Fulano de Tal

Título da Monografia

Trabalho de conclusão de curso apresentada ao Departamento de Matemática, Estatística e Computação (DMEC), da Universidade Estadual Paulista "Júlio de Mesquita Filho", sob o título **"Título da Monografia"**.

Orientador: Prof. Dr. Nome do Orientador

Co-orientador: Prof. Dr. Nome do Coorientador

Presidente Prudente

2010

Termo de Aprovação

Aluno Fulado de Tal

Título da Monografia

Monografia sob o título **"Título da Monografia"**, defendida por Nome do Aluno e aprovada em 29 de fevereiro de 2002, em Santo Antônio do Aracanguá, Estado de São Paulo, pela banca examinadora constituída pelos doutores:

Prof. Dr. Nome do Professor 1
Universidade de Ágora

Prof. Dr. Nome do Professor 2
Universidade de Constantinopla

Prof. Dr. Nome do Professor 3
Universidade de Ágora

Presidente Prudente, XX de dezembro de 2010

Agradecimentos

A realização do presente curso foi possível devido à colaboração de muitas pessoas que me auxiliaram durante os 5 anos de curso. Manifesto assim minha gratidão:

primeiramente a Deus, que sempre me ajudou a não desistir dessa longa caminhada, e que sempre me acompanha durante a execução de minhas tarefas.

outros agradecimentos .....

Resumo

Elemento obrigatório, constituído de uma sequência de frases concisas e objetivas e não de uma simples enumeração de tópicos. Deve conter de 150 a 500 palavras, seguido logo abaixo das palavras-chave (conforme NBR 6028).

Elemento obrigatório. Consiste em uma versão do resumo em idioma de divulgação internacional (em inglês Abstract, em espanhol Resumen, em francês Résumé). Também deve ser seguido de palavras-chave,

palavras-chave: palavras representativas do conteúdo do trabalho, separadas entre si por ponto (.) e finalizadas também por ponto. Devem aparecer logo abaixo do resumo, antecedidas da expressão “palavras-chave:”.

Abstract

Elemento obrigatório. Consiste em uma versão do resumo em idioma de divulgação internacional (em inglês Abstract, em espanhol Resumen, em francês Résumé). Também deve ser seguido de palavras-chave, na língua.

keywords: palavras representativas do conteúdo do trabalho, separadas entre si por ponto (.) e finalizadas também por ponto. Devem aparecer logo abaixo do resumo, antecedidas da expressão “keywords:”.

Sumário

Lista de Figuras

Capítulo 1

Bancos de Dados Não Relacionais - NoSQL

Em um trabalho acadêmico intitulado "Selected Topics on Software-Technology Ultra-Large Scale Sites"(Tópicos Selecionados na Tecnologia de Softwares de Sites de Escala Ultra Grande) na Stuttgart Media University(Universidade de Mídia de Stuttgart) Christof Strauch reúne várias referências e tópicos sobre todo o assunto de base do NoSQL, conforme seu trabalho, os bancos de dados relacionais são os tipos predominantes para armazenamento de dados estruturados, desde 1970 este tipo de base de dados se suporta em uma aritmética relacional com o proliferado uso do SQL(Secure Query Language) para inserir, alterar, recuperar e remover os dados. Bases de dados não convencionais eram utilizadas apenas para aplicações de algum nicho específico como projetos utilizando SOAP que utiliza o padrão XML, porém o pensamento por cima desse uso disseminado "um tamanho para caber todos"tem sido questionado tanto pelas empresas como pelos acadêmicos devido ao aumento de usuários e volume de dados no meio.

Conforme segue na argumentação de Christof Strauch, para suprir essa demanda algumas alternativas começam a ser propostas, vários outros modelos são criados para abrangir mais nichos antes dominados pelos SGBDRs, esses modeloss adotam caminhos não convencionais nem sempre garantindo a ACID (Atomicidade, Consistência, Isolamento e Durabilidade) dos dados, porém conseguem optimizar o espaço de armazenamento e a performance das bases de dados, não necessariamente se desfazendo do uso do SQL, por isso a sigla NoSQL significa "*not only secure query language*".

1.1 Definição

1.2 Aspectos Comuns ao NoSQL

Esta seção destina-se a criar uma base sólida sobre os principais aspectos e conceitos que envolvem o NoSQL, além de apontar diferenças e temas relacionados com SGBDs também serão abordados teoremas e tópicos específicos sobre o NoSQL

1.2.1 Recapitulando SGBDRs

Em uma tese de mestrado intitulada "Extracting Data from NoSQL Databases. A Step towards Interactive Visual Analysis of NoSQL Data"(Extração de dados de bases NoSQL. Um passo avante para análise de visualização interativa de NoSQL) da Chalmers University of Technology(Chalmers, Universidade de Tecnologia) Petter Näsholm apresenta uma recapitulação rápida sobre as bases de dados convencionais. SGBDRs se baseiam no conceito de tabelas com conjunto de atributos podendo ser chamados de colunas da tabela, cada atributo é associado a um tipo de dado(como inteiro, real ou string). Cada linha desta tabela é considerada uma tupla, ao passo que os atributos dentro de cada tupla podem ser preenchidos com componentes referente ao seu tipo. As tabelas também podem ter colunas com informações sobre a relação entre elas mesmas.

1.2.2 ACID e BASE

Petter Näsholm também aborda as garantias que a maioria dos SGBDRs adotam sendo elas a sigla ACID(Atomicidade, Consistência, Isolação, Durabilidade) onde Atomicidade é a garantia de que ou uma trasação é executada de forma completa ou não é executada de forma alguma, a Consistência garante que toda transação saia de um estado da base válido para outro, Isolação é a garantia de que não interfere outra no tempo em que estão ocorrendo e a Durabilidade garante que o efeito de uma transação deve persistir e, portanto, nunca ser perdido. No trabalho de Christof Strauch o autor apresenta o BASE que são as garantias dos NoSQL:

Basic available(Basicamente disponível): Uma aplicação funciona basicamente todo o tempo.

Soft-state(Estado "leve, macio", simples): Não precisa ser consistente o tempo todo.

Eventual consistency(Eventual consistência): Em algum momento conhecido haverá

consistência.

O aspecto com maior diferença entre os modelos é a parte da consistência, no modelo ACID existe uma consistência estrita enquanto no base há uma consistência eventual. A consistência estrita implica que toda operação de leitura deve retornar resultados a partir da última escrita, devido a isso as operações de leitura e escrita devem ser feitas no mesmo servidor ou obedecer a um protocolo de transação distribuída, porém, como veremos abaixo pelo teorema CAP, essa forma de operar implica ou na perda da disponibilidade dos dados ou na tolerância do particionamento dos mesmos. A consistência eventual permite que leitores acessem a base conforme os dados vão sendo escritos, não importando necessariamente se aquele registro foi o último a ser escrito, eventualmente em um estado estável será retornado o último registro escrito, até isso ocorrer o sistema pode estar em um estado de inconsistência.

Além das garantias colocadas acima Christof Strauch lista outras que também podem ser fornecidas:

RYOW - Read Your Own Writes(leia seus próprios escritos): O leitor deve visualizar imediatamente aquilo que ele próprio escreveu, independentemente se o leitor acessa pelo próprio servidor que escreveu ou por outro.

Consistência da Sessão: Acompanha a garantia do RYOW porém apenas para a sessão que o leitor estiver conectado, ou seja, caso escreva os dados por um servidor e tenter ler por outro a consistência não será garantida.

Consistência ocasional: se foi escrito y após ser lido x, qualquer leitor que ver y também verá x.

1.2.3 Teorema CAP

O Teorema CAP que, conforme Christof Strauch, é muito adotado pela comunidade NoSQL, apresenta um conceito referente as limitações estritamente associadas as capacidades que oferecem.

A sigla CAP se refere à Consistência(Consistency), Disponibilidade(Availability), Tolerância ao particionamento(Partition Tolerance), sendo respectivamente a habilidade de manter os dados compartilhados disponíveis a todos os leitores após um escritor executar um update, a habilidade de continuar disponível mesmo com a falta de algum cluster ou parte do sistema ficar indisponível e a habilidade de continuar operando mesmo que um

servidor não consiga se conectar a outro servidor da mesma aplicação também entendido como a habilidade de poder adicionar ou remover nós de uma rede particionada. Este Teorema propõe que só é possível para qualquer aplicação combinar duas dentre as três características, ou seja ela pode ter muita consistencia e disponibilidade porém com pouca tolerância ao particionamento, muita consistencia e tolerância ao particionamento porém com pouca disponibilidade ou muita disponibilidade e tolerância ao particionamento e pouca consistência.

Observando pelo teorema CAP os NoSQL buscam ofertar alternativas para aplicações que nem sempre dependem de consistência ou disponibilidade ou tolerância ao particionamento.

Também existem outros aspectos mais profundos mas que não seram abordados neste documento como o versionamento do conjunto dos dados, os cenários distribuídos e o particionamento. Os aspectos vistos acima nos dão aparato suficiente para progredir o trabalho focando nas questões principais sobre as necessidades da aplicação e o uso de diferentes tipos de SGBDs para qual se encaixam melhor.

1.3 Formas de Armazenamento

Nesta seção seram abordados algums tipos de bases NoSQL, o modelo orientado à documento utilizando JSON será aprofundado pois foi escolhido para dar suporte a este trabalho devido a sua fácil integração com a ferramenta de visualização de dados escolhida.

1.3.1 Armazenamento baseado em linhas

1.3.2 e serviços rest, json e datavisualization

1.4 Do SQL ao NOSQL

1.5 NoSQL Orientado a Documento

falar dos tipos existentes

1.6 Críticas ao NoSQL pag 15 http://www.christof-strauch.de/nosqldbs.pdf

Capítulo 2

PostgreSQL e os tipos de dados não convencionais

2.1 PostgreSQL

Falar sobre o PostgreSQL que é um Banco que segue as mais recentes ISOS que tem as garantias ACID, é transacional, antigo maduro, existe em várias plataformas, suporte para dados complexos, opensource, arquitetura extensível, e agora tem suporte para dados sem schema json, xml e hstore.

Sobre os dados sem schema que utiliza, o Xml é estruturado de maneira hierarquica sendo o padrão W3C standard 1998 o padrão é base para o protocolo de troca de mensagens SOAP, porém seu uso é um pouco complexo e seu desempenho de velocidade é o pior dos dados em armazenamento em documentos postgresql, sem indexação.

Outro tipo é o Hstore, este tipo tem um desempenho bem expressivo pois baseia-se apenas em chave para valores em strings ou outros valores hstore, embora seja eficiente a falta de aninhamento e diferentes tipos dificulta sua utilização para situações mais complexas, não é muito flexível.

Por último temos o tipo JSON e JSONB, que é o tipo Orientado a Documento baseado na notação JavaScript, é utilizado como forma para armazenamento e troca de dados entre várias plataformas web e mobile. Abaixo temos um aprofundamento sobre este tipo por ser base ao trabalho.

2.2 JSON

JSON significa JavaScript Object Notation, é um formato de dados de fácil leitura e escrita para os humanos e de fácil geração e análise para as máquinas. Um objeto em JSON é um conjunto de pares de chave e valor que podem ser organizados de forma aninhada e(ou) listados em arrays, a chave é uma string e o valor pode ser uma string, um numero, um booleano, um array e até mesmo outro objeto aninhado. O array pode ter valores dos mesmos tipos citados. Pode haver também uma lista de objetos no mesmo nível, esta lista pode obedecer ou não a sua ordem.

2.3 BSON

O termo BSON é uma junção de Binary JSON, ele é a serialização de documentos JSON para binário, foi desenvolvido para melhorar o desempenho na busca das chaves. Além de armazenar os dados em arquivos binários a maior diferença entre ele e o JSON é o uso de tipos de dados com definição de tamanho e tipo pré-determinadas, para armazenar um número inteiro o JSON utiliza uma cadeia de caracteres arbitrária enquanto o BSON utiliza 32 e 64 bits para inteiros e 64 bits para double e floats, tipos booleanos em JSON são representados pela cadeia de caracteres 'True' ou 'False', em BSON é representado por apenas um bit 0 ou 1. Para valores como cadeias de strings são armazenadas meta-informações como o comprimento dela e, como os outros tipos já tem comprimentos definidos, um parsing na busca por uma chave pode facilmente saltar para os pontos em que interessa no documento.

2.4 Json vs JsonB

Antes do JSONB fazer parte do PosgreSQL Geoffrey Litt desenvolveu em seu projeto final "Improving performance of schemaless document storage in PostgreSQL using BSON"(Aprimorando a performance de armazenamento orientado a documento em PostgrSQL utilizando BSON) da Yale University(Universidade de Yale), através do fonte do PostgreSQL, bibliotecas de parsing JSON e bibliotecas de parsing BSON do Mongodb, um campo similar chamado BSON, em sua tese pode testar as diferenças de performance destes campos. Posteriormente o PostgreSQL adotou este campo.

Para o primeiro teste utilizou o mesmo documento e variando seu tamanho de 1 à 90 vezes, nele executava uma query simples para encontrar um valor no meio do documento,

o teste demonstrou que quanto maior o documento maior era a velozidade do BSON sobre o JSON, com o tamanho máximo o BSON foi 8.28 vezes mais rápido para o JSON.

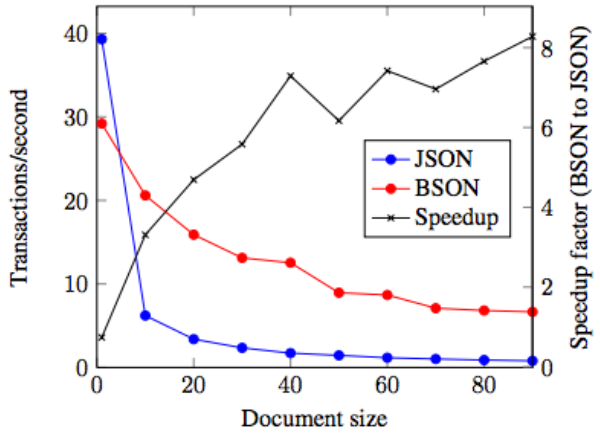

Figura 1: Performance relativa de BSON e JSON para aumento de documento sobre número constante de tuplas (GREEN P.J.; SIBSON, 1978).

No segundo teste utilizou o mesmo tamanho de documentos mas aumentou o número de tuplas, nele pode constatar que a velocidade de BSON sobre o JSON aumentava com o número de tuplas porém após o número de 100 tuplas essa velocidade começa a convergir para um valor constante de 6 vezes em relação ao JSON.

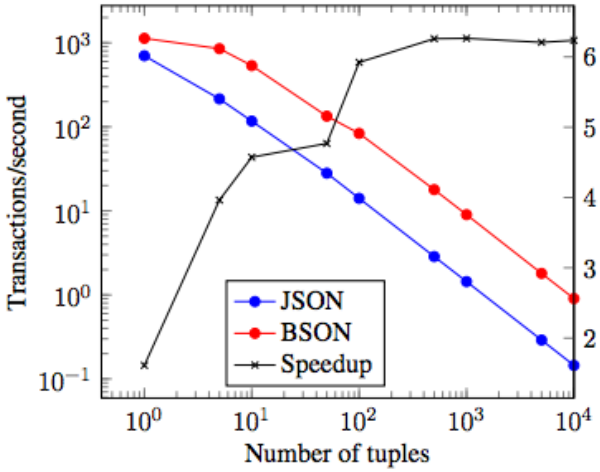

Figura 2: Performance relativa de BSON e JSON para aumento de documento sobre número constante de tuplas (GREEN P.J.; SIBSON, 1978).

Outro teste foi realizar queries por uma chave variando seu lugar do começo ao fim do documento e variando o tamanho do documento, a velocidade do bson foi aproximadamente 4 vezes maior em relação ao json para todos os tamanhos de documentos com chaves próximas do começo do documento, para chaves próximas do fim do documento a velocidade do bson foi aproximadamente 6 vezes maior em relação ao json para todos os

tamanhos de documentos, isso deve-se à capacidade que o bson tem de atravessar rapidamente as chaves por ter meta-informações sobre seu comprimento enquanto o json tem que analisar caracter a caracter.

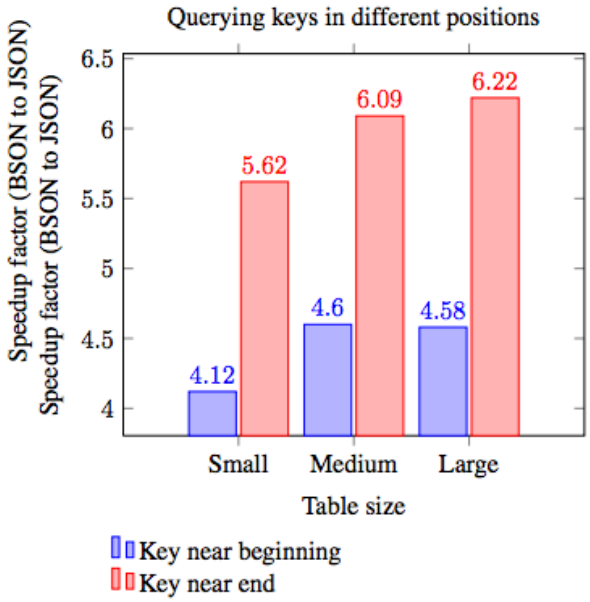

Figura 3: Performance relativa de BSON e JSON para aumento de documento sobre número constante de tuplas (GREEN P.J.; SIBSON, 1978).

Foi realizado também um teste sobre a composição do documento, ou seja, os tipos de dados dentro do documento, executando queries em 3 documentos, o primeiro com apenas stringsi de 10 caracteres, o segundo apenas com inteiros e o terceiro com apenas booleanos, para strings a valocidade do bson foi 3 vezes superior sobre o json, para inteiros e booleanos a velocidade do bson foi aproximadamente 5,5 vezes superior ao json, o resultado superior de inteiros e booleanos deve-se ao fato do bson utilizar inteiros de 32bits e booleanos de 1 bit, como foi comentado acima, e o json utilizar cadeias de caracteres para representá-los.

Por último foi realizado um teste para verificar o tempo de carregamento dos arquivos em disco, como o bson precisa de um passo a mais no parsing transformando os valores de strings para os tipos de inteiros, booleanos, etc, criando os metadados sobre strings e comprimento das chaves, seu tempo de carregamento é pouco mais demorado em relação ao json, como pode-se verificar na figura 5, entretanto para esse teste foi utilizado o parsing de validação do json e posteriormente foi salvo em disco, caso tivesse sido usado um parsing de validação já em bson essa diferença poderia se igualar ou até mesmo inverter-se.

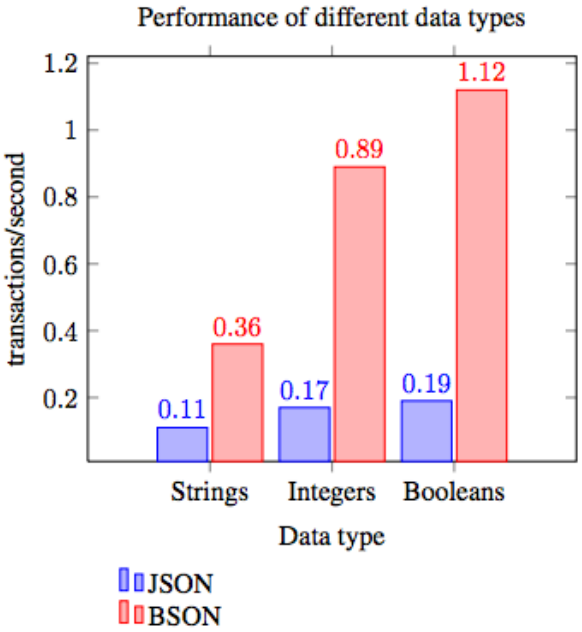

Figura 4: Performance relativa de BSON e JSON para aumento de documento sobre número constante de tuplas (GREEN P.J.; SIBSON, 1978).

	1,000 tuples	10,000 tuples	100,000 tuples
JSON	6.100s	62.09s	630000s
BSON	6.139s	62.95s	635600s

Figura 5: Performance relativa de BSON e JSON para aumento de documento sobre número constante de tuplas (GREEN P.J.; SIBSON, 1978).

2.5 Técnicas para Execução de Queries

O capítulo 3 contém...

O capítulo 4 destina-se...

O capítulo 5 discute sobre...

Por fim, no capítulo 6 será feita uma conclusão...

Capítulo 3

Fundamentação Teórica

3.1 Triangulação

A Triangulação de um conjunto de pontos não possui unicidade. Em outras palavras, a partir de um mesmo conjunto de pontos é possível obter diferentes triangulações cada uma delas com $2(n-1)k$ triângulos e $3(n-1)-k$ arestas, onde k é o número de pontos pertencentes ao (*Fecho Convexo*) da Triangulação. No caso a forma dos triângulos também podem influenciar na aplicação que se deseja fazer, é necessário evitar triângulos muito finos(ângulos internos muito agudos) na Triangulação como na figura 6 a) e b), o ideal é termos triângulos o mais próximo possível dos triângulos equiláteros com os ângulos se aproximando de 60^{O}, figura 6 c).

Figura 6: Possíveis triangulações para um mesmo conjunto de pontos.

3.2 Primitivas Geométricas

Nessa pequena introdução falaremos...

Pontos cocirculares e polígonos inscritos possuem muitas propriedades especiais que permitem uma construção eficiente de uma triangulação equiangular (SIBSON, 1978). Algumas dessas propriedades serão descritas a partir dos *LEMAS*.

Lema 1 *: Cada Triangulação de um polígono de n lados inscrito terá entre seus ângulos o mesmo conjunto de n ângulos, isto é, os n ângulos opostos às n cordas do lado de fora.*

A seguir trataremos...

3.3 CounterClockWise

A seguir introduziremos...

Se a dimensão $n = 1$, então a orientação de $(\mathbf{p}_1, \mathbf{p}_2)$ é positiva se $\mathbf{p}_1 > \mathbf{p}_2$ e negativa se $\mathbf{p}_1 < \mathbf{p}_2$ comparar com a figura 7. Se $n = 2$, então$(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ tem orientação positiva se três pontos rotacionam se à esquerda no plano, isto é, $\mathbf{p}_3$, fica à esquerda da linha que passa por $\mathbf{p}_1$,$\mathbf{p}_2$ nesta ordem. Se $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ faz uma rotação à direita, então a orientação é negativa. Note que a orientação de $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ é a mesma de $(\mathbf{p}_1, \mathbf{p}_2)$ como visto. De fato, a linha que passa por $\mathbf{p}_2$ e $\mathbf{p}_3$ pode ser identificada com R^1 tão logo escolhamos a direção da linha. Essa direção é dada pela localização de $\mathbf{p}_1$; que vai da esquerda para direita a partir de $\mathbf{p}_1$.

Figura 7: Ilustração dos possíveis resultados da primitiva *Ccw(.)*

Para $n > 2$, a orientação de $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4)$, é dada a partir da resolução de uma determinante, no qual pode ser usada a técnica da eliminação de Gauss, Laplace ou outras técnicas para resolvê-lo.

Seja $\mathbf{p}_1 = (\mathbf{p}_{1x}, \mathbf{p}_{1y})$, $\mathbf{p}_2 = (\mathbf{p}_{2x}, \mathbf{p}_{2y})$, $\mathbf{p}_3 = (\mathbf{p}_{3x}, \mathbf{p}_{3y})$ três pontos no plano, consideraremos $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ indefinidos se forem coplanares. No caso indefinido, o ponto $\mathbf{p}_3$ é uma combinação linear de $\mathbf{p}_1$ e $\mathbf{p}_2$, $\mathbf{p}_3 = \lambda_1\mathbf{p}_1 + \lambda_2\mathbf{p}_2$ com $\lambda_1 + \lambda_2 = 1$. Dessa maneira λ_1, λ_2 existirão se e somente se o determinante de

$$Ccw(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3) = \begin{pmatrix} 1 & \mathbf{p}_{1x} & \mathbf{p}_{1y} \\ 1 & \mathbf{p}_{2x} & \mathbf{p}_{2y} \\ 1 & \mathbf{p}_{3x} & \mathbf{p}_{3y} \end{pmatrix} = 0 \tag{3.1}$$

Se não acontecer isso então $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ orientam-se para esquerda ou direita e o uso do determinante dará a orientação de $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$.

Para $n+1$ pontos $(\mathbf{p}_1, \ldots, \mathbf{p}_n)$ o teste de orientação vai indicar como os pontos estão orientados no hiperplano e nesse caso geral a primitiva *Ccw* é dada pelo sinal do determinante:

$$\begin{vmatrix} 1 & \mathbf{p}_{12} & \cdots & \mathbf{p}_{1n} & \mathbf{p}_{11} \\ 1 & \mathbf{p}_{22} & \cdots & \mathbf{p}_{2n} & \mathbf{p}_{21} \\ \vdots & \vdots & & \vdots & \vdots \\ 1 & \mathbf{p}_{n+1,2} & \cdots & \mathbf{p}_{n+1,n} & \mathbf{p}_{n+1,1} \end{vmatrix} \tag{3.2}$$

Teste do plano: Seja $T = \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$ e ht é o único plano que contém todos três pontos de T. Esse plano pode ser orientado se substituirmos o conjunto T por uma seqüencia T, por exemplo, $T = (\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$. Assim, um lado de ht pode ser o positivo e o outro o negativo, e os referimos como os lados da seqüência T.

$T = (\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$: $\mathbf{p}_4$ fica no lado positivo (esquerda) se e somente se $\det(Ccw(.)) > 0$ e orienta-se para o lado negativo (direita) se e somente se $\det(Ccw(.)) < 0$

Prova Verificaremos para $\mathbf{p}_1 = (0, 0)$, $\mathbf{p}_2 = (1, 0)$ e $\mathbf{p}_3 = (0, 1)$, geometricamente é obvio que $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ orientam-se para esquerda (lado positivo), de fato.

$$\det \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = 1 \tag{3.3}$$

Portanto...

3.4 InCírculo

Esta primitiva tem por finalidade verificar se o ponto $\mathbf{p}_4$ está no interior do círculo definido pelos pontos $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$. Consideraremos $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$ degenerados se $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ coplanares ou $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$ cocirculares. Para mostrar se os pontos são cocirculares faremos $\mathbf{p}'_1 = (\mathbf{p}_{1x}, \mathbf{p}_{1y}, \mathbf{p}_{1z})$, com $\mathbf{p}_{1z} = \mathbf{p}_{1x}^2 + \mathbf{p}_{1y}^2$ e assim por diante. Os pontos serão cocirculares se $\mathbf{p}'_1$, $\mathbf{p}'_2$, $\mathbf{p}'_3$, $\mathbf{p}'_4$ forem coplanares. Em outras palavras, $\mathbf{p}'_4$ é combinação linear de $\mathbf{p}'_1$, $\mathbf{p}'_2$, $\mathbf{p}'_3$ (BERG; OVERMARS; SCHWARZKOPF, 1977). O resultado de *InCirculo* é obtido, solucionando o seguinte determinante:

$$InCirculo\ (\mathbf{p}_1,\ \mathbf{p}_2,\ \mathbf{p}_3,\ \mathbf{p}_4) = \begin{vmatrix} \mathbf{p}_{1x} & \mathbf{p}_{1y} & \mathbf{p}_{1x}^2 + \mathbf{p}_{1y}^2 & 1 \\ \mathbf{p}_{2x} & \mathbf{p}_{2y} & \mathbf{p}_{2x}^2 + \mathbf{p}_{2y}^2 & 1 \\ \mathbf{p}_{3x} & \mathbf{p}_{3y} & \mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2 & 1 \\ \mathbf{p}_{4x} & \mathbf{p}_{4y} & \mathbf{p}_{4x}^2 + \mathbf{p}_{4y}^2 & 1 \end{vmatrix}$$

Se:

InCirculo ($\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$) = 0 então os pontos são cocirculares;

InCirculo ($\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$) > 0 então $\mathbf{p}_4$ pertence ao exterior do círculo;

InCirculo ($\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$) < 0 então $\mathbf{p}_4$ pertence ao interior do círculo; Veja a representação na figura 9

Essa primitiva é baseada no seguinte lema:

Lema 2 *: O teste InCirculo(.) é equivalente a:*

$$InCirculo(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4) = \begin{vmatrix} \mathbf{p}_{1x} & \mathbf{p}_{1y} & \mathbf{p}_{1x}^2 + \mathbf{p}_{1y}^2 & 1 \\ \mathbf{p}_{2x} & \mathbf{p}_{2y} & \mathbf{p}_{2x}^2 + \mathbf{p}_{2y}^2 & 1 \\ \mathbf{p}_{3x} & \mathbf{p}_{3y} & \mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2 & 1 \\ \mathbf{p}_{4x} & \mathbf{p}_{4y} & \mathbf{p}_{4x}^2 + \mathbf{p}_{4y}^2 & 1 \end{vmatrix}. \tag{3.4}$$

Prova Para provar a validade desse resultado teremos que olhar para equação do parabolóide Γ representado no R^3 de equação 3.5 e ilustrado na figura 8.

Seja H um plano não-vertical de equação 3.6, a projeção O_{xy} de $H \bigcap \Gamma$ no plano da origem a um círculo de equação 3.7. Ver figura 8 que contém a projeção mencionada.

Seja $\mathbf{p} = (\mathbf{p}_x, \mathbf{p}_y)$ elevaremos o ponto $\mathbf{p}$ e obtemos um $\mathbf{p}'$ de equação 3.8. Essa transformação $\mathbf{p} \longrightarrow \mathbf{p}'$ é chamada **mapa de elevação**. A elevação dos pontos $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ e $\mathbf{p}_4$ resulta respectivamente nas equações 3.9, 3.10, 3.11 e 3.12. Assim se *InCirculo* ($\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$) = 0, significa que $\det(Ccw(.)) = 0$ e o ponto $\mathbf{p}'_4 \in H$ e consequentemente $\mathbf{p}_4$ é cocircular à circunferência definida por $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, caso *InCirculo* ($\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$) > 0 então $\det(Ccw(.)) > 0$ e o ponto $\mathbf{p}'_4$ está acima do plano H e $\mathbf{p}_4$ fora do círculo $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, por último se *InCirculo* ($\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$) < 0 então $\det(Ccw(.)) < 0$ e o ponto $\mathbf{p}'_4$ abaixo do plano H e $\mathbf{p}_4$ no interior do círculo $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$.

Figura 8: Parabolóide Γ e a projeção da área interceptada pelo plano H no R^2

$$z = x^2 + y^2 \tag{3.5}$$

$$z = \alpha x + \beta y + \gamma \tag{3.6}$$

$$x^2 + y^2 = \alpha x + \beta y + \gamma \tag{3.7}$$

$$\mathbf{p}' = (\mathbf{p}_x, \mathbf{p}_y, {\mathbf{p}_x}^2 + {\mathbf{p}_y}^2) \tag{3.8}$$

$$\mathbf{p}'_1 = (\mathbf{p}_{1x}, \mathbf{p}_{1y}, {\mathbf{p}_{1x}}^2 + {\mathbf{p}_{1y}}^2) \tag{3.9}$$

$$\mathbf{p}'_2 = (\mathbf{p}_{2x}, \mathbf{p}_{2y}, {\mathbf{p}_{2x}}^2 + {\mathbf{p}_{2y}}^2) \tag{3.10}$$

$$\mathbf{p}'_3 = (\mathbf{p}_{3x}, \mathbf{p}_{3y}, {\mathbf{p}_{3x}}^2 + {\mathbf{p}_{3y}}^2) \tag{3.11}$$

$$\mathbf{p}'_4 = (\mathbf{p}_{4x}, \mathbf{p}_{4y}, {\mathbf{p}_{4x}}^2 + {\mathbf{p}_{4y}}^2) \tag{3.12}$$

Convém lembrar que o teste do *Incirculo*, em uma situação geral, serve também para orientação de um ponto em relação à uma esfera no R^3 e também no hiperplano. Dada uma sequência de $n+1$ pontos $(\mathbf{p}_2, \ldots, \mathbf{p}_{n+1})$, se $\mathbf{p}_1$ pertencer ao exterior da esfera, dizemos que está orientado positivamente caso contrário, está orientado negativamente, e o sinal do determinante indica, assim como no caso do círculo, a orientação de $\mathbf{p}_1$ e esse determinante no espaço R^n está ilustrado como abaixo:

$$\begin{vmatrix} \mathbf{p}_{11} & \cdots & \mathbf{p}_{1n} & \mathbf{p}_{11}^2 + \ldots + \mathbf{p}_{1n}^2 & 1 \\ \mathbf{p}_{21} & \cdots & \mathbf{p}_{2n} & \mathbf{p}_{21}^2 + \ldots + \mathbf{p}_{2n}^2 & 1 \\ \vdots & & \vdots & \vdots & \vdots \\ \mathbf{p}_{n+2,1} & \cdots & \mathbf{p}_{n+2,n} & \mathbf{p}_{n+2,1}^2 + \ldots + \mathbf{p}_{n+2,n}^2 & 1 \end{vmatrix} \tag{3.13}$$

A resolução do determinante pode ser obtida através da relação:

InCirculo ($\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$) =

$$
\begin{array}{ll}
-((\mathbf{p}_{1y}) * (\mathbf{p}_{2x}) * (\mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2)) & +((\mathbf{p}_{1x}) * (\mathbf{p}_{2y}) * (\mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2)) \\
+((\mathbf{p}_{1y}) * (\mathbf{p}_{2x}^2 + \mathbf{p}_{2y}^2) * (\mathbf{p}_{3x})) & -((\mathbf{p}_{1x}^2 + \mathbf{p}_{1y}^2) * (\mathbf{p}_{2y}) * (\mathbf{p}_{3x})) \\
-((\mathbf{p}_{1x}) * (\mathbf{p}_{2x}^2 + \mathbf{p}_{2y}^2) * (\mathbf{p}_{3y})) & +((\mathbf{p}_{1x}^2 + \mathbf{p}_{1y}^2) * (\mathbf{p}_{2x}) * (\mathbf{p}_{3y})) \\
+((\mathbf{p}_{1y}) * (\mathbf{p}_{2x}) * (\mathbf{p}_{4x}^2 + \mathbf{p}_{4y}^2)) & -((\mathbf{p}_{1x}) * (\mathbf{p}_{2y}) * (\mathbf{p}_{4x}^2 + \mathbf{p}_{4y}^2)) \\
-((\mathbf{p}_{1y}) * (\mathbf{p}_{3x}) * (\mathbf{p}_{4x}^2 + \mathbf{p}_{4y}^2)) & +((\mathbf{p}_{2y}) * (\mathbf{p}_{3x}) * (\mathbf{p}_{4x}^2 + \mathbf{p}_{4y}^2)) \\
+((\mathbf{p}_{1x}) * (\mathbf{p}_{3y}) * (\mathbf{p}_{4x}^2 + \mathbf{p}_{4y}^2)) & -((\mathbf{p}_{2x}) * (\mathbf{p}_{3y}) * (\mathbf{p}_{4x}^2 + \mathbf{p}_{4y}^2)) \\
-((\mathbf{p}_{1y}) * (\mathbf{p}_{2x}^2 + \mathbf{p}_{2y}^2) * (\mathbf{p}_{4x})) & +((\mathbf{p}_{1x}^2 + \mathbf{p}_{1y}^2) * (\mathbf{p}_{2y}) * (\mathbf{p}_{4x})) \\
+((\mathbf{p}_{1y}) * (\mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2) * (\mathbf{p}_{4x})) & -((\mathbf{p}_{2y}) * (\mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2) * (\mathbf{p}_{4x})) \\
-((\mathbf{p}_{1x}^2 + \mathbf{p}_{1y}^2) * (\mathbf{p}_{3y}) * (\mathbf{p}_{4x})) & +((\mathbf{p}_{2x}^2 + \mathbf{p}_{2y}^2) * (\mathbf{p}_{3y}) * (\mathbf{p}_{4x})) \\
+((\mathbf{p}_{1x}) * (\mathbf{p}_{2x}^2 + \mathbf{p}_{2y}^2) * (\mathbf{p}_{4y})) & -((\mathbf{p}_{1x}^2 + \mathbf{p}_{1y}^2) * (\mathbf{p}_{2x}) * (\mathbf{p}_{4y})) \\
-((\mathbf{p}_{1x}) * (\mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2) * (\mathbf{p}_{4y})) & +((\mathbf{p}_{2x}) * (\mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2) * (\mathbf{p}_{4y})) \\
+((\mathbf{p}_{1x}^2 + \mathbf{p}_{1y}^2) * (\mathbf{p}_{3x}) * (\mathbf{p}_{4y})) & -((\mathbf{p}_{2x}^2 + \mathbf{p}_{2y}^2) * (\mathbf{p}_{3x}) * (\mathbf{p}_{4y}));
\end{array}
$$

A função auxiliar *Area* utiliza a relação da equação 3.14.

$$
Area\ (\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k) = 1/2 * [(\mathbf{p}_{jx} - \mathbf{p}_{ix}) * (\mathbf{p}_{ky} - \mathbf{p}_{iy})] - [(\mathbf{p}_{jy} - \mathbf{p}_{iy}) * (\mathbf{p}_{kx} - \mathbf{p}_{ix})]; \qquad (3.14)
$$

Figura 9: Ilustração dos resultados da primitiva *InCículo*.

3.5 Circuncentro...

A primitiva circuncentro...

Mediatriz:

Definição 1 *Consideramos um segmento $\overline{AB}$ e seja M o ponto médio do segmento. A reta perpendicular ao segmento $\overline{AB}$ e que passa por M é denominada* ***mediatriz****.*

Circuncentro:

Definição 2 *Dado um $\Delta(\boldsymbol{p}_1\boldsymbol{p}_2\boldsymbol{p}_3)$ se traçarmos as mediatrizes dos três lados elas intercectam-se num mesmo ponto $\boldsymbol{c}$, que está eqüidistante dos vértices do triângulo. Este ponto $\boldsymbol{c}$ é chamado de* ***circuncentro do triângulo****.*

Tomando $\mathbf{c}$ como centro e o raio como a distância de $\mathbf{c}$ a um dos *vértices*, obteremos a circunferência que circunscreve o $\Delta(\mathbf{p_1p_2p_3})$. Esta descrição pode ser observado na figura 10

Figura 10: Representação do Circuncentro de um elemento triangular e da circunferência por ele determinada.

Prova Seja o $\Delta\mathbf{p_1p_2p_3}$

Hipótese: $\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3$ mediatrizes de $\overline{\mathbf{p_2p_3}}$, $\overline{\mathbf{p_1p_3}}$, e $\overline{\mathbf{p_1p_2}}$ respectivamente:

1) $\mathbf{m}_1 \cap \mathbf{m}_2 \cap \mathbf{m}_3 = \{\mathbf{c}\}$

2) $\overline{\mathbf{cp}_1} \equiv \overline{\mathbf{cp}_2} \equiv \overline{\mathbf{cp}_3}$

Seja $\mathbf{c}$ o ponto tal que:

$\mathbf{m}_2 \cap \mathbf{m}_3 = \{\mathbf{c}\}$

$\mathbf{c} \in \mathbf{m}_2 \Longrightarrow \overline{\mathbf{cp}_1} \equiv \overline{\mathbf{cp}_3}$

$\mathbf{c} \in \mathbf{m}_3 \Longrightarrow \overline{\mathbf{cp}_1} \equiv \overline{\mathbf{cp}_2} \qquad \Longrightarrow \mathbf{cp}_1 \equiv \mathbf{cp}_2 \equiv \mathbf{cp}_3$

Logo:

1) $\mathbf{m}_1 \cap \mathbf{m}_2 \cap \mathbf{m}_3 = \{\mathbf{c}\}$

2) $\overline{\mathbf{cp}_1} \equiv \overline{\mathbf{cp}_2} \equiv \overline{\mathbf{cp}_3}$

As coordenadas de $\mathbf{c}$ são obtidas conforme as equações descritas a seguir:

$$\mathbf{c}_x = \frac{\begin{vmatrix} (x^2_{\mathbf{p_3p_2}} + y^2_{\mathbf{p_3p_2}}) & y_{\mathbf{p_3p_2}} \\ (x^2_{\mathbf{p_1p_2}} + y^2_{\mathbf{p_1p_2}}) & y_{\mathbf{p_1p_2}} \end{vmatrix}}{2\begin{vmatrix} x_{\mathbf{p_3p_2}} & y_{\mathbf{p_3p_2}} \\ x_{\mathbf{p_1p_2}} & y_{\mathbf{p_1p_2}} \end{vmatrix}} \tag{3.15}$$

$$\mathbf{c}_y = \frac{\begin{vmatrix} x_{\mathbf{p_3p_2}} & (x^2_{\mathbf{p_3p_2}} + y^2_{\mathbf{p_3p_2}}) \\ x_{\mathbf{p_1p_2}} & (x^2_{\mathbf{p_1p_2}} + y^2_{\mathbf{p_1p_2}}) \end{vmatrix}}{2\begin{vmatrix} x_{\mathbf{p_3p_2}} & y_{\mathbf{p_3p_2}} \\ x_{\mathbf{p_1p_2}} & y_{\mathbf{p_1p_2}} \end{vmatrix}} \tag{3.16}$$

Onde:

$$x_{\mathbf{p}_1\mathbf{p}_2} = \mathbf{p}_{1x} - \mathbf{p}_{2x}, \quad x_{\mathbf{p}_3\mathbf{p}_2} = \mathbf{p}_{3x} - \mathbf{p}_{2x} \tag{3.17}$$

e

$$y_{\mathbf{p}_1\mathbf{p}_2} = \mathbf{p}_{1y} - \mathbf{p}_{2y}, \quad y_{\mathbf{p}_3\mathbf{p}_2} = \mathbf{p}_{3y} - \mathbf{p}_{2y} \tag{3.18}$$

O raio da cincunferência que circunscreve o triângulo segue:

$$\mathbf{R}_{circ} = \frac{1}{2}\sqrt{\frac{(\mathbf{d}_1 + \mathbf{d}_2)(\mathbf{d}_2 + \mathbf{d}_3)(\mathbf{d}_3 + \mathbf{d}_1)}{\mathbf{c}}} \tag{3.19}$$

de maneira que:

$$\begin{aligned}
\mathbf{d}_1 &= (\mathbf{p}_3 - \mathbf{p}_1)(\mathbf{p}_2 - \mathbf{p}_1) && (3.20)\\
\mathbf{d}_2 &= (\mathbf{p}_3 - \mathbf{p}_2)(\mathbf{p}_1 - \mathbf{p}_2) && (3.21)\\
\mathbf{d}_3 &= (\mathbf{p}_1 - \mathbf{p}_3)(\mathbf{p}_2 - \mathbf{p}_3) && (3.22)\\
\mathbf{c} &= \mathbf{d}_1\mathbf{d}_2 + \mathbf{d}_2\mathbf{d}_3 + \mathbf{d}_3\mathbf{d}_1 && (3.23)\\
\mathbf{p}_i &= (\mathbf{p}_{ix}, \mathbf{p}_{iy}) && (3.24)
\end{aligned}$$

3.6 Estrutura de Dados

3.6.1 Introdução

A maneira como...

3.6.2 Fundamentação

Um grafo $\mathbf{G}(V, E)$ é um conjunto finito não-vazio V de *vértices*, e um conjunto E de *aretas*, que são formadas a partir de pares não-ordenados de vértices distintos de V. Uma aresta $\mathbf{e} = (\mathbf{u}, \mathbf{v})$ é constituida pelos vértices $\mathbf{u}$ e $\mathbf{v}$, que são seus extremos.

O número de *vétices*, *arestas* e *faces* em qualquer subdivisão planar, dados respectivamente pr *nv*, *ne* e *nf*, se relacionam combinatoriamente através da relação de Euler dada pela equação 3.25.

$$nv + nf - ne = 2 \tag{3.25}$$

Num *grafo planar* cada face é formada por, no mínimo, três arestas, cada aresta possui exatamente dois vértices e é adjacente a exatamente duas faces. Logo, $2ne \geq 3nf$, e, ao se substituir esta desigualdade na equação 3.25, obtém-se as inequações 3.26, 3.27 e 3.28, que são sempre válidas. Se for acrescentada a restrição de que cada *vértice* possui pelo menos três *arestas* incidentes, então, as inequações 3.29, 3.30 e 3.31 também são válidas.

Um *grafo* $\mathbf{G}(V, E)$ é dito *conexo* quando existe uma sequência de arestas entre dois vértices quaisquer de V. Dado um conjunto $\mathbf{S}$ e um subconjunto $\mathbf{S'}(\mathbf{S'} \subseteq \mathbf{S})$, diz-se que $\mathbf{S'}$ é *maximal* em relação a uma determinada propriedade $\mathbf{M}$ quando $\mathbf{S'}$ satisfaz a propriedade $\mathbf{M}$ e não existe outro subconjunto $\mathbf{S''}$ que contém $\mathbf{S'}$ e também satisfaz a propriedade $\mathbf{M}$. Muitos outros resultados teóricos sobre teoria dos grafos podem ser encontrados em (SZWARCFITER, 1984).

$$ne \leq 3nv - 6 \tag{3.26}$$

$$nf \leq \frac{2}{3}ne \tag{3.27}$$

$$nf \leq 2nv - 4 \tag{3.28}$$

$$nv \leq \frac{2}{3}ne \tag{3.29}$$

$$ne \leq 3nf - 6 \tag{3.30}$$

$$nv \leq 2nf - 4 \tag{3.31}$$

3.6.3 Estrutura de Dados baseada em Fronteira

Todos os modelos de fronteiras representam *faces* em termos de nós explícitos numa estrutura de dados de fronteiras. Isso, possibilita muitas alternativas para representação da geometria e da topologia de um modelo de fronteiras, algum dos quais são descritos

a seguir. Para deixar mais simples e mais claro, devemos ilustrá-los baseados na representação do bloco mostrado na figura 11.

Figura 11: Orientação das *arestas* relacionada com o *loop* exterior do objeto.

3.6.3.1 Modelo de fronteiras baseada em *vértice.*

Note que a simples representação da figura 12 lista os *vértices* de cada *face* em uma ordem consistente (sentido horário) como o visto do lado de fora do cubo. Essa orientação robusta é útil em muitos algoritmos. Por exemplo, em linhas ocultas ou numa superfície remota, permite a eliminação de *faces traseiras* na base das *faces* normais através dos ponteiros consistentemente apontando para elas. No caso da figura 12 as *faces* f_1, f_4 e f_5 podem ser imediatamenete descartadas pelas linhas ocultas e um algoritmo de remoção de superfície.

A representação da figura 12 não inclui todas informações da superfície; como todas as *faces* são planares, suas geometrias são completamente definidas através das coordenadas dos *vértices*. Por outro lado, se a equação da *face* é necessária para cálculo numérico (digamos, para sombreamento) essa informação também podem ser incorporadas junto às *faces*.

Figura 12: Modelo de fronteira baseado em *vértice* segundo o modelo 11.

3.6.3.2 Modelo de fronteira baseada em *arestas*

Um modelo de fronteiras baseado em *arestas* representa uma fronteira de *face* em uma sequência fechada de *arestas*, ou por um pequeno *loop*. Os *vértices* das *faces* estão representados através das *arestas*. Essa aproximação nos leva ao modelo da figura 13.

Figura 13: Modelo de fronteira baseado em *aresta* de acordo com a figura 11.

3.7 Primitivas Topológicas

3.7.1 Operadores de Euler

De acordo com Braid (BRAID I. C.; HILLYARD, 1980) cinco operadores topológicos são necessários para manipulação dos modelos, porém para maior prática no desenvolvimento de um sistema, mais operadores podem ser implementados. As seis entidades envolvidas no modelo planar utilizado são: *vértices*, *arestas*, *faces*, *ciclos*(*loops*), e corpos ou shell que são baseados na estrutura de dados *winged-edge modificada*. Em nossas implementação utilizamos primitivas topológicas para executar atualizações durante o processo construtivo da malha e garantindo sua consistência e robustez.

3.7.2 As Relações entre as Entidades Topológicas

Há relações de adjacência entre as entidades topológicas *vértice*, *aresta* e *face* encontradas em uma *subdivisão planar*, a Figura 14 representa essa situação. Cada uma dessas relações é representada por um par de letras, em que a segunda identifica o conjunto de entidades incidentes ou adjacentes à primeira. Assim, **VF** e **FF** significam, respectivamente, o conjunto de *faces* incidentes a um *vértice* dado e o conjunto de *faces* adjacentes a uma *face* dada.

Segue as nove relações de adjacência possíveis em toda estrutura topológica:

- **VV**: dado um *vértice*, encontrar os *vértices* adjacentes a ele. Dois *vértices* são adjacentes se estão conectados por uma *aresta*;
- **VE**: dado um *vértice*, encontrar as *arestas* adjacentes. Uma *aresta* é adjacente a um *vértice* se o *vértice* é seu extremo;

Todas as nove relações podem ser obtidas de forma muito simples na estrutura *winged-edge modificada*, que foi a adotada na implementação, com o auxílio de apenas duas primitivas topológicas, descritas abaixo ilustradas na Figura 15.

ccw_nev(v, e, ne) Dado um *vértice* **v** e uma *aresta* **e**, esta primitiva retorna a *aresta* **ne**, que é a próxima *aresta* incidente a **v** no sentido anti-horário depois de **e**.

ccw_nel(l, e, ne) Dado um *ciclo* **l** e uma de suas *arestas* **e**, esta primitiva retorna a *aresta* **ne**, que é a *aresta* seguinte a **e** quando se percorre o *ciclo* **l** no sentido anti-horário.

A seguir, apresentam-se formalmente todas as componentes no modelo hierárquico topológico:

- **vértice:** representa um simplexo de dimensão zero, que é um ponto no espaço euclidiano.
- **aresta:** representa um segmento de curva limitado por dois *vértices*, não necessariamente distintos, e *homeomorfo* ao intervalo [0,1]. Na implementação que estamos realizando, uma *aresta* será sempre um segmento de reta, podendo pertencer a um ou no máximo dois *ciclos*.

Figura 14: Relações de adjacências entre *aresta*, *vértice* e *face*, entidades presentes na estrutura.

Figura 15: Primitivas utilizadas para se obter todas as relações de adjacências entre as entidades *aresta*,*vértice* e *face*.

Capítulo 4

Triangulação de Delaunay

Em alguns casos, há mais de uma *Triangulação de Delaunay* para um conjunto de pontos, estes casos são chamados de degenerados. Um exemplo de caso degenerado é mostrado na Figura 16, onde há quatro pontos localizados nos *vértices* de um quadrado. Neste caso há um único *vértice* de Voronoi localizado no centro do quadrado, e está associado com quatro regiões de Voronoi. Duas triangulações são possíveis com este diagrama e ambas são válidas. Na prática, os casos degenerados devem ser resolvidos fazendo uma escolha entre as triangulações válidas.

Figura 16: Casos degenerados.

Figura 17: a) Conjunto de pontos distribuidos arbitrariamente no plano; b) Fecho convexo do conjunto de pontos em a) c) Triangulação de Delaunay incidente sobre o conjunto de pontos a); d) Diagrama de Voronoi do conjunto de pontos a).

Para ilustrar a base geométrica da *Triangulação de Delaunay* é conveniente considerar sua dupla geometria, o diagrama de Voronoi. Considere um conjunto de pontos no plano como mostrado na figura 17 a). O diagrama de Voronoi associado com estes pontos é mostrado pelas linhas tracejadas na figura 17 d). A *Triangulação de Delaunay* é mostrada na figura 17 c).

Figura 18: Combinação de todos os problemas geométricos listados na figura 17. O conjunto de pontos é exatamente o mesmo da figura 17 a).

4.1 Definições

Seja $P = \{p_1, p_2, ..., p_n\}$ um conjunto de n ($n \geq 3$) pontos distintos do plano pertencentes a R^n.

Definição 3 *O Diagrama de Voronoi de P é a subdivisão do plano em n células, para cada ponto em P, com a propriedade que um ponto q pertence a célula correspondente ao ponto p_i se e somente se*

$$dist(\mathbf{q}, \mathbf{p}_i) < dist(\mathbf{q}, \mathbf{p}_j) \tag{4.1}$$

para cada $p_j \in P$ com $i \neq j$.

Denotamos o Diagrama de Voronoi por $V(P)$. A célula que corresponde ao ponto p_i, denotamos por $V(p_i)$.

Definição 4 *Consideremos dois pontos p_1 e p_2. Definimos o bisector perpendicular do segmento $\overline{p_1 p_2}$ como sendo $B(p_1, p_2) = B_{12}$. Então cada ponto x sobre B_{12} está equidistante de p_1 e p_2.*

4.2 Propriedades

Veremos a seguir algumas características importantes da *Triangulação de Delaunay* que pode ser encontrada em (MUSIN, 1997), (O'ROURKE, 1994). Sendo P um conjunto de pontos e chamaremos de $D(P)$ a *Triangulação de Delaunay* do respectivo conjunto de pontos P:

- Se P é um conjunto finito $\in R^2$ e seja abc 3 pontos que compõem um círculo, abc é um triângulo de Delaunay se o círculo não conter nenhum outro ponto;
- Entre todas as triangulações de um dado conjunto de pontos, a Triangulação de Delaunay maximiza lexicograficamente o menor dos ângulos de qualquer triângulo;
- Uma *aresta* obedece à regra de Delaunay se:

- Pertence a um único triângulo e fica na fronteira da região convexa ou;

- Pertence a 2 triângulos **abc** e **abd** sendo que **d** fica no exterior do círculo determinado por **abc.**

A partir dessa postura citaremos alguns teoremas e lemas importantes, inclusive alguns obtidos em Sibson(GREEN P.J.; SIBSON, 1978), que farão uma pequena explanação entre as primitivas e a triangulação e também destacará características relevante das propriedades da triangulação.

Teorema*(Círculo Circunscrito): A Circunferência Circunscrita em qualquer triângulo de Delaunay não contém nenhum outro ponto da triangulação.*

Corolário: *A pré-Triangulação de Delaunay é não-degenerada, e deste modo é uma triangulação, unicamente caracterizada pelos equiângulos locais. O resultado é afirmado por Green e Sibson(GREEN P.J.; SIBSON, 1978) e a presente nota validou a sua afirmação. Green e Sibson descreveram um algoritmo para construir apenas a Triangulação de Delaunay; este algoritmo foi implementado e posteriormente executado com sucesso em grande escala - superando 10.000 dados apresentados - precisa de pouco espaço para ser armazenado e é muito rápido. O presente teorema permitiu considerarmos este como um dos meios de se construir equiângulos localizados nas triangulações para interpolação. Diferente do algoritmo (apresentado por Lawson(LAWSON, 1972)) que trabalhava diretamente em termos de ângulos máximos-mínimos, critérios concorrentes para adquirir direto os recursos aproximados no algoritmo Green-Sibson.*

1 *V e D são duais, isto é, para $P, P' \subseteq S$, $V(P)$ é uma face de $V(P')$ se $D(P)$ é uma face de $D(P')$.*

2 *Se $P \subseteq S$, $V(P)$ não possuirá fronteira se cada lado de R está sobre a fronteira do Fecho Convexo de S.*

Lema 3 *: Seja T_L e T_R triangulações de Delaunay com um conjunto de L e R vértices. Então podemos sempre construir uma triangulação de Delaunay T para o conjunto $L \cup R$ tais que cada aresta de T que não está em T_L ou T_R tem um ponto final em L e um em R.*

4.3 Como gerar uma Triangulação Delaunay

Aqui será abordada...

4.3.1 A técnica do *flip-edge*

Para verificar se o triângulo satisfaz a regra de Delaunay primeiramente passeamos pela lista duplamente encadeada, através das primitivas **pcw** e **pccw**, de *arestas*(**e**), verificando se cada uma de suas *faces*(f1,f2) possui algum *vértice* no interior da circunferência que circunscreve a *face* triangular. Observe a Figura 19 b) para melhor compreensão.

Figura 19: a) Triângulos cuja aresta comum entre eles não passou no teste do *InCírculo*; b) Círculo formado pelos pontos pertencentes aos dois triângulos que revela um quarto ponto em seu interiores; c) Remoção da aresta ilegal; d) Após o *flip*(troca da aresta do quadrilátero) ambos triângulos satisfazem o critério de Delaunay.

Figura 20: a) Pelo fato de existirem três pontos cocirculares $\mathbf{p}_1$, $\mathbf{p}_2$ e $\mathbf{p}_3$, trata-se de um caso degenerado do *flip-edge* e deve ser tratado de maneira especial. b) como não se trata de uma região convexa nesse triângulo não se aplica a técnica do *flip-edge*.

Descrição do pseudo-código *flip-edge.*

Algoritmo 1 flip-edge

Entrada: Dois triângulos que não passaram no teste do Incírculo.
Saida: A diagonal do quadrilátero formado pelos triângulos trocada e o teste do Incírculo satisfeito plenamente.
para Os triângulos que não satisfazem o Incírculo **faça**
Remover uma *aresta* uma *face* e um *loop*
Realizar atualizações topológicas
Criar uma nova *aresta* na outra diagonal da *face* do quadrilátero resultante um novo *loop* e uma nova *face*.
Atualizar novamente a topologia das *arestas* da *face* e do *loop*.
fim para

Com essa técnica conseguimos triângulos que satisfazem o *Critério de Delaunay*, semelhante ao critério MaxMin(seção seguinte), com um número reduzido de operações. Assim poderemos tanto obter uma Triangulação de Delaunay a partir de uma triangulação arbitrária como também usando o algoritmo de inserção verificar durante a inserção dos pontos se os triângulos são Delaunay.

4.4 Maximizando os ângulos mínimos.

Seja T uma triangulação de um conjunto de pontos S, e suponha que ela tenha m triângulos. Consideremos os $3m$ ângulos dos triângulos em T, ordenados em ordem

crescente. Seja $\alpha_1, \alpha_2, \ldots, \alpha_{3m}$ o resultado da ordenação dos ângulos, consequentemente, $\alpha_i \leq \alpha_j$, para $i < j$ e chamaremos de $A(T) = (\alpha_1, \alpha_2, \ldots, \alpha_{3m})$ o vetor-ângulo de T. Seja T' uma triangulação do mesmo conjunto de pontos S, e seja $A(T') = (\alpha'_1, \alpha'_2, \ldots, \alpha'_{3m})$ seu vetor-ângulo. Dizemos que o vetor-ângulo de T é maior que o de T' se $A(T)$ é lexicograficamente maior que $A(T')$, ou, em outras palavras, se existe um índice i com $1 \leq i \leq 3m$ de forma que:

$$\begin{cases} \alpha_j = \alpha'_j & j \leq i \\ \alpha_i > \alpha'_i \end{cases} \tag{4.2}$$

Denotamos como $A(T) > A(T')$. Uma Triangulação de T é chamada de ângulo-ótimo se $A(T) \geq A(T')$ para todas as triangulações de T' de S. Note que uma triangulação que maximiza o vetor-ângulo para uma ordem lexicográfica também maximiza o menor de seus ângulos, e tal triangulação é chamada de *globalmente equiangular*.

Teorema: *Seja* **C** *um círculo, l uma linha cruzando* **C** *nos pontos* **a** *e* **b**, *e* **p**, **q**, **r** *e* **s** *pontos localizados no mesmo lado da linha l. Suponha que* **p** *e* **q** *estejam sobre* **C**, *que* **r** *esteja dentro de* **C**, *e que* **s** *esteja fora de* **C**. Figura 21. Então:

$$\angle\mathbf{arb} > \angle\mathbf{apb} = \angle\mathbf{aqb} > \angle\mathbf{asb}. \tag{4.3}$$

Figura 21: Maximização dos ângulos mínimos.

Agora considerando uma *aresta* $e = \overline{p_i p_j}$ de uma triangulação T de S. Se e não é uma *aresta* pertencente ao *Fecho Convexo*, ela é incidente a dois triângulos $\triangle(p_i p_j p_k)$ e $\triangle(p_i p_j p_l)$. Se estes dois triângulos formam um quadrilátero convexo, nós podemos ter uma nova triangulação T' simplesmente fazendo o *flip-edge* retirando $\overline{p_i p_j}$ e inserindo $\overline{p_k p_l}$. A figura 19 ilustra a ação. Logo dizemos que uma *aresta* é ilegal se conseguirmos fazer o *flip* nessa *aresta* aumentando o menor dos ângulos.

Lema: *Seja a aresta* $\overline{p_i p_j}$ *incidente aos triângulos* $p_i p_j p_k$ *e* $p_i p_j p_l$, *e seja C o círculo passando por* p_i, p_j, *e* p_k. *A aresta* $\overline{p_i p_j}$ *será ilegal se e somente se* p_l *está no inteiror de C. Além disso, se os pontos* p_i, p_j, p_k, p_l *formam um quadrilátero convexo e não estão em um círculo comum, então exatamente uma das arestas* $\overline{p_i p_j}$ *e* $\overline{p_k p_l}$ *é uma aresta ilegal.* Figura 19(b).

4.5 Aplicações Práticas da Triangulação de Delaunay

Muitas aplicações...

$$MST \subseteq RNG \subseteq GG \subseteq Delaunay \tag{4.4}$$

4.5.1 GG - Grabriel graph

O grafo de Gabriel é um grafo $\mathbf{G}(V, E)$ que conecta um conjunto de pontos no plano euclidiano. Dois pontos $\mathbf{p}_1$ e $\mathbf{p}_2$ são conectados por uma aresta no grafo de Gabriel se o círculo contendo o diâmetro $\overline{\mathbf{p}_1\mathbf{p}_2}$ não possui nenhum outro ponto no seu interior figura 22 a), caso haja um terceiro ponto no interior da circunferência figura 22 b), a aresta não fará parte do grafo de Gabriel. Em qualquer outra dimensão o grafo de Gabriel conecta dois pontos formando o diâmetro de uma esfera vazia. O grafo de Gabriel foi apresentada por K. Ruben Gabriel e intitulada por Robert R. Sokal em 1969 (SOKAL R.R; GABRIEL, 1969). Esse grafo é um sub-grafo da Triangulação de Delaunay e com a triangulação dada pode ser obtido em tempo linear (MATULA D.W.; SOKAL, 1980). Esse gráfico contém a MST e o RNG na instância de um esqueleto-beta. Sua aplicabilidade também se encontra na área de redes, na montagem de redes wireless. Um exemplo de grafo pode ser visto na figura 23.

Figura 22: a) circunferência que contém o diâmetro de $\overline{\mathbf{p}_1\mathbf{p}_2}$, com um terceiro ponto $\mathbf{p}_3$ no exterior da circunferência. b) circunferência que contém o diâmetro de $\overline{\mathbf{p}_1\mathbf{p}_2}$, porém com um terceiro ponto $\mathbf{p}_3$ no interior da circunferência

Figura 23: Gabriel Graph de um determinado conjunto de pontos em destaque sobre a Triangulação de Delaunay - linha pontilhadas

Capítulo 5

Como Obter a Triangulação de Delaunay

5.1 Introdução

Introdução...

5.2 Algoritmo de Inserção Randomizado

Veremos abaixo o pseudo-código desenvolvido do algoritmo a ser implementado:

Figura 24: Ilustração das atualizações topológicas na inserção do primeiro ponto **V**.

5.2.1 Escolha adequada das coordenadas de um supertriângulo

No algoritmo incremental de Lawson, citamos a necessidade de criar-se um grande triângulo que contenha todos os pontos da triangulação, as coordenadas dos vértices do grande triângulo segundo o criterio devem ser escolhidas como sendo os maiores valores possíveis. Porém o ideal é que escolhamos números adequados para que o supertriângulo contenha todos os pontos que serão processados e que não tenha coordenadas muito exageradas. Então o que faremos será escolher os pontos $\mathbf{p}-1$, $\mathbf{p}-2$ e $\mathbf{p}-3$ e modificar o teste feito para arestas ilegais, funcionar como se estivéssemos os escolhido com coordenadas bem grandes. Uma sugestão feita em (BERG; OVERMARS; SCHWARZKOPF, 1977) é que a geração de pontos seja feita no interior de um quadrilátero de lado M e que o supertriângulo deverá envolver o quadrilátero tendo coordenadas de distância $3M$

Algoritmo 2 Algoritmo incremental

Entrada: Um conjunto P de n pontos no plano

Saida: A Triangulação de Delaunay de P. Seja $\mathbf{p}-1$, $\mathbf{p}-2$ e $\mathbf{p}-3$ um conjunto de três pontos que formam uma grande triângulos imaginário que contem P. Inicializar a triangulação $\mathcal{T}$ que possui apenas um triângulo simples $\mathbf{p}-1$, $\mathbf{p}-2$ e $\mathbf{p}-3$

para r $\Longleftarrow$ 1 **até** n **faça**

Insere-se um novo ponto em $\mathcal{T}$

Encontre o triângulo $p_ip_jp_k$ que contém o ponto p.

Se p está no interior do triângulo $p_ip_jp_k$ **então**

Ligue o ponto p aos três *vértices* formados criando *arestas* e dividindo $p_ip_jp_k$ em três triângulos.

flip $(p, \overline{p_ip_j})$

flip $(p, \overline{p_jp_k})$

flip $(p, \overline{p_kp_i})$

senão

Adicione arestas de p até $\mathbf{p}_k$ e ao terceiro *vértice* $\mathbf{p}_l$
do outro triângulo que é incidente a $\overline{p_ip_j}$, logo dividindo os dois
triângulos incidentes a $\overline{p_ip_j}$ em quatro.

flip $(p, \overline{p_ip_l})$

flip $(p, \overline{p_lp_j})$

flip $(p, \overline{p_jp_k})$

flip $(p, \overline{p_kp_i})$

fim Se

fim para

Remova os *vértices* p-1, p-2 e p-3 e também todas *arestas*
do supertriângulo incidentes na triangulação

($[0, 3M]$, $[3M, 0]$, $[-3M, -3M]$), embora ao invés de $3M$ alguns autores sugerem o uso de uma distância de $6M$ ($[0, 6M]$, $[6M, 0]$, $[-6M, -6M]$) figura 25. Uma outra sugestão é utilizar um circulo de raio M, e gerar os pontos no interior do quadrilátero inscrito na circunferência segundo a figura 26

Figura 25: Supertriângulo envolvendo o quadrilátero com coordenadas $3M$.

Figura 26: Circunferência de raio $3M$ inscrita no Supertriângulo envolve um quadrilátero onde serão gerados os pontos.

Capítulo 6

Conclusão

Considerando a amplitude dos algoritmos existentes, optamos por aqueles pertencentes ao paradigma de *inserção*, também referenciado na literatura por *incremental* (BERG; OVERMARS; SCHWARZKOPF, 1977). As maiores vantagens dos algoritmos pertencentes a essa classe, são: em primeiro lugar, a sua simplicidade algorítmica e, em segundo, a possibilidade deles serem facilmente generalizados para outras dimensões, em particular, o R^3.

Para ser mais preciso, foram implementados dois algoritmos. O primeiro deles tem como base o trabalho de Watson (WATSON, 1981). Entretanto, inúmeras variantes foram incorporadas, como por exemplo, o princípio de Lawson (LAWSON, 1972) para manutenção do critério de Delaunay e o conceito de randomização sobre o conjunto de pontos de entrada (GUIBAS; KNUTH; SHARIR, 1985), que combinadas com eficientes técnicas de busca, aumentam significativamente a perormance dese algoritmo. Só para exemplificas, atualmente, mesmo sem ter finalizado muitas das otimizações necessárias, conseguimos obter triangulações da ordem de 10^5 em torno de 25 segundos, o que é bastante razoável para um trabalho dessa natureza.

6.1 Futuros Trabalhos

E agora..... ?

6.2 Como Inserir Figuras Obtidas de Outras Fontes

É importante ressaltar que o padrão da característica biométrica obtida no momento da verificação ou identificação não será idêntico ao armazenado previamente pelo sistema. O sistema biométrico deve comparar as duas representações da característica e verificar a correlação existente entre elas. A correlação geralmente é determinada pelo grau de similaridade, um valor normalizado que pode variar entre 0 e 1. Quanto mais se aproximar de 1, maior é a probabilidade dos dois padrões terem sido obtidos da mesma pessoa. Por outro lado, quanto mais próximo for de 0, menor é esta possibilidade. O que determina a aceitação ou a rejeição é um limiar. Se a comparação tiver um grau de similaridade maior ou igual a este limiar, ocorre uma aceitação, caso contrário, há uma rejeição (JAIN K. A.; BOLLE, 2006).

Figura 27: Situações possíveis em sistemas biométricos.
Fonte: adaptada a partir da referência (JAIN K. A.; BOLLE, 2006).

Há quatro situações possíveis no processo de autenticação biométrica. Estas possibilidades são ilustradas na Figura 27 e são denominadas: Correta Aceitação (CA), Correta Rejeição (CR), Falsa Aceitação (FA) e Falsa Rejeição (FR). CA e CR caracterizam o funcionamento esperado do sistema, pois um usuário legítimo é aceito e um impostor é rejeitado, respectivamente, nestas ocasiões. Já FR e FA constituem situações de erro do sistema. No primeiro caso, uma identidade genuína é considerada falsa e no segundo, um impostor é tratado como um usuário autêntico.

FA e FR são quantificadas, respectivamente, pela taxa de falsa aceitação (FAR - *False Acceptance Rate*) e pela taxa de falsa rejeição (FRR - *False Rejection Rate*). Estas medidas referem-se a probabilidade de ocorrer uma falsa aceitação e uma falsa rejeição em um algoritmo de verificação, e são constantemente usadas para avaliar o desempenho de sistemas biométricos.

A taxa de falsa aceitação (FAR) e a taxa de falsa rejeição (FRR) são mutuamente dependentes. A diminuição de uma aumenta a outra. A Tabela 1 apresenta as taxas médias de desempenho FAR e FRR de alguns sistemas biométricos típicos encontrados

1	2	3
4	5	6
7	8	9

Tabela 1: Taxas de Desempenho

no mercado (PINHEIRO, 2008).

Referências

BERG, M. K.; OVERMARS, M.; SCHWARZKOPF, M. O. *Computational geometry*: algorithms and applications. [S.l.]: Springer-Verlag, 1977.

BRAID I. C.; HILLYARD, R. C. S. I. A. Stepwise construction of polyhedrain geometric modeling. In: BRODLIE, K. (Ed.). *Mathematical methods in Computer Graphics and Design*. [S.l.]: Academic Press, 1980. p. 123–141.

GREEN P.J.; SIBSON, R. Computing Dirichlet tessellations in the plane. *Computer Journal*, v. 2, n. 21, p. 168–173, 1978.

GUIBAS, L.; KNUTH, D.; SHARIR, M. Randomized incremental construction of Delaunay and Voronoi diagrams. *ACM Transactions on Graphics*, v. 4, n. 2, p. 75–123, 1985.

JAIN K. A.; BOLLE, R. P. S. *Biometrics*: personal identification in networked society. [S.l.: s.n.], 2006.

LAWSON, C. *Generation of a triangulation grid with applications to contour plotting*. [S.l.], 1972.

MATULA D.W.; SOKAL, R. Properties of Gabriel graphs relevant to geographic variation research and clustering of points in the plane. In: *Geographic Anal*. [S.l.: s.n.], 1980. v. 12, p. 205–222.

MUSIN, O. Properties of the Delaunay triangulation. In: *PROCEEDINGS OF THE ANNUAL SYMPOSIUM ON COMPUTATIONAL GEOMETRY*. Nice, France: [s.n.], 1997. v. 13, p. 424–426.

O'ROURKE, J. *Computational geometry in C*. 2. ed. [S.l.]: Cambridge University Press, 1994. 161-163 p.

PINHEIRO, M. J. *Biometria nos sistemas computacionais*: você é a senha. [S.l.: s.n.], 2008.

SIBSON, R. Locally equiangular triangulations. *Computer Journal*, v. 21, n. 3, p. 243–245, 1978.

SOKAL R.R; GABRIEL, K. A new statistical approach to geographic variation analysis. *Systematic Zoology*, v. 18, p. 259–270, 1969.

SZWARCFITER, J. *Grafos e algoritmos computacionais*. Rio de Janeiro: Editora Campus, 1984. 216 p.

WATSON, D. Computing the n-dimensional Delaunay tesselation with application to voronoi polytopes. *Computer Journal*, v. 24, n. 2, p. 167–172, 1981.

APÊNDICE A – Código Fonte 1

Código A.1: 'Código Fonte de Wolfstein - Enemy Territory'

```
1  void CG_mvZoomBinoc( float x, float y, float w, float h ) {
2          float ws = w / 640;
3          float hs = h / 480;
4
5          // an alternative.  This gives nice sharp lines at the
6      // expense of a few extra polys
7          if ( cgs.media.binocShaderSimple ) {
8                  CG_DrawPic( x, y, 640.0f * ws, 480.0f * ws,
9              cgs.media.binocShaderSimple );
10         }
11
12         CG_FillRect( x + 146.0f * ws, y + 239.0f * hs,
13             348.0f * ws, 1, colorBlack );
14
15         CG_FillRect( x + 188.0f * ws, y + 234.0f * hs,
16             1, 13.0f * hs, colorBlack );    // ll
17         CG_FillRect( x + 234.0f * ws, y + 226.0f * hs,
18             1, 29.0f * hs, colorBlack );    // l
19         CG_FillRect( x + 274.0f * ws, y + 234.0f * hs,
20             1, 13.0f * hs, colorBlack );    // lr
21         CG_FillRect( x + 320.0f * ws, y + 213.0f * hs,
22             1, 55.0f * hs, colorBlack );    // center
23         CG_FillRect( x + 360.0f * ws, y + 234.0f * hs,
24             1, 13.0f * hs, colorBlack );    // rl
25         CG_FillRect( x + 406.0f * ws, y + 226.0f * hs,
26             1, 29.0f * hs, colorBlack );    // r
```

```
27          CG_FillRect( x + 452.0f * ws, y + 234.0f * hs,
28              1, 13.0f * hs, colorBlack );    // rr
29  }
```

APÊNDICE B – Código Fonte 2

Código B.1: 'Código Fonte de Wolfstein - Enemy Territory'

```
1  void AngleVectors (const vec3_t angles, vec3_t forward,
2      vec3_t right, vec3_t up)
3  {
4          float angle;
5          static float sr, sp, sy, cr, cp, cy;
6
7          angle = angles[YAW] * (M_PI * 2 / 360);
8          sy = sin(angle);
9          cy = cos(angle);
10
11         angle = angles[PITCH] * (M_PI*2/360);
12         sp = sin(angle);
13         cp = cos(angle);
14
15         angle = angles[ROLL] * (M_PI*2/360);
16         sr = sin(angle);
17         cr = cos(angle);
18
19         if (forward) {
20                 forward[0] = cp * cy;
21                 forward[1] = cp * sy;
22                 forward[2] = -sp;
23         }
24         if (right) {
25                 right[0] = (-1*sr*sp*cy + -1*cr*-sy);
26                 right[1] = (-1*sr*sp*sy + -1*cr*cy);
```

```
27                  right[2] = -1 * sr * cp;
28          }
29          if (up) {
30                  up[0] = (cr*sp*cy + -sr*-sy);
31                  up[1] = (cr*sp*sy + -sr*cy);
32                  up[2] = cr*cp;
33          }
34  }
```