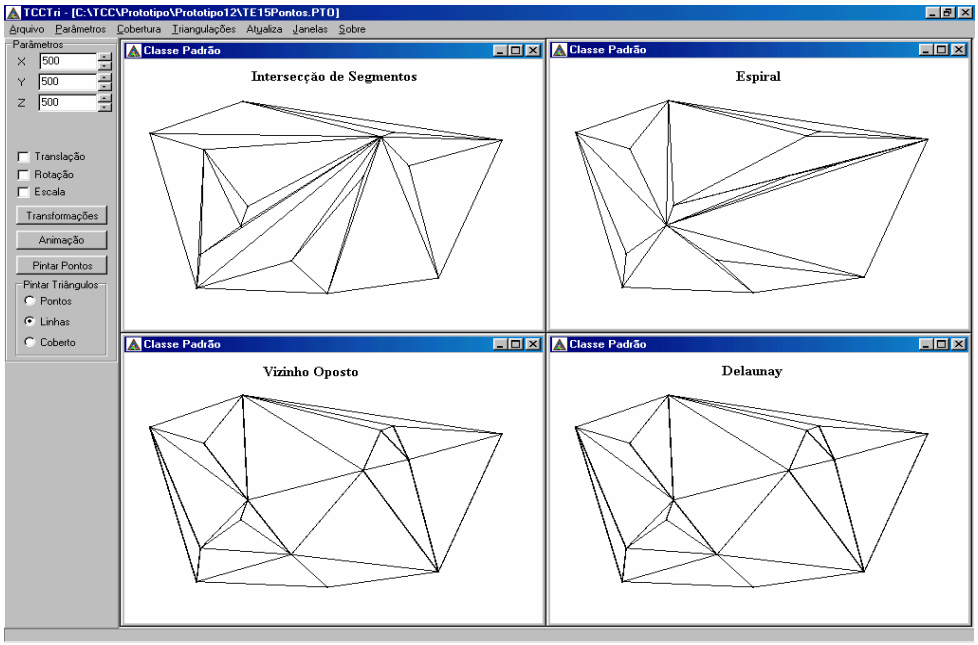
em qualquer umas das opções será apresentada a respectiva triangularização selecionada; ao término do cálculo será apresentada a superfície triangularizada e a opção **Gerar Delaunay** será habilitada, caso se venha a clicar nesta opção será apresentada a superfície otimizada.

FIGURA 39 – Opção Triangulações




A FIGURA 40 contém o resultado de cada um dos métodos de triangularização e uma otimização estudados neste trabalho.

FIGURA 40 – Triangulações




No menu a opção **Atualiza** conforme a FIGURA 41 exibirá as opções **Ativa** (F5) e **Todas**, serve, respectivamente, para atualizar o desenho da janela selecionada (F5) ou atualizar todas as janelas que o protótipo apresenta.

FIGURA 41 – Atualiza


Na opção **Janelas** (FIGURA 42) são apresentadas as opções de layout das janelas que são criadas pelo protótipo as opções **Cascata** e **Lado à Lado** irão apresentar, respectivamente, a FIGURA 43 e a FIGURA 44. A opção **Fechar Todas**, irá fechar todas as janelas geradas pelo protótipo.

FIGURA 42 – Janelas

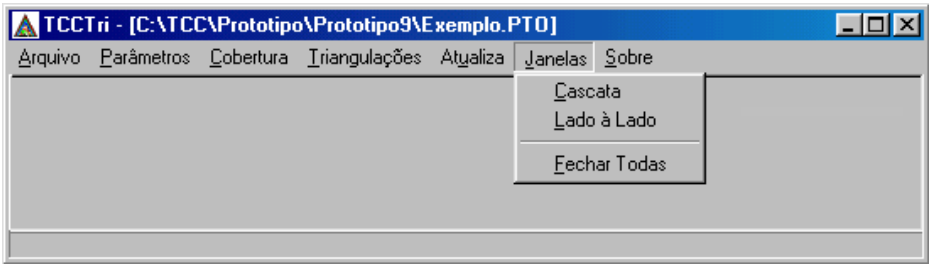


FIGURA 43 – Cascata

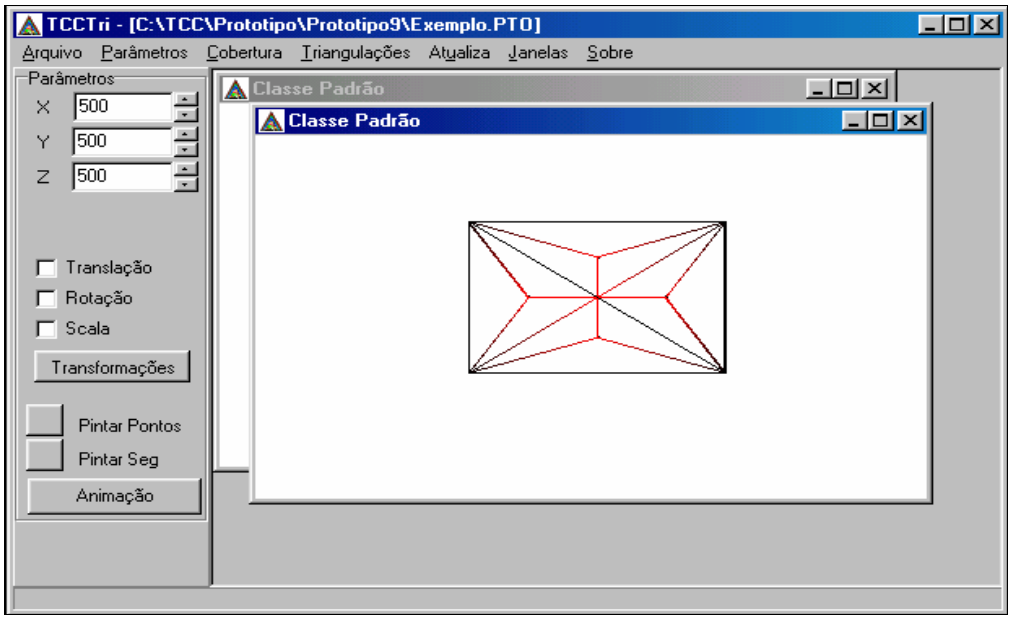
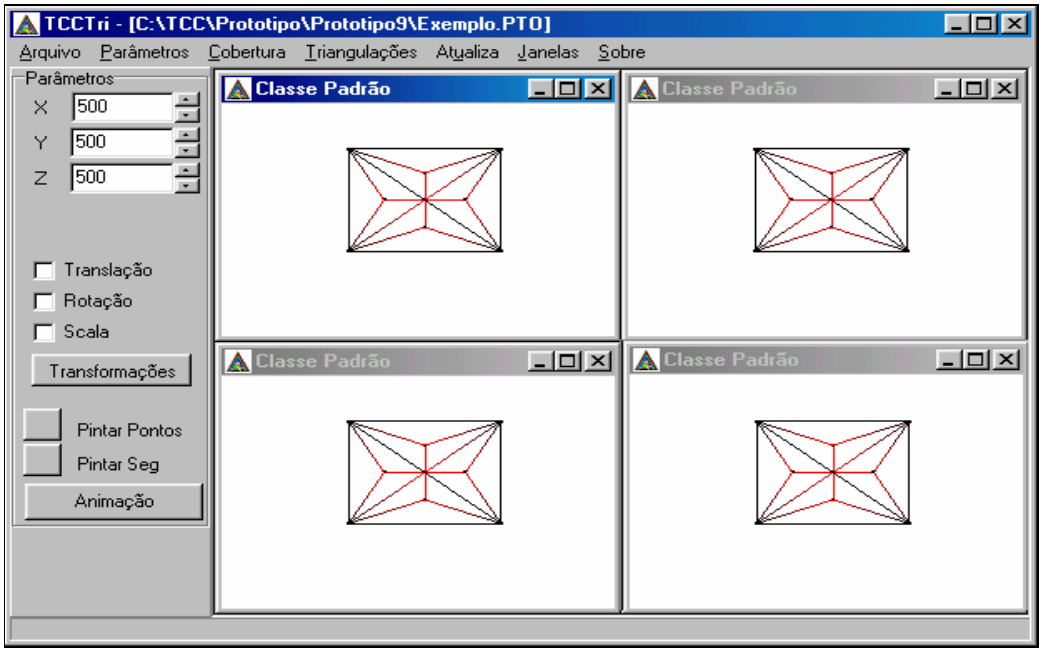
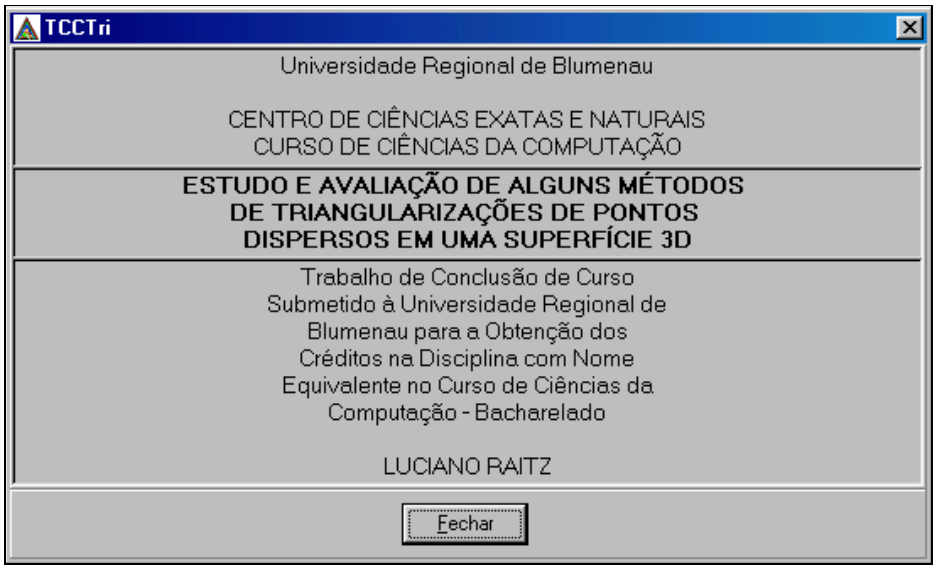



FIGURA 44 – Lado à Lado



A opção Sobre, não possui nenhuma funcionalidade para o sistema. Simplesmente apresenta uma tela com informações gerais como detalhes do curso e título do trabalho feito (FIGURA 45).

FIGURA 45 – Janela Sobre



# 5 CONSIDERAÇÕES FINAIS

A seguir apresentam-se a análise dos resultados, conclusão, limitações e sugestões para trabalhos futuros.

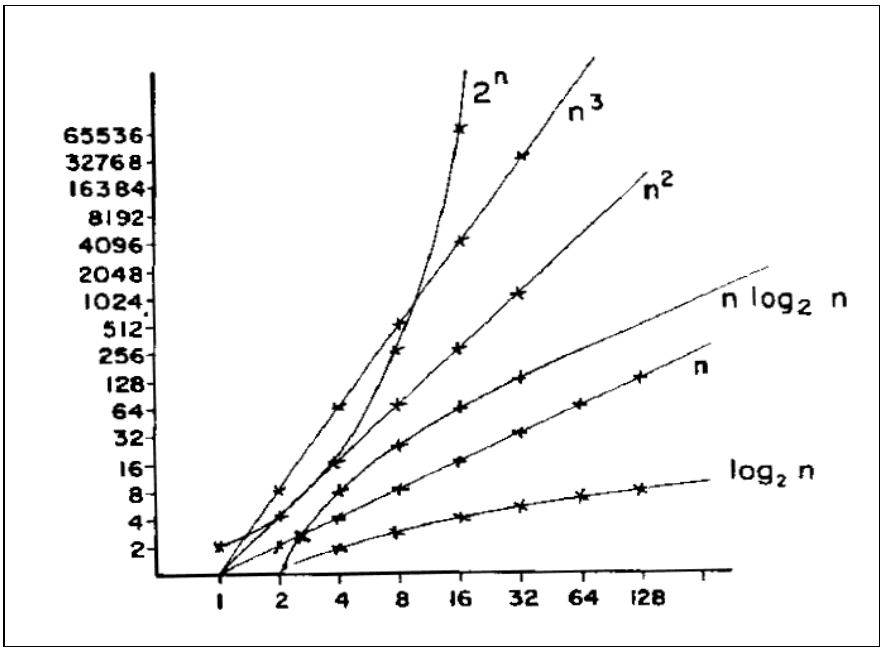
## 5.1 ANÁLISE DOS RESULTADOS

Segundo Reis (1997), conceitualmente, percebe-se que não existe grande dificuldade na construção manual de uma triangularização, a não ser pelo fato de se ter grandes quantidades de pontos dispersos tornando o procedimento cansativo e propenso a erros (devido a aspectos ergonômicos, fadiga, etc.). Ao utilizar procedimentos computacionais, tem-se a vantagem de se tratar de um processo automatizado e menos propenso a erros, tendo-se somente o custo da implementação das rotinas necessárias a este procedimento. Desta forma, poder-se-ia questionar o quanto estes procedimentos computacionais são eficientes e próximos da representação real.

Para avaliar os algorítmos estudados anteriormente no capítulo 4, foi utilizado o fator da ordem de complexidade teórica, a fim de comparar a eficiência de um algoritmo em relação aos outros.

A ordem de complexidade é um fator que está relacionado com a velocidade de execução do algoritmo, verificando-se o seu comportamento em relação ao tempo para diferentes tamanhos e valores de entrada (Azeredo, 1996). No presente estudo os algoritmos possuem seus valores de entrada somente como pontos distribuídos aleatoriamente no plano *x*, *y* e a elevação (coordenada *z*) determinando as inclinações de uma superfície específica. As taxas de crescimento das funções de tempo de computação mais comuns podem ser verificadas na FIGURA 46, ($\log_2 n$, $n$, $n \log_2 n$, $n^2$, $n^3$, $2^n$) (Horowitz, 1984).

FIGURA 46 – Taxa de crescimento das funções de tempo




## 5.1.1 AVALIAÇÃO DOS ALGORITMOS

A análise dos algoritmos pode ser teórica ou experimental. A abordagem teórica, segundo Azeredo (1996), permite uma análise independente da implementação, estimando-se analiticamente as quantidades de operações que o algoritmo deve efetuar diante de diferentes volumes e valores de entrada. O resultado da análise é uma função *f*, real, não-negativa, da variável $n \geq 0$ que representa a quantidade de pontos dispersos (Azeredo, 1996).

Ao se analisar sob o aspecto teórico o algoritmo do tipo divisão e conquista (Pesquisa dos Cantos) (Toscani, 1986), considerou-se que os subconjuntos eram particionados aproximadamente ao meio. Já uma análise experimental, segundo Azeredo (1996), submete a implementação dos algoritmos a um variado e representativo conjunto de valores para permitir o monitoramento da sua execução.

Este trabalho realiza apenas o estudo teórico dos algorítmos implementados e, desta forma, apresentará tabelas ou curvas que indicam a variação de seu comportamento em função dos diversos tipos de entradas.

Um algoritmo pode ser analisado levando-se em consideração o melhor caso, o pior caso e o caso médio. Na análise teórica, optou-se pelo estudo do pior caso levando-se em conta interesses práticos a fim de verificar o desempenho máximo exigido por um algoritmo, pois, mesmo que raro na prática, este caso ainda pode ocorrer (Azeredo, 1996).

Pode-se calcular as quantidades de triângulos e lados através das equações:

$$n = c + i$$
$$t = c + 2\,(i-1) \quad \leq 2n$$
$$t = 2\,(n-1) - c$$
$$l = 2\,c + 3\,(i-1) \quad \leq 3n$$
$$l = 3(n-1) - c$$

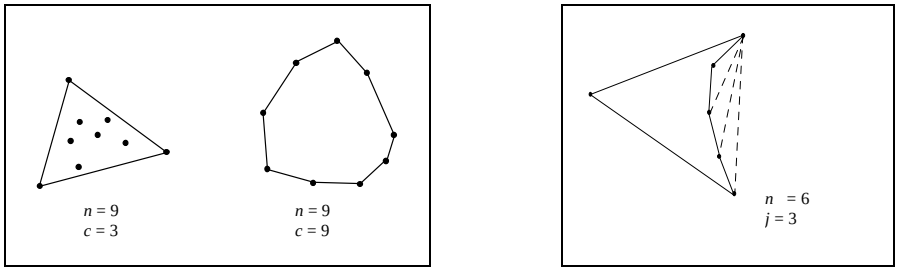
onde,

*n*: número de pontos dispersos;
*c*: número de pontos da cobertura convexa;
*i*: número de pontos internos;
*t*: número de triângulos;
*l*: número de lados.

Mesmo existindo muitas maneiras diferentes de se triangularizar os pontos dispersos, todas as possíveis triangularizações têm o mesmo número de triângulos e de lados (Lawson, 1977).

Em relação à quantidade de pontos que uma cobertura convexa (*c*) pode ter, verifica-se claramente (FIGURA 47-a) que no mínimo são 3 pontos e no máximo o total de pontos dispersos (*n*). Já quanto ao número de possíveis concavidades (*j*), formadas por pontos da cobertura não pertencentes à cobertura convexa, pode ter-se no máximo o número de pontos dispersos menos três pontos (FIGURA 47-b).

FIGURA 47 - a) Pontos na cobertura convexa - b) Número de concavidades




Fonte: Reis, 1997.

Ao se conhecer os números mínimo e máximo de pontos da cobertura convexa, pode-se calcular as quantidades de triângulos e lados através das equações desta seção. Com uma cobertura mínima, o número de triângulos é de 2n - 5 (n = número de pontos dispersos) e 3n - 6 lados. Para uma cobertura máxima tem-se n - 2 triângulos e 2n - 3 lados. Nota-se que, com a quantidade de pontos acima de 3, a cobertura mínima atinge sempre valores maiores em triângulos e lados.

### 5.1.2 ORDEM DE COMPLEXIDADE TEÓRICA

Nos processos de triangularização que necessitam da definição dos pontos do conjunto ***V*** da cobertura convexa, utilizou-se o algoritmo da seção 3.1. A análise aqui considerada refere-se ao pior caso em relação aos valores de entrada no algoritmo.

Nas tabelas seguintes, têm-se os valores colocados em ordem crescente de complexidade. A fim de quantificar o crescimento das funções de ordem de complexidade dos algoritmos, utilizaram-se conjuntos de 100, 200, 500 e 1000 pontos como valores de entrada.

Um estudo detalhado das análises de ordem de complexidade teórica dos algoritmos encontra-se no anexo 5.

O algoritmo de cobertura convexa apresenta a função de ordem de complexidade mostrada na TABELA 1.

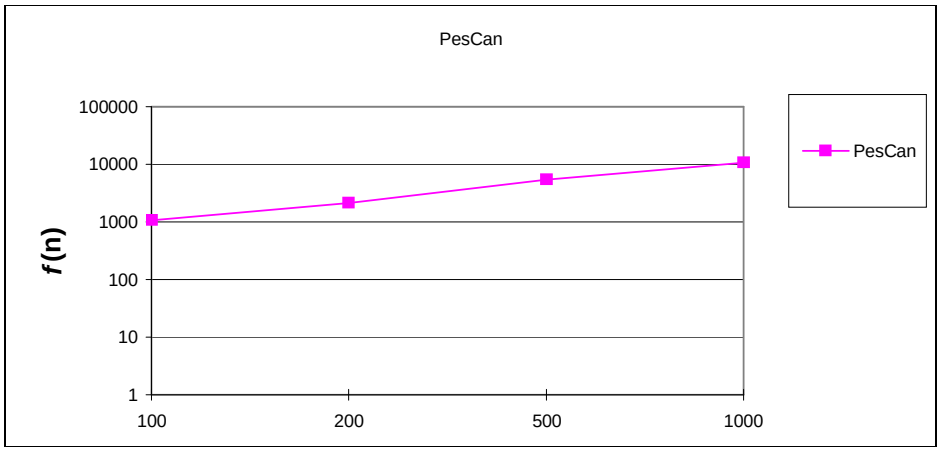
TABELA 1 - Complexidade teórica - cobertura convexa

| Cobertura convexa | | |
|---|---|---|
| Algoritmo | função - $f(n)$ | ordem complexidade |
| 1- Pesquisa dos Cantos | $11n - 8\log_2(n+4) + 16$ | $n$ |

Como o valor do custo do algoritmo (TABELA 1) é muito variável, devido as entradas, optou-se por fazer uma representação gráfica através de escala logarítmica. Desta forma, as diferenças ficam mais visíveis (FIGURA 48).

O algoritmo Pesquisa dos Cantos, utiliza a abordagem do tipo divisão e conquista (Toscani, 1986). Também apresenta um baixo custo, mas tende a aumentar a diferença de sua curva de acordo com o aumento do número de pontos dispersos.

FIGURA 48 - Complexidade teórica - cobertura - ponto




A rotina de pesquisa dos cantos, utilizada somente no algoritmo Pesquisa dos Cantos (custo igual a *n*), possibilita reduzir o conjunto de pontos a serem processados pela rotina de pesquisa dos pontos mais afastados (figura 5.1 do anexo 5). Já a rotina pesquisa dos pontos mais afastados é implementada de uma forma diferente no algoritmo Pesquisa dos Cantos, onde efetua-se somente esta pesquisa num dos subconjuntos obtido pelo processo divisão dos pontos. Desta forma, estas diferenças tendem a diminuir a quantidade de rotinas recursivas (pontos mais afastados) e o custo associado a esta rotina, proporcionando assim um melhor resultado no custo final do pior caso ao algoritmo Pesquisa dos Cantos.

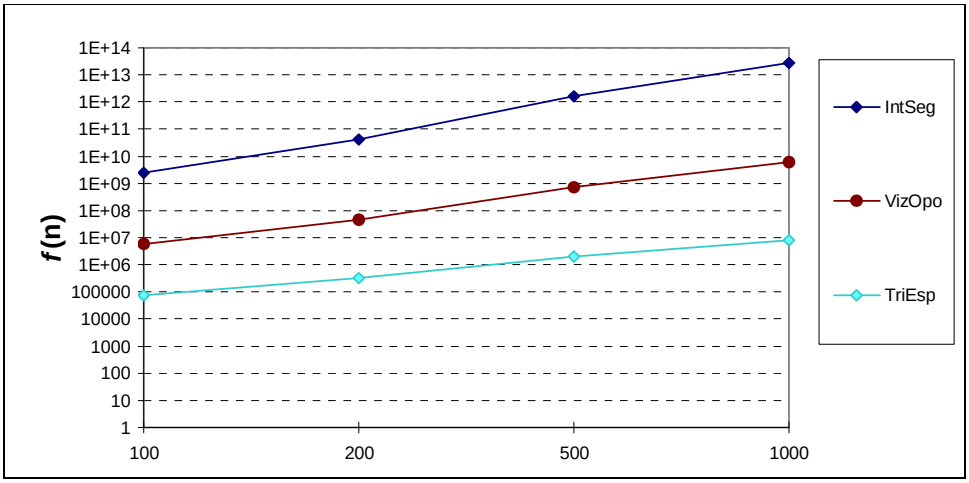
Na análise dos algoritmos de triangularização, obtêm-se as funções e ordem de complexidade da TABELA 2.

TABELA 2 - Complexidade teórica - triangularização

| Triangularização | | |
|---|---|---|
| Algoritmo | função - $f(n)$ | $O(n)$ |
| 1- Triangularização Espiral | $O_f(n) = 8n^2 - 19n - 8\log_2(n+4) + 46$ | $n^2$ |
| 2- Pesquisa do Vizinho Oposto | $O_f(n) = 6n^3 - 15n^2 - 22n + 60$ | $n^3$ |
| 3- Intersecção de Segmentos | $O_f(n) = 27n^4 - 186n^3 + 429n^2 - 330n$ | $n^4$ |

A representação gráfica (escala logarítmica) dos custos dos algoritmos de triangularização (TABELA 2) pode ser observado na FIGURA 49.

FIGURA 49 - Complexidade teórica - triangularização – pontos




Na análise das curvas de custos dos algoritmos de triangularização (FIGURA 49), verificaram-se comportamentos distintos entre os algoritmos.

O algoritmo de Triangularização Espiral (anexo 5.4), obteve o menor custo apesar de ser elevado. Este custo elevado ocorre devido às pesquisas repetitivas estarem relacionadas a vários conjuntos e não a um subconjunto.

O algoritmo de custo médio estudado neste trabalho é o de Pesquisa do Vizinho Oposto, o custo tornou-se elevado pois este algoritmo gera uma triangularização Delaunay, e o outro compreende o método de triangularização mais simples implementado (Intersecção de Segmentos). Nestes dois algoritmos o custo elevado é resultado de pesquisas ordenadas repetitivas entre os pontos $P_i$ do conjunto ***P***. Estas pesquisas são produtos de combinações entre os próprios pontos $P_i$, resultando, respectivamente, em custos de ordens $O(n) = n^3$ e $O(n) = n^4$.

A otimização dos triângulos, a qual tem por objetivo gerar uma triangularização Delaunay (seção 3.3.1), possui um custo de $O(n) = 84n^2 - 420\,n + 525$ quando a cobertura convexa possui 3 pontos (anexo 5.5), o pior caso. Este custo representa um “ciclo” de otimização, ou seja, aplica-se um critério de otimização (seção 3.3.1) a cada par de triângulos adjacentes. Caso ainda se encontre um triângulo não otimizado, geram-se novos ciclos.

Como objetiva-se ter uma triangularização Delaunay dos pontos dispersos, calculou-se o custo de cada algoritmo até atingir esta triangularização otimizada. Este cálculo foi obtido pela soma de cada custo de triangularização mais o custo da otimização (TABELA 3), exceto no algoritmo de Pesquisa do Vizinho Oposto que não necessita de um processo de otimização.

TABELA 3 - Complexidade teórica - triangularização otimizada

| Triangularização | | |
|---|---|---|
| Algoritmo | função - $f(n)$ | $O(n)$ |
| 1- Triangularização Espiral | $92n^2 - 439n - 8\log_2(n+4) + 571$ | $n^2$ |
| 2- Pesquisa do Vizinho Oposto | $6n^3 - 15n^2 - 22n + 60$ | $n^3$ |
| 3- Intersecção de Segmentos | $27n^4 - 186n^3 + 513n^2 - 750n + 525$ | $n^4$ |

Como o custo do processo de otimização no pior caso é igual ao de todos os algoritmos que utilizam este processo, era de se esperar que as ordens de custos fossem iguais às da TABELA 2. Pode-se verificar que o custo de otimização adicionado aos custos dos algoritmos que necessitam deste processo, para gerar uma triangularização Delaunay, não interfere na classificação feita na TABELA 2, pois o algoritmo Pesquisa do Vizinho Oposto mantêm sua posição. Isto ocorre porque a ordem, no algoritmo de triangularização Delaunay, foi $O(n) = n^3$ e, no processo de otimização, $O(n) = n^2$.

Isto ocorreu devido ao fator de que a mais alta ordem, no processo de otimização, ser $O(n) = n^2$ e, nos algoritmos de triangularização Delaunay, terem a ordem $O(n) = n^3$.

Os valores da TABELA 3 não foram graficados, já que as curvas desta tabela têm comportamentos em relação à ordem de custo iguais aos da FIGURA 49.

## 5.2 CONCLUSÃO

A pesquisa dos conceitos estudados viabilizou a geração de superfícies em três dimensões, que foram construídas por algoritmos de triangularização formando uma malha triangular, através da implementação de um protótipo.

Pode também ser identificado a diferença entre os métodos de triangularização, desde sua dificuldade na implementação, quanto no resultado final da disposição de seus triângulos. Que após a otimização apresentam o mesmo resultado.

Também verificou-se que a malha triangular gerada pode ser utilizada para o reconhecimento de regiões, bastando passar as coordenadas dos pontos da superfície a ser estudada.

O ambiente de desenvolvimento Delphi 5 junto com a biblioteca gráfica OpenGL mostraram-se eficientes no que diz respeito a oferecerem recursos para a visualização de desenhos em três dimensões.

## 5.3 LIMITAÇÕES

Os algorítmos de triangularização implementados neste trabalho, não abordam superfícies côncavas, ou seja, superfícies que possuem cavernas; caso pontos simulando cavernas venham a ser triangularizados, o protótipo irá tratar esta situação como se fosse um plano qualquer. Pontos de mesma coordenada x e y serão ignorados pelo protótipo.

## 5.4 EXTENSÕES

Como extensão sugere-se para a área de pesquisa de trabalhos futuros, o uso de outros métodos para se gerar a triangularização de uma superfície através de pontos dispersos. Realizar a análise experimental dos algorítmos para se fazer testes de memória e comparar a imagem gerada com a superfície real.

Implementar algorítmo de triangularização que não haja restrição de cavernas.

Comparar a superfície gerada com a superfície real, observando o aspecto visual, utilizando a representação em perspectiva num ambiente de câmera sintética.

Fazer o estudo do Diagramas de Voronoi para a realizar a triangularização.

# ANEXO 1: POSIÇÃO DO PONTO EM RELAÇÃO AO SEGMENTO

```
function TPontos.PosPonto(POrigem, PAtual, PDestino :TRegPontos): Double;
begin  // Sobre = 0  -  Direita > 0  -  Esquerda < 0
  PosPonto:=(((PAtual.X-POrigem.X)*(PDestino.Y-POrigem.Y))-((PAtual.Y-
POrigem.Y)*(PDestino.X-POrigem.X)));
end;
```

# ANEXO 2: ORIENTAÇÃO DOS TRIÂNGULOS

```
function TPontos.PosPontosTri(PriX,PriY, SegX,SegY, TerX,TerY :Double):
Double;
begin     // Colinear = 0  -  Sent Horário > 0  -  Anti-Horário < 0
  PosPontosTri:=((PriX * SegY) + (PriY * TerX) + (SegX * TerY) -
                 (TerX * SegY) - (TerY * PriX) - (SegX * PriY));
end;
```

# Anexo 3: ALGORITMO DE INTERSECÇÃO DE SEGMENTOS.

```
Function TTriInterSeg.Intersec(Pt, P1I, P1F, P2I, P2F: TPontoXY): boolean;
Var
  Min1, min2, max1, max2: Double;
Begin
  Min1 := min(P1I.X, P1F.X);
  Max1 := max(P1I.X, P1F.X);
  Min2 := min(P2I.X, P2F.X);
  Max2 := max(P2I.X, P2F.X);
  If   ((min1- max1) = 0)
  Or   ((min2- max2) = 0) then
       Begin
         min1 := min(P1I.Y, P1F.Y);   max1 := max(P1I.Y, P1F.Y);
         min2 := min(P2I.Y, P2F.Y);   max2 := max(P2I.Y, P2F.Y);
         if   (min1 = max1) then
              begin
                min1 := min(P1I.X, P1F.X);
                max1 := max(P1I.X, P1F.X);
                if   (Pt.X > Min1) and  (Pt.X < Max1)
                and  (Pt.Y > min2) and (Pt.Y < max2) then
                     result:=true
                else result:=False;
              end;
         if   (min2 = max2) then
              begin
                min2 := min(P2I.X, P2F.X);   max2 := max(P2I.X, P2F.X);
```

```
                if   (Pt.X > Min2) and (Pt.X < Max2)
                and  (Pt.Y > min1) and (Pt.Y < max1) then
                     result:=true
                else result:=false;
              end;
          if   (Pt.Y > Min1) and (Pt.Y < Max1)
          and  (Pt.Y > min2) and (Pt.Y < max2) then
               result:=true
          else result:=false;
        end;
  if  (FormatFloat('00000000.0000',Pt.X)>FormatFloat('00000000.0000',Min1))
  and (FormatFloat('00000000.0000',Pt.X)<FormatFloat('00000000.0000',Max1))
  and (FormatFloat('00000000.0000',Pt.X)>FormatFloat('00000000.0000',min2))
  and (FormatFloat('00000000.0000',Pt.X)<FormatFloat('00000000.0000',max2))
  then
       result:= true
  else result:= false;
end;

function TTriInterSeg.PtoInterSeg(P1I, P1F, P2I, P2F: TPontoXY; var retPar:
Byte): Boolean;
var
  a1, b1, a2, b2, den1, den2: Double;
  Pint: TPontoXY;
Begin
  den1 := (P1F.X - P1I.X);  den2 := (P2F.X - P2I.X);
  if   (den1 = 0) and (den2 = 0) then
       begin
         Result:= false;    retPAR:= 1;
       end;//Retas paralelas x constantes
  if   den1 <> 0 then
       begin
         a1 := (P1F.Y - P1I.Y)/den1;   b1 := P1I.Y - (a1 * P1I.X);
       end;
  if   den2 <> 0 then
       begin
         a2 := (P2F.Y - P2I.Y)/den2;   b2 := P2I.Y - (a2 * P2I.X);
       end;
  if   (a1 - a2) = 0 then
       begin
         Result:= false;    retPAR:= 2;
       end;//Retas paralelas
  if   (den1 = 0) then
       begin
         Pint.X:= P1I.X;
         den1:= min(P2I.X, P2F.X);  den1:= a2 * (P1I.X- den1);
         if   (a2 > 0) then
              begin
                den2:= min(P2I.Y, P2F.Y);  Pint.Y:= den2+den1;
              End
         else begin
                den2:= max(P2I.Y, P2F.Y);  Pint.Y:= den2+den1;
              end;
         if   Intersec(Pint, P1I, P1F, P2I, P2F) then
              result:= true
         else result:= false;
```

```
      end;
  if  (den2 = 0) then
      begin
        Pint.X:= P2I.X;
        den2:= min(P1I.X, P1F.X);  den2:= a1 * (P2I.X- den2);
        if   (a1 > 0) then
             begin
               den1:= min(P1I.Y, P1F.Y);    Pint.Y:= den2+den1;
             End
        else begin
               den1:= max(P1I.Y, P1F.Y);    Pint.Y:= den2+den1;
             end;
        if   Intersec(Pint, P1I, P1F, P2I, P2F) then
             result:= true
        else result:= false;
      end;
  Pint.X:= (b2-b1)/(a1-a2);
  Pint.Y:= (a1 * Pint.X) + b1;
  if   Intersec(Pint, P1I, P1F, P2I, P2F) then
       result:= true
  else result:= false;
end;
```

# Anexo 4: PONTO INTERNO AO TRIÂNGULO.

```
function TTriangulo.PtoInternoTri(PtoA, PtoB, PtoC, PtoXY: TRegPontos):
Boolean;
var
  Positivo:Boolean;
  ProdVet :Double;
Begin
  Positivo:=False;
  ProdVet:=0;
  if(((ptoA.x = ptoXY.x) and (ptoA.y = ptoXY.y)) or
     ((ptoB.x = ptoXY.x) and (ptoB.y = ptoXY.y)) or
     ((ptoC.x = ptoXY.x) and (ptoC.y = ptoXY.y)))Then
  Begin
    Result:=TRUE;exit;
  End;
  ProdVet := PosPonto(ptoB, ptoXY, ptoA);
  If(ProdVet > 0) Then Positivo := TRUE;
  ProdVet := PosPonto(ptoC, ptoXY, ptoB);
  if ((Positivo)and(ProdVet >= 0) Or (Not Positivo)and(ProdVet <= 0)) Then
      Begin
        ProdVet := PosPonto(ptoA, ptoXY, ptoC);
        if ((Positivo) and (ProdVet >= 0)
        or  (Not Positivo) and (ProdVet <= 0)) Then
                  Result:=TRUE
          else    result:=FALSE;
      End
      else result:=FALSE;
end;
```

# ANEXO 5: DEFINIÇÃO DAS FUNÇÕES DE ORDEM DE COMPLEXIDADE TEÓRICA

A seguir, têm-se as descrições das funções de ordem de complexidade teórica, onde *n* indica o número de pontos dispersos, *t*, o número de triângulos, *l*, o número total de lados, *c*, o número de pontos na cobertura convexa e *j*, o número de concavidades em uma cobertura convexa (FIGURA 47). Em quase todas as análises, utiliza-se a igualdade $\sum_{i=0}^{q} 2^i = 2^{q+1} - 1$ (1) que é a soma dos $q + 1$ termos de uma progressão geométrica de razão 2 com $a_0 = 1$.

Utiliza-se também indução matemática na definição das funções associadas às subárvores.

Para mostrar que uma propriedade ( $P$ ) é válida, para $\forall n \in \mathrm{N} \rightarrow \begin{matrix} \{0,1,2,\ldots\} \\ \{1,2,3,\ldots\}* \end{matrix}$ (2) deve-se ter:

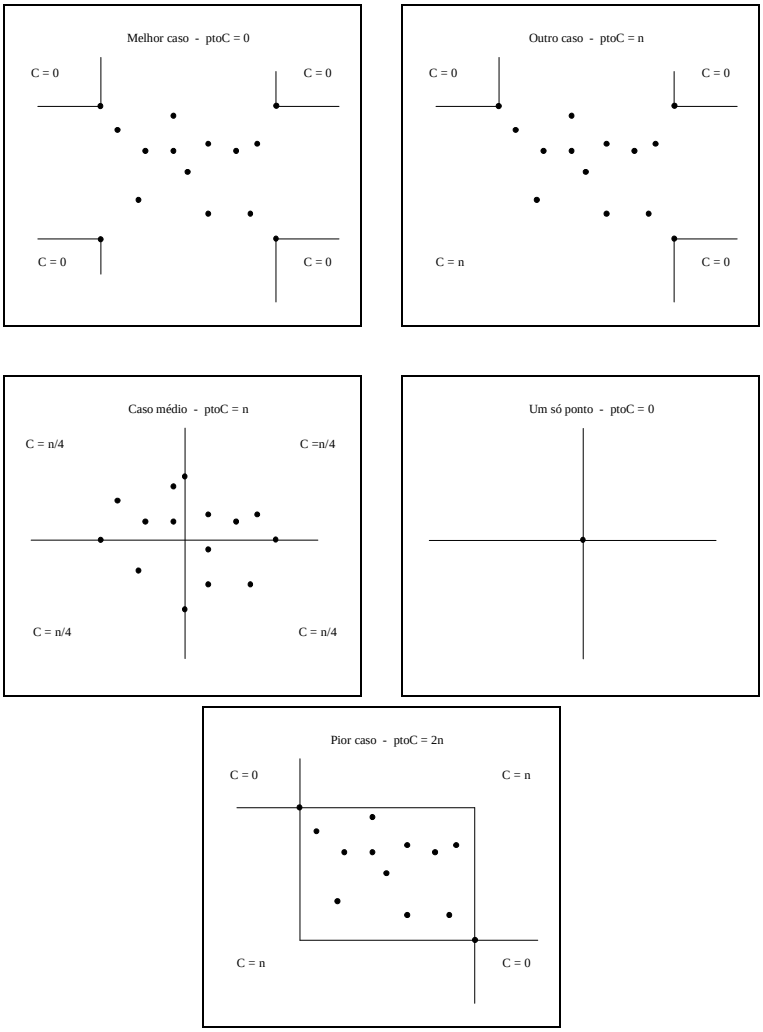
i) $P(1)$ é verdadeiro;

ii) assumindo que $P(n)$ é verdadeiro, provar que $P(n+1)$ também é verdadeiro.

## ANEXO 5.1 PESQUISA DOS CANTOS

Neste algoritmo considera-se o cálculo da ordem de complexidade em duas partes, onde tem-se um custo até a rotina de Pesquisa dos Cantos e outro após esta etapa. Entre os possíveis subconjuntos de pontos (figura anexo 5-1) que a rotina Pesquisa dos Cantos pode formar optou-se pelo pior caso, ou seja, aquele que em média resulta o maior número de pontos a cada canto pesquisado. Desta forma se o pior caso contém $2n$ pontos têm-se $\frac{2n}{4} = \frac{n}{2}$ pontos para cada canto pesquisado.

FIGURA Anexo 5-1 - Pontos em Pesquisa dos Cantos




Fonte: Reis, 1997.

Para se calcular o custo teórico no pior caso deste algoritmo, verifica-se o comportamento de cada rotina em relação ao número de pontos dispersos $n$.

Inicialmente tem-se uma pesquisa de todos os pontos $P_i$ do conjunto ***P*** para verificar os pontos mais à esquerda e mais à direita da coordenada x, representando um custo igual a $n$. Em seguida, definem-se os pontos para os respectivos cantos através da rotina pesquisa dos cantos, anteriormente demonstrada. Após estas pesquisas iniciais, seguem-se dois processos, divisão dos pontos e pesquisa do ponto mais afastado, os quais são repetidos sucessivamente

até que cada subconjunto de ***P*** tenha no máximo um ponto. Isto significa a cada repetição dividir por 2 o conjunto e de cada subtrair 1 ponto.

Isto pode ser representado esquematicamente como:

$$
\begin{array}{lll}
 & n & \text{pesq. min / max x e y} \\
 & / \quad / \quad \backslash \quad \backslash & \\
 & n \quad n \quad n \quad n & \text{pesquisar os cantos} \\
---\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!- & | \quad | \quad | \quad | & \\
\text{serie } S(n) & \frac{n}{2} \quad \frac{n}{2} \quad \frac{n}{2} \quad \frac{n}{2} & \text{dividir os pontos} \\
 & / \backslash \quad / \backslash \quad / \backslash \quad / \backslash & \\
 & \frac{n}{4} \ \phi \quad \frac{n}{4} \ \phi \quad \frac{n}{4} \ \phi \quad \frac{n}{4} \ \phi & \text{ponto mais afastado} \\
 & | & \\
 & \left(\frac{n}{4}\right)-1 & \text{dividir os pontos} \\
 & / \quad \backslash & \\
 & \left(\frac{\left(\frac{n}{4}\right)-1}{2}\right) \quad \phi & \text{ponto mais afastado}
\end{array}
\qquad (3),
$$

onde ϕ indica que o processo pesquisa do ponto mais afastado é feita somente a um dos subconjuntos obtidos pelo processo divisão dos pontos.

Percebe-se, exceto pelas duas primeiras rotinas, pesquisa de todos os pontos e pesquisa dos cantos, que os próximos níveis de execução possuem um comportamento padrão permitindo definir uma série $S(n)$ associada a este comportamento.

$$S(n) = 4\frac{n}{2} + 4\frac{n}{4} + 4\left(\frac{n}{4}-1\right) + 4\left(\frac{\left(\frac{n}{4}-1\right)}{2}\right) + 4\left(\left(\frac{\left(\frac{n}{4}-1\right)}{2}\right)-1\right) + \ldots \quad \text{ou}$$

$$S(n) = 2n + n + (n-4) + (n-2) + \left(\frac{n}{2}-6\right) + \ldots \quad .$$

Para calcular o custo expresso pela série $S(n)$, simplifica-se esta série a um ramo da árvore *A*. Desta forma, após obter o custo para esta série simplificada, deve-se multiplicar este custo por 4. Têm-se então a série $S(n)$ e a árvore *A* simplificada:

$$S(n)=\frac{n}{2}+\frac{n}{4}+\frac{n-4}{4}+\frac{n-4}{8}+\frac{n-12}{8}+\frac{n-12}{16}+\ldots \quad \text{e}$$

$$\begin{array}{lll}
q=3 & \frac{n}{2} & \text{dividir os pontos} \\
 & | & \\
 & \frac{n}{4} & \text{ponto mais afastado} \\
 & | & \\
q=2 & \left(\frac{n}{4}-1\right) & \text{dividir os pontos} \\
 & | & \\
 & \left(\frac{\left(\frac{n}{4}-1\right)}{2}\right) & \text{ponto mais afastado} \\
 & | & \\
q=1 & \left(\left(\frac{\left(\frac{n}{4}-1\right)}{2}\right)-1\right) & \text{dividir os pontos} \\
 & | & \\
 & \left(\frac{\left(\left(\frac{\left(\frac{n}{4}-1\right)}{2}\right)-1\right)}{2}\right) & \text{ponto mais afastado}
\end{array} \qquad (4).$$

Com a definição da árvore *A* (4), quantifica-se cada valor dos nós desta árvore iniciando-se com o valor 1 numa folha da árvore *A*. Este é o valor que indica o término do algoritmo. Tem-se:

$$\begin{array}{llll}
q=4 & 30 & \left(2\left(\left(2\left(\left(2\left(\left(2\times 1\right)+1\right)\right)+1\right)\right)+1\right)\right) & = 2^4+2^3+2^2+2^1 \\
 & | & & \\
 & 15 & \left(\left(2\left(\left(2\left(\left(2\times 1\right)+1\right)\right)+1\right)\right)+1\right) & = 2^3+2^2+2^1+2^0 \\
 & | & & \\
q=3 & 14 & \left(2\left(\left(2\left(\left(2\times 1\right)+1\right)\right)+1\right)\right) & = 2^3+2^2+2^1 \\
 & | & & \\
 & 7 & \left(\left(2\left(\left(2\times 1\right)+1\right)\right)+1\right) & = 2^2+2^1+2^0 \\
 & | & & \\
q=2 & 6 & \left(2\left(\left(2\times 1\right)+1\right)\right) & = 2^2+2^1 \\
 & | & & \\
 & 3 & \left(\left(2\times 1\right)+1\right) & = 2^1+2^0 \\
 & | & & \\
q=1 & 2 & \left(2\times 1\right) & = 2^1 \\
 & | & & \\
 & 1 & 1 & = 2^0
\end{array} \qquad (5).$$

As subárvores da árvore *A* são definidas pelos processos divisão dos pontos e pesquisa do ponto mais afastado (4). Desta forma, quantifica-se algebricamente uma sub-árvore $A_S$ e o seu relacionamento com as outras subárvores de *A*. Este relacionamento entre subárvores é feito por q indicando o nível de uma subárvore, com base em (4):

$$\left(\frac{n}{4}-1\right)$$
$$|$$
$$\frac{\left(\frac{n}{4}-1\right)}{2}$$

para o nível q = 1, tem-se (5):

$$2$$
$$| \quad \text{ou}$$
$$1$$
$$\left(2^{q+1}-2\right)$$
$$| \quad \text{com q = 1,}$$
$$\left(\frac{2^{q+1}-2}{2}\right)$$

ou seja, o custo $A_S$ da subárvore é

$$A_S = 2^{q+1} - 2 + \left(\frac{2^{q+1}-2}{2}\right) = \frac{3}{2}\left(2^{q+1}-2\right) \qquad (6).$$

Para certificar que a função $A_S$ (6), associada à subárvore do nível q = 1, representa o relacionamento com todas as subárvores de *A* (5), usa-se indução matemática.

Com a definição da função $A_S$ (6) em relação a cada nível q (5), têm-se:

$q = 1 \Rightarrow 1$ subárvore $\rightarrow 1\times\left(\frac{3}{2}\left(2^{1+1}-2\right)\right)$ ,

$q = 2 \Rightarrow 1$ subárvore $\rightarrow 1\times\left(\frac{3}{2}\left(2^{2+1}-2\right)\right)$ ,

$q = 3 \Rightarrow 1$ subárvore $\rightarrow 1\times\left(\frac{3}{2}\left(2^{3+1}-2\right)\right)$ ,

...

$q = K \Rightarrow 1$ subárvore $\rightarrow 1\times\left(\frac{3}{2}\left(2^{K+1}-2\right)\right)$ (7).

Somando-se todas as funções $A_S$ (6) das respectivas subárvores de cada nível de acordo com (7), tem-se a seguinte série, que é a soma de todos custos em função do nível K.

$$S_T(K) = \frac{3}{2}\left(2^2-2\right)+\frac{3}{2}\left(2^3-2\right)+\ \ldots+\frac{3}{2}\left(2^{K+1}-2\right) \quad \text{ou}$$

$$S_T(K) = \underbrace{\left(-2-2-\ldots-2\right)}_{\times Ks} + \frac{3}{2}\left(2^2+2^3+\ldots+2^{K+1}\right) \quad \text{ou}$$

$$S_T(K) = -2\times K + \frac{3}{2}\times 2^2 \times \underbrace{\left(2^0+2^1+2^2+\ldots+2^{K-1}\right)}_{2^{K-1}} \quad \text{ou}$$

$$S_T(K) = -2K+\frac{3}{2}\left(2^2\left(2^K-1\right)\right) \quad \text{ou}$$

$$S_T(K) = 3\times\left(2^{K+1}\right)-2K-6 \qquad (8).$$

O próximo passo é definir a função $S_T(K)$ em relação ao número de pontos *n*. Primeiramente obtêm-se expressões para $2^{K+1}$ e K.O número de elementos no topo é $\frac{n}{2}$.

Para isolar K de (8), tem-se de acordo com (1):

$$2^q+2^{q-1}+\ \ldots\ +2^1=\frac{n}{2} \quad \text{ou}$$

$$\underbrace{2\left(2^{q-1}+2^{q-2}+\ \ldots\ +2^0\right)}_{2^K-1}=\frac{n}{2} \quad \text{ou}$$

$$2\left(2^K-1\right)=\frac{n}{2} \quad \text{ou}$$

$$n = 2^2\left(2^K-1\right) \quad \text{ou} \qquad (9)$$

$$n = 2^{K+2}-4 \quad \text{ou}$$

$$\log_2(n+4) = \log_2 2^{K+2} \quad \text{ou}$$

$$K = \log_2(n+4)-2 \qquad (10).$$

Para isolar $2^{K+1}$ de (8), tem-se conforme (9):

$$n = 2^2\left(2^K-1\right) \quad \text{ou}$$

$$2\left(\frac{n}{4}+1\right) = 2^1 \times 2^K \quad \text{ou}$$

$$2^{K+1} = \frac{n}{2}+2 \qquad (11).$$

Aplicando (10) e (11) à expressão (8), e como

$$O(n) = S_T(K),$$

tem-se:

$$O(n) = \frac{3}{2}n - 2\log_2(n+4) + 4.$$

O custo acima exclui as duas primeiras rotinas, a pesquisa dos pontos mais à esquerda e mais à direita e pesquisa dos cantos (3). Portanto o custo final do algoritmo Pesquisa dos Cantos será:

$$O_f(n) = n + 4n + 4\left(\frac{3}{2}n - 2\log_2(n+4) + 4\right) \quad \text{ou}$$

$$O_f(n) = 11n - 8\log_2(n+4) + 16.$$

Considerando-se o termo de mais alta ordem, tem-se:

$$O_f(n) = n.$$

## ANEXO 5.2 INTERSECÇÃO DE SEGMENTOS

Para calcular o custo teórico no pior caso deste algoritmo, verifica-se o comportamento de cada rotina em relação ao número de pontos dispersos *n*. Este custo total foi de $O_f(n) = 27n^4 - 186n^3 + 429n^2 - 330n$ com os seguintes custos parciais:

1) $n(n-1)(l)$ gerar lados

$|$

2) $l(l-1)(l-2)\times n$ triangularizar

Seguem-se os detalhamentos destas partes:

[DSR1] Comentário:

**1) Custo da geração dos lados:**

Este custo representa a pesquisa ordenada aos pontos $P_i$ do conjunto ***P*** a fim de gerar o conjunto ***S*** que contém todas as combinações de segmentos. Estes segmentos são formados pelos pares de índices A e B, onde A se constitui de todos os pontos $P_i$ do conjunto ***P*** e B são todos os pontos do conjunto ***P*** que numa iteração anterior de pesquisa ainda não tenham sido um ponto A. A cada segmento novo de pesquisa, deve-se verificar se este não intersecciona um segmento já existente no conjunto ***S***. Desta forma, tem-se a seguinte função de custo:

$$O_1(n) = n\,(n-1)\,(l) \qquad (12).$$

Como

$$l = 3\,(n-1) - c \quad (13) \text{ (seção 3.1.1),}$$

visto no final da seção 3.2.1, e $c = n$ tem-se:

$$l = 2\,n - 3 \qquad (14)$$

que substituído em (12), dá:

$$O_{1a}(n) = 2\,n^3 - 5\,n^2 + 3\,n\,.$$

Se $l = 3\,(n-1) - c$ (13) e $c = 3$ tem-se:

$$l = 3n - 6\,(15)$$

que substituído em (12), dá:

$$O_{1b}(n) = 3\,n^3 - 9\,n^2 + 6\,n\,.$$

**2) Custo da geração dos triângulos:**

O custo da geração dos triângulos representa a pesquisa ordenada de todos os segmentos do conjunto ***S*** para gerar o conjunto ***T***, que contém combinações de índices de vértices que definem uma triangularização. Esta pesquisa verifica todas as combinações de segmentos do conjunto ***T*** e após pesquisa todos os pontos $P_i$ para formar os triângulos em ***T***. Portanto, conforme seção 4.2.1 tem-se o seguinte custo:

$$O_2(n) = l\,(l-1)\,(l-2) \times n \qquad (16).$$

Para $c = n$, utilizando-se (14) e substituindo em (16), tem-se:

$O_{2a}(n) = 8\,n^4 - 48\,n^3 + 94\,n^2 - 60\,n$.

Para *c* = 3, utilizando-se (15) e substituindo em (16), tem-se:

$O_{2b}(n) = 27\,n^4 - 189\,n^3 + 438\,n^2 - 336\,n$.

O custo final do algoritmo Intersecção de Segmentos, considerando o pior caso, com número de pontos na cobertura convexa igual a 3, é:

$O_f(n) = O_{1b}(n) + O_{2b}(n)$ ou

$O_f(n) = 27n^4 - 186n^3 + 429n^2 - 330n$.

Considerando-se o termo de mais alta ordem, tem-se:

$O_f(n) = n^4$.

## ANEXO 5.3 PESQUISA DO VIZINHO OPOSTO

No cálculo do custo teórico no pior caso deste algoritmo, verifica-se o comportamento de cada rotina em relação ao número de pontos dispersos $n$. Este custo total foi de $O_f(n) = 6n^3 - 15n^2 - 22n + 60$ com os seguintes custos parciais:

1) $n$ pesq. esq/dir x
   |
2) $n$ pontos mais proximo
   |
3) $t(3(n^2 - 4n + 6 + 2(t)))$ triangular izar

**1) Custo da pesquisa dos pontos mínimo e máximo:**

Inicialmente tem-se uma pesquisa a todos os pontos do conjunto ***P*** para verificar o valor mais à esquerda da coordenada x (ponto A), representando um custo igual a $O_1(n) = n$.

**2) Custo da pesquisa dos pontos mais próximos:**

Com o ponto A e com dois pontos mais próximos a este, defini-se o triângulo inicial (seção 3.2.3), através de uma pesquisa a todos os pontos do conjunto ***P***, resultando num custo igual a $O_2(n) = n$.

**3) Custo da geração dos triângulos:**

Após definir o triângulo inicial, faz-se uma pesquisa dos novos triângulos a serem formados com os lados deste triângulo. Esta pesquisa segundo Mclain (1976), pode ser restringida a um conjunto ***Q*** de pontos $P_i$ que estão do lado oposto ao vértice que não pertence ao lado pesquisado.

Após esta restrição, os pontos pertencentes ao conjunto ***Q*** são ordenados pela distância entre o lado de pesquisa ao circuncentro de uma circunferência, que é formada pelos pontos do lado em pesquisa (A e B) e um ponto do conjunto ***P***. O ponto no conjunto ***Q*** mais próximo do lado de pesquisa formará um novo triângulo. Portanto, conforme seção 3.2.3, tem-se o seguinte custo:

$$O_3(n) = t\left(3\left(n^2 - 4n + 6 + 2\,(t)\right)\right) \qquad (17).$$

Para $c = n$, utilizando-se

$$t = n - 2 \quad (18)$$

e substituindo em (17), tem-se:

$$O_{3a}(n) = 3n^3 - 12n^2 + 18n - 12\,.$$

Para $c = 3$, utilizando-se

$$O_{3b}(n) = n^2 + n - 2\,, \quad (19)$$

e substituindo em (17), tem-se:

$$O_{3c}(n) = 6n^3 - 15n^2 - 24n + 60 \qquad (20).$$

O custo final do algoritmo Pesquisa do Vizinho Oposto, considerando o pior caso, com número de pontos na cobertura convexa igual a 3, é:

$$O_f(n) = O_1(n) + O_2(n) + O_{3c}(n) \quad \text{ou}$$

$$O_f(n) = 6n^3 - 15n^2 - 22n + 60\,.$$

Considerando-se o termo de mais alta ordem, tem-se:

$O_f(n) = n^3$.

## ANEXO 5.4 TRIANGULARIZAÇÃO ESPIRAL

Para calcular o custo teórico no pior caso deste algoritmo, verifica-se o comportamento de cada rotina em relação ao número de pontos dispersos *n*. Considera-se o número de pontos da cobertura convexa igual a 3 pontos dispersos, já que este valor proporcionou o custo no pior caso (FIGURA 49).

Este custo total foi de $O_f(n) = 8n^2 - 19n - 8\log_2(n+4) + 46$ com os seguintes custos parciais:

| | | |
|---|---|---|
| 1) | $11n - 8\log_2(n+4) + 16$ | gerar cobertura |
| | \| | |
| 2) | 3n | definir pontos internos |
| | \| | |
| 3) | $n$ | gerar grade |
| | \| | |
| 4) | $n - 3$ | inserir pontos |
| | \| | |
| 5) | $8n^2 - 35n + 33$ | triangular izar |

Seguem-se os detalhamentos destas partes:

[DSR2] Comentário:

**1) Custo da geração da cobertura convexa:**

Inicialmente tem-se o custo da definição do conjunto ***V*** de pontos da cobertura convexa, que no caso definiu-se como o menor custo entre os algoritmos de pior custo. Este custo é definido pela ordem de complexidade igual a

$O_1(n) = 11n - 8\log_2(n+4) + 16$ (ver anexo 5.1).

**2) Custo da definição dos pontos internos:**

Após a definição dos pontos do conjunto ***V*** de pontos da cobertura convexa, excluem-se estes pontos do conjunto ***P*** para definir os pontos internos I (seção 3.2.2). Este processo representa um custo de:

$O_2(n) = n \times c$.

Como o pior custo é $c = 3$, tem-se $O_2(n) = 3n$.

**3) Custo da geração da grade G:**

Como definiu-se que a grade G contém tanto na horizontal como na vertical um número de caixas retangulares igual à raiz quadrada da quantidade de pontos P do conjunto ***P*** (ver seção 3.2.2), o custo deste procedimento resulta em $O_3(n) = \sqrt{n} \times \sqrt{n} = n$.

**4) Custo da inserção dos pontos internos I na grade G:**

Este custo representa uma pesquisa seqüencial de todos os pontos internos I. Como o pior custo é $c = 3$, tem-se $O_4(n) = n - 3$.

**5) Custo da geração dos triângulos:**

Este custo representa uma pesquisa de todos os pontos internos I para gerar os triângulos. Devendo-se verificar para cada ponto interno I se este ponto está sobre ou interno a um triângulo do conjunto ***T***. Após encontrar o triângulo $T_i$ que contém o ponto $P_i$, testa-se se este ponto está sobre um dos lados do triângulo $T_i$ para definir um novo triângulo. Desta forma, tem-se a seguinte função de custo:

$O_5(n) = (n-3)\left(t + 3(c+t)\right)$ (21).

Como o pior custo é $c = 3$, utilizando-se

$t = 2n - 5$

e substituindo em (21), tem-se:

$O_5(n) = (n-3)\left(2n - 5 + 3(3 + 2n - 5)\right)$ ou

$O_5(n) = 8n^2 - 35n + 33$.

O custo final do algoritmo Triangularização Espiral, considerando o pior caso, com número de pontos na cobertura convexa igual a 3, é:

$O_f(n) = O_1(n) + O_2(n) + O_3(n) + O_4(n) + O_5(n)$ ou

$O_f(n) = 11n - 8\log_2(n+4) + 16 + 3n + n + n - 3 + 8n^2 - 35n + 33$ ou

$O_f(n) = 8n^2 - 19n - 8\log_2(n+4) + 46$.

Considerando-se o termo de mais alta ordem, tem-se:

$O_f(n) = n^2$.

## ANEXO 5.5 OTIMIZAÇÃO DOS TRIÂNGULOS

No processo de otimização das triangularizações, calcula-se o custo teórico no pior caso pela verificação do comportamento de cada rotina em relação ao número de pontos dispersos *n*. Com o comportamento destas rotinas, tem-se a seguinte definição:

$O(n) = t \times 3(7 \times t)$ ou

$O(n) = 21t^2$ (22).

Para $c = n$, utilizando-se (18) e substituindo em (22), tem-se:

$O(n) = 21(n-2)^2$ ou

$O(n) = 21n^2 - 84\,n + 84$.

Para $c = 3$, utilizando-se (19) e substituindo em (22), tem-se:

$O(n) = 21(2n-5)^2$ ou

$O(n) = 84n^2 - 420\,n + 525$.

O custo final no processo de otimização das triangularizações, considerando o pior caso, com número de pontos na cobertura convexa igual a 3, é:

$O(n) = 84n^2 - 420\,n + 525$.

Considerando o pior caso $c = 3$, o termo de mais alta ordem é:

$O_f(n) = n^2$.

# REFERÊNCIAS BIBLIOGRÁFICAS

AGISHTEIN, Michael E.; MIGDAL, Alexander A. ***Smooth surface reconstruction from scattered data points***. Computers & Graphics, New York, v.15, n.1, p.29-39, 1991.

AZEREDO, Paulo A.**Métodos de classificação de dados e análise de suas complexidades**. Rio de Janeiro: Campos, 1996.

BORGES, José Antônio dos Santos. **Introdução às técnicas de computação gráfica 3D.** Rio de Janeiro: Núcleo de Computação Eletrônica, 1988.

CÂMARA, Gilberto et al. Spring: processamento de imagens e dados georeferenciados. In: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO GRÁFICA E PROCESSAMENTO DE IMAGENS, 5., 1992, Águas de Lindóia. **Anais**... São Paulo: SBC/INPE, 1992.

CHOI, B. K. et al. ***Triangulation of scattered data in 3D space***. Computer Aided Design, v.20, n.5, p.239-248, June 1988.

CINTRA, Jorge Pimentel. **Contribuições ao estudo de representação de superfícies com o auxílio do computador**. 1984. 270 f. Tese (Doutorado em Engenharia) - Escola Politécnica da Universidade de São Paulo, São Paulo.

EDDY, William F. A new convex hull algorithm for planar sets. ***ACM Transactions on Mathematical Software***. New York, v.3, n.4, p.398-403, dec.1977.

FELGUEIRAS, Carlos Alberto. **Desenvolvimento de um sistema de modelagem digital de terreno para microcomputadores**. 1987. 243 f. Dissertação - INPE , São José dos Campos – SP.

FELGUEIRAS, Carlos A.; GOODCHILD, Michael F. An incremental constrained delaunay triangulation. In: *NCGIA - ANNUAL GIS BIBLIOGRAPHY FOR 1995*. **Anais**... California: University of California, 1995.

GOODCHILD, Michael F.; KEMP, Karen K. *Data structures & algorithms for surfaces, volumes & time. In: NCGIA – ANNUAL GIS BIBLIOGRAPHY FOR 1995*. **Anais...** Santa Barbara: University of California, 1995. p. 38. 1-42.9.

HOROWITZ, Ellis; SAHNI, Sartaj. **Fundamentos de estruturas de dados**. Rio de Janeiro-RJ: Campus, 1984.

LAWSON, C. L. ***Mathematical Software III***. New York: Academic Press, 1977. p.161-194.

LEE, D. T.; SCHACHTER, B. J. ***Two algorithms for construting a delaunay triangulation***. International Journal of Computer and Information Sciences, v.9, n.3, p.219-242, 1980.

MCLAIN, D. H. *Two dimensional interpolation from random* data. **The Computer Journal**, [S.l.], v.19, n.2, p.178-181, May 1976.

MIRANTE, Anthony; WEINGARTEN, Nicholas H. The radial sweep algorithm for constructing triangulated irregular networks. ***IEEE computer graphics and applications***, New York, v.53, 1982.

NEWMAN, William M.; SPROULL, Robert F. ***Principles of interactive computer graphics.*** 2. Ed. Singapore: McGraw-Hill, 1979.

PAREDES, Evaristo Atencio. **Sistema de informação geográfica – geoprocessamento**: princípios e aplicações. São Paulo: Érica, 1994.

PARK, Chan S. ***Interactive microcomputer graphics.*** Alabama: Addison Wesley, 1985.

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

PFEIFLE, Ron et al. ***Fitting triangular b-splines to functional scattered data.*** Computer Graphics Forum, v.15, n.1, p.15-23, 1996.

PREPARATA, Franco P.; SHAMOS, Michael Ian. ***Computational geometric – an introduction.*** New York: Spring Verlar, 1985. 398 p.

REIS, Dalton Solano dos. **Análise da geração de grades irregulares triangularizadas em modelos numéricos de terreno**. 1997. 214 f. Dissertação (Mestrado em Ciências da Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

RIEDER, André. OpenGL informações & tutorial. UNICAMP, São Paulo, Abr. 2000. Disponível em: http://www.dca.fee.unicamp.br/~arieder/opengl/. Acesso em:18 abr 2001.

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

SENE, Eustáqui de; Moreira, João Carlos. **Geografia geral e do Brasil**. São Paulo : Scipione, 1999.

SPATIAL DATA: *applications, concepts, techniques*. **The Computer Journal**, [s.1], v.37, n.1, p.81, 1994.

TOSCANI, Laira V.; VELOSO, Paulo A. S. Divisão e conquista: análise da complexidade. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 6., 1986. **Anais**... Recife-Olinda: [s.n.], 1986.

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.