Anderson Oliveira dos Santos
Igor Thiago Weidman
Sergio Henrique Costa

Malhas triangulares

CURITIBA

2009

Anderson Oliveira dos Santos
Igor Thiago Weidman
Sergio Henrique Costa

Malhas triangulares

Trabalho de Conclusão de Curso apresentado à banca examinadora da Universidade Federal do Paraná, como exigência para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. André Guedes

CURITIBA

2009

Sumário

Lista de Figuras ..... **iv**

Resumo ..... **v**

1 Introdução **1**

1.1 Motivação 1

1.2 Objetivos 1

1.3 Organização do Trabalho 2

2 Algoritmos **3**

2.1 Simplificação 3

2.1.1 Algoritmo escolhido 4

2.1.2 Resultados obtidos 6

2.2 Adição de ponto a face 10

2.2.1 Definição 10

2.2.2 Exemplo de aplicação 10

2.3 Remoção de ponto e triângulo 12

2.3.1 Ponto 12

2.3.2 Triângulo 13

3 Bibliotecas utilizadas 17

3.1 Ply 17

3.2 Tinlib 19

3.3 OpenGL 19

3.4 GTK-Glade 20

4 Manual 21

4.1 Compilação, Execução e Encerramento 21

4.2 Janela de Renderização da Malha 21

4.3 Janela de Manipulação da Malha 23

Conclusão 25

Referências Bibliográficas 26

Lista de Figuras

Figura 2.1 *Remoção de aresta* 5

Figura 2.2 *Exemplo de simplificação 1* 7

Figura 2.3 *Exemplo de simplificação 2* 8

Figura 2.4 *Exemplo de simplificação 3* 9

Figura 2.5 *Adiç?o de ponto a uma face* 11

Figura 2.6 Ponto 5 a ser retirado 12

Figura 2.7 Quadrado gerado 13

Figura 2.8 Faces retiradas 13

Figura 2.9 Malha retriangulada 14

Figura 2.10 Triângulo a ser retirado 15

Figura 2.11 Segundo ponto do triângulo 15

Figura 2.12 Terceiro ponto do triângulo 16

Figura 2.13 Resultado final da retirada 16

Figura 4.1 *Janela de Renderização da Malha* 22

Figura 4.2 *Janela de Manipulação da Malha* 24

Resumo

Este trabalho implementa diversas formas de manipulação de malhas triangulares, fazendo com que seja possível sua modificação, através de retirada de pontos e faces, adição de pontos e também da simplificação. A simplificação se dá através da retirada das menores arestas da malha, sendo possível escolher entre retirar 5% ou 20% dessas arestas.

Palavras-chave: Malhas triangulares, Simplificação, Adição de pontos, Retirada de pontos, Retirada de faces.

1 Introdução

Malhas triangulares são estruturas geométricas formadas por linhas regulares, baseadas no desenvolvimento planar de um triângulo. Malhas triangulares são amplamente utilizadas em aplicativos de desenho auxiliado por computador (CAD), sistemas de informação geográfica (GIS), jogos 3D, etc. Faz-se então necessária a possibilidade de se manipular, editar, gerar e armazenar estas malhas.

Neste trabalho focamos na aplicação de alguns métodos de manipulação e simplificação das malhas triangulares visando o desenvolvimento de uma ferramenta básica que possibilita um futuro desenvolvimento de ferramentas mais especializadas.

1.1 Motivação

A motivação para a realização deste trabalho foram as poucas fontes de informação referentes ao assunto e a existirem poucas ferramentas de manipulação para as malhas triangulares.

1.2 Objetivos

Este trabalho tem por objetivo implementar algumas formas de manipulação de malhas triangulares, sendo possível sua modificação. Estes algoritmos de manipulação são úteis para se modificar a malha para moldá-la, podendo até mesmo simplificar a malha para que se possa trabalhar com um menor número de dados.

1.3 Organização do Trabalho

Este trabalho apresenta 4 capítulos, sendo este o capitulo 1. No capitulo 2 serão apresentados os algoritmos que desenvolvemos para a manipulação das malhas triangulares. No capitulo 3 apresentamos as bibliotecas que utilizamos para nos auxiliar no desenvolvimento dos algoritmos. No capitulo 4 está o manual de utilização e compilação do trabalho.

2 Algoritmos

Neste capítulo mostraremos todos os algoritmos de manipulação da malha desenvolvidos por nós, os algoritmos são:

- Simplificação;
- Adição de ponto a uma face;
- Remoção de ponto e triângulo.

2.1 Simplificação

Atualmente ferramentas de CAD facilmente geram uma malha triangular com milhares de pontos e milhões de faces, mas o desenvolvimento de hardware para computar tal volume de dados não acompanha a evolução na complexidade de tais malhas. Muitas vezes tem-se malhas extremamente complexas em ambientes que não requerem tais nível de detalhamento. Em visualizações interativas, este problema é ainda mais agravado, pois há a necessidade de se gerar imagens com rapidez, e malhas muito detalhadas podem não ser uma boa solução, prejudicando o desempenho da aplicação. Para minimizar tal problemas, são usadas diversas técnicas para a simplificação das malhas triangulares, tais como:

- **Junção de faces coplanares (Coplanar facets merging)**

  É a simplificação de faces coplanares ou proximamente coplanares, sendo juntadas em um polígono maior e retriangularizadas. [CIGNONI; MONTANI; SCOPGNO, 1998; DEHAEMER; ZYDA, 1991; HINKER; HANSEN, 1993]

- **Decimação controlada de vértices,faces ou arestas (Controllated vertex/edge/face decimation)**

  Eliminação iterativa de componentes determinadas por algum critério de otimização, geralmente, preserva a topologia da malha. [ALGORRI; SCHMITT, 1996; RONFARD; ROSSIGNAC; ROSSIGNAC, 1996; GUÉZIEC, 1996]

- **Re-tiling**

  Neste método, vários vértices novos são inseridos na malha em posições aleatórias e então movidos ao longo da superfícies, então os vértices originais são removidos e faz-se uma nova triangularização. [TURK, 1992; CIGNONI; MONTANI; SCOPGNO, 1998]

- **Clusterização de vértices**

  Este método agrupa vértices em clusters, e para cada cluster é computado um novo vértice representativo , Baseado em proximidade geométrica, não preserva a topologia nem detalhes de formas com escala pequena. [LOW; TAN, 1997; CIGNONI; MONTANI; SCOPGNO, 1998]

2.1.1 Algoritmo escolhido

Para este trabalho escolhemos utilizar a *decimação controlada de arestas*, pois segundo nossas pesquisas, este se mostrou ser um algoritmo eficiente na taxa de redução, pelo seu bom desempenho, pela implementação e uso simples, por ser eficiente em grandes malhas e por preservar a topologia desta.

Primeiramente calculamos a distância euclidiana (Eq. 2.1) para cada aresta, a partir destas distancias criamos uma lista com as arestas que serão removidas, as escolhidas são as menores arestas da malha. Esta condição se mostrou eficiente pois pequenas arestas servem para aumentar o grau de detalhamento da malha, ao contrario das maiores, que se retiradas podem deformar a malha.

$$d(P1,P2) = \sqrt{(X1-X2)^2 + (Y1-Y2)^2 + (Z1-Z2)^2} \tag{2.1}$$

Em seguida, os vértices da lista serão retirados um de cada vez, para isso calculamos um novo vértice, exatamente no ponto médio (Eq. 2.2) desta aresta (Figura 2.1(b)), os triângulos que eram formados pela aresta retirada são removidos e os demais triângulos, que eram formados pelos vértices da menor aresta, são agora formados pela nova aresta, como na figura 2.1(c).

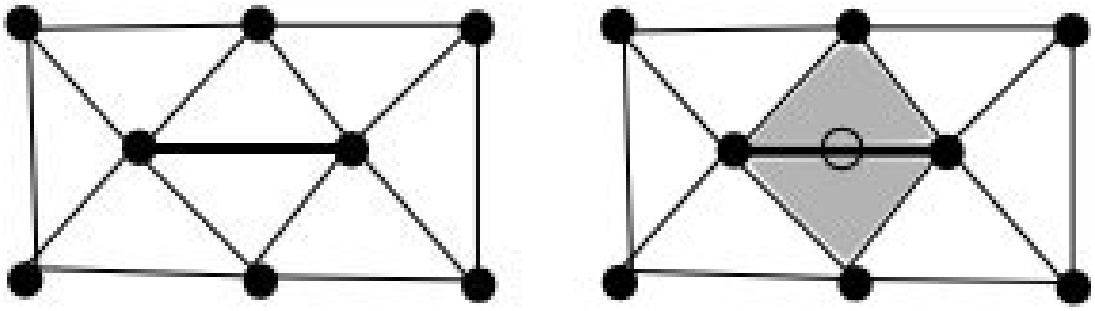
Finalizada a remoção de cada aresta presente na lista, temos uma boa simplificação de uma malha triangular, de forma simples e eficiente.

$$PM(A(X1,Y1,Z1),B(X2,Y2,Z2)) = (\frac{(X1+X2)}{2}, \frac{(Y1+Y2)}{2}, \frac{(Z1+Z2)}{2}) \tag{2.2}$$

2.1.2 Resultados obtidos

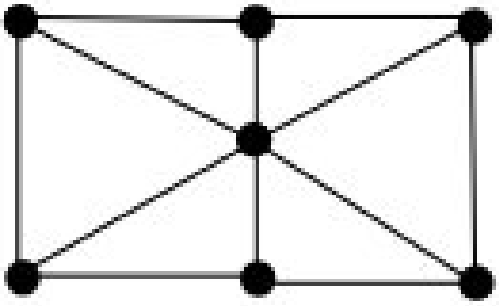
As figuras 2.2, 2.3 e 2.4, mostram os resultados obtidos por nossa equipe. A partir das malhas iniciais (2.2(a), 2.3(a) e 2.4(a)), cada malha seguinte é a simplificação de 50% da malha anterior.

As figuras mostram como o método escolhido é eficaz, visto que depois de sucessivas simplificações o formato da malha continua o mesmo.



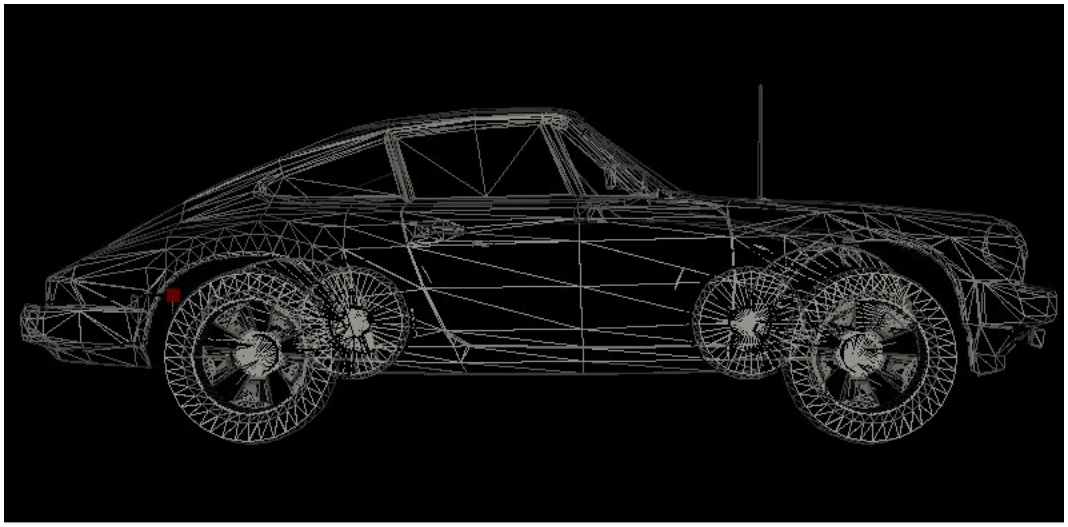
(a) Malha inicial

(b) Aresta e triângulos a serem retirados e novo vértice criado

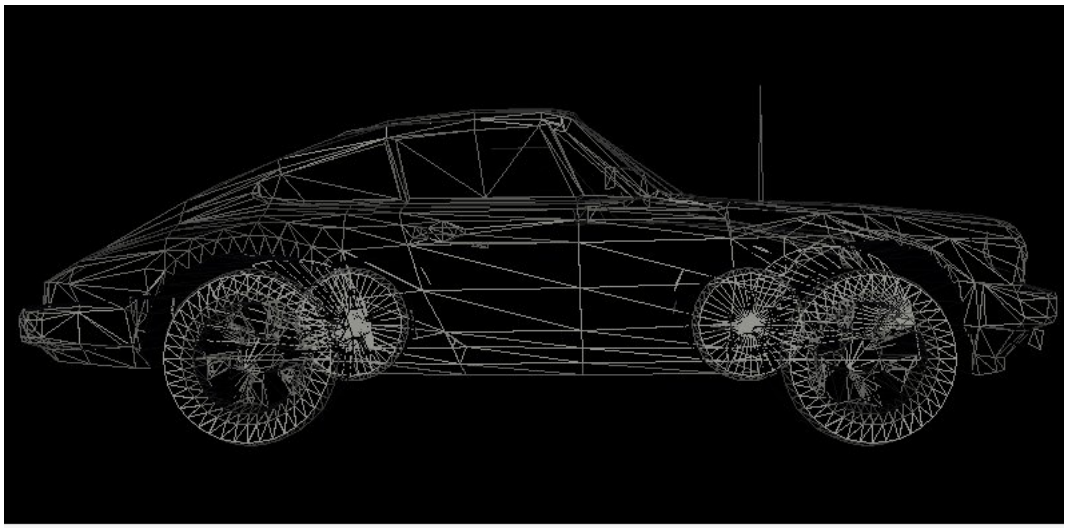


(c) Malha simplificada

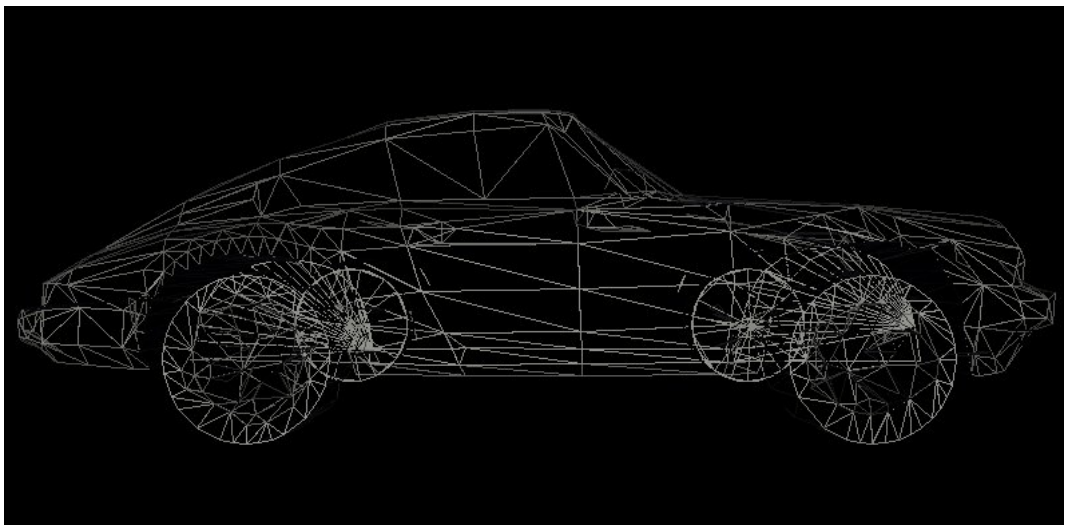
Figura 2.1: *Remoção de aresta*



(a)

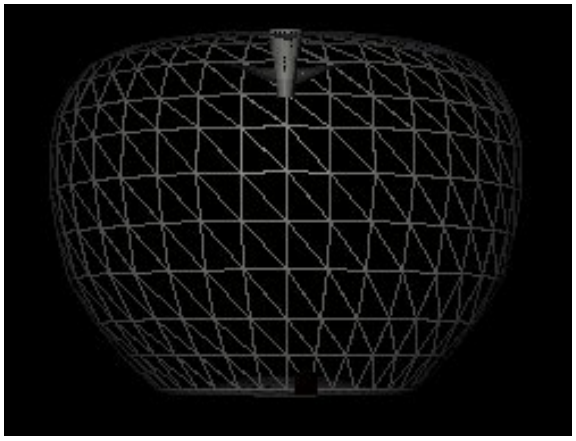


(b)

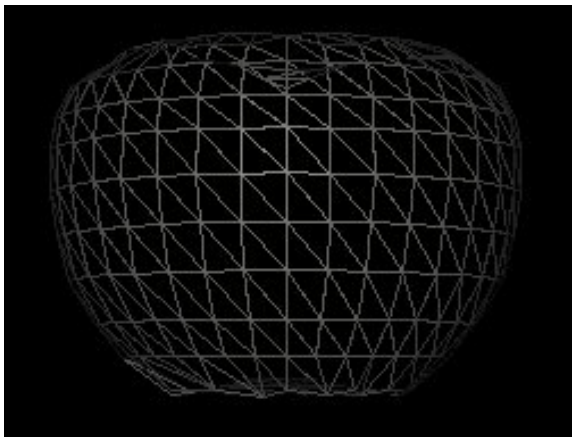


(c)

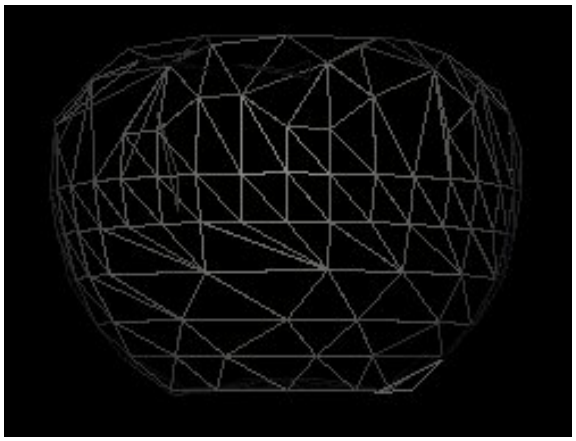
Figura 2.2: *Exemplo de simplificação 1*



(a)

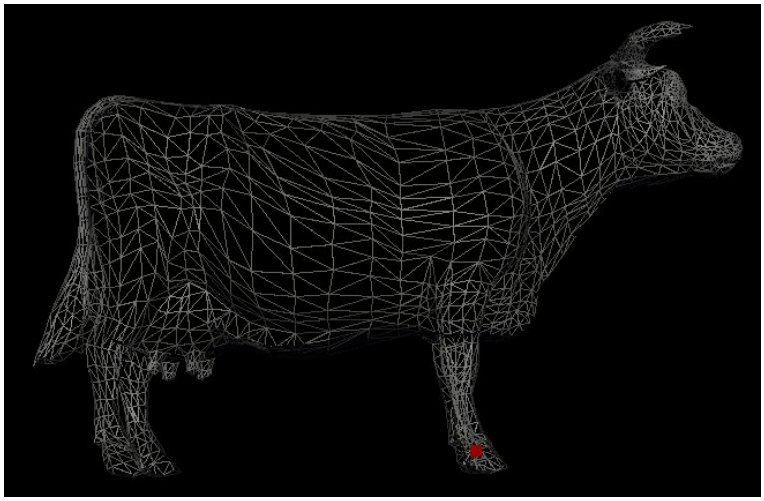


(b)

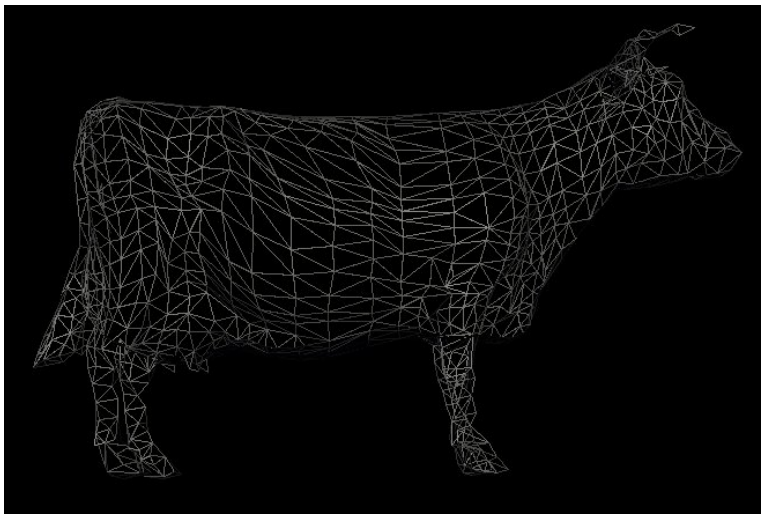


(c)

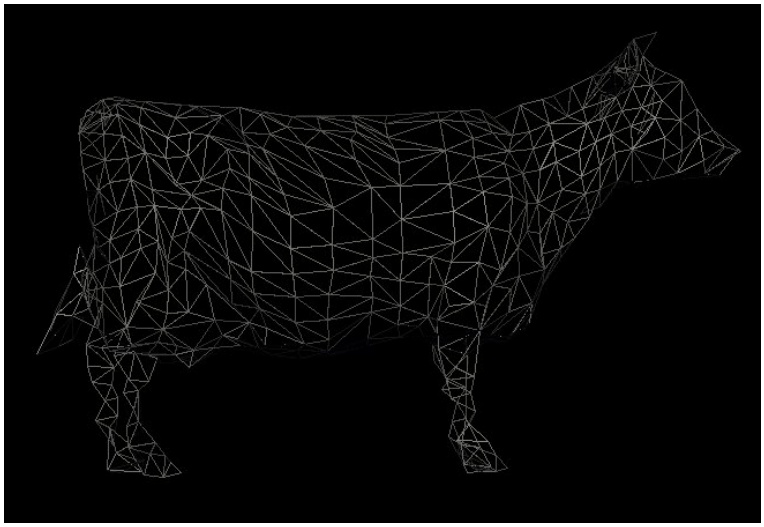
Figura 2.3: *Exemplo de simplificação 2*



(a)



(b)



(c)

Figura 2.4: *Exemplo de simplificação 3*

2.2 Adição de ponto a face

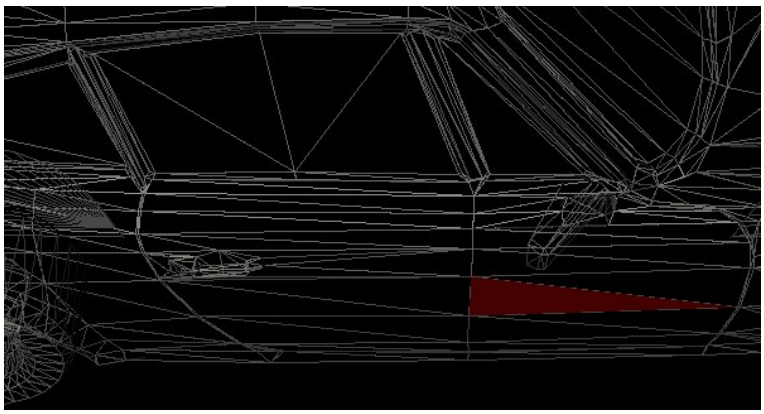
Uma forma de manipulação de malhas que garante ao usuário uma maneira fácil de aumentar ou criar detalhes em uma malha.

2.2.1 Definição

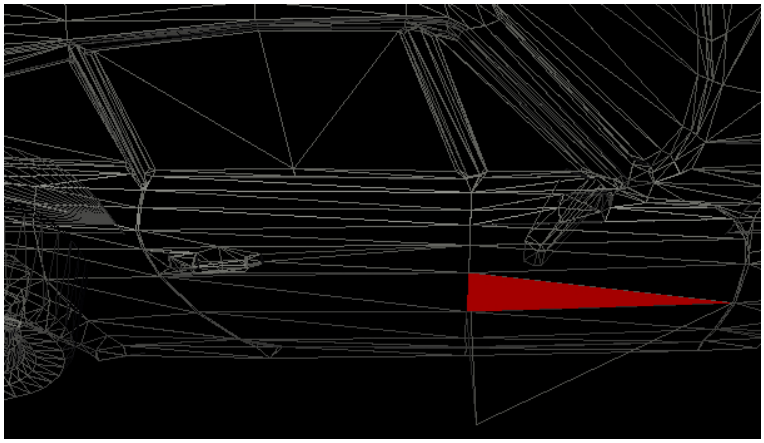
Para este operador de malha, primeiramente adicionamos um ponto sobre uma das arestas da face triangular escolhida, deslocado de uma unidade no eixo X, em seguida criamos três triângulos formados pelos vértices adjacentes da face selecionada com o ponto criado. Para finalizar a adiçao do ponto, o usuário pode então editar a posição do novo vértice.

2.2.2 Exemplo de aplicação

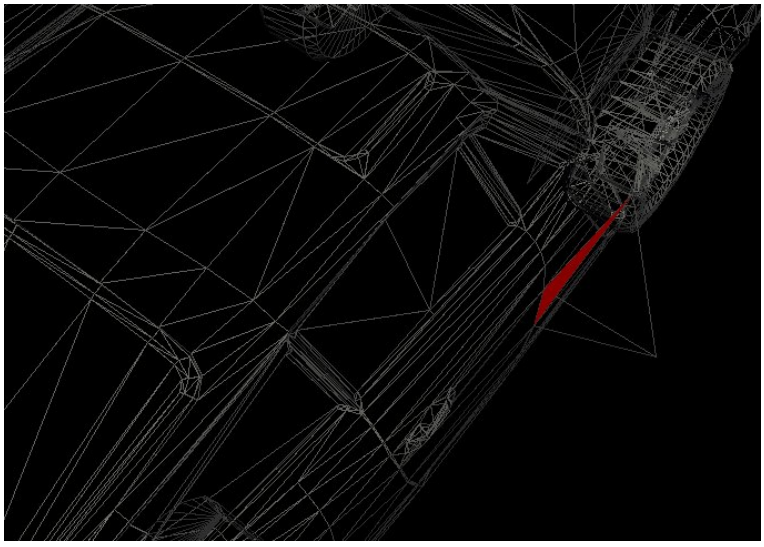
A figura 2.5 mostra o resultado desta operação em uma malha triangular.



(a) Face escolhida para a operação



(b) Resultado da operação



(c) Resultado da operação

Figura 2.5: *Adiç?o de ponto a uma face*

2.3 Remoção de ponto e triângulo

Outras duas formas implementadas de manipulação das malhas são a remoção de pontos e remoção de triângulos. Abaixo são detalhados os dois metodos.

2.3.1 Ponto

Quando um ponto é retirado, a malha possivelmente não conterá apenas triângulos, sendo necessário refazer a triangulação. Para que o impacto na malha seja o mínimo possível, fazemos a retriangulação apenas localmente, considerando apenas as faces afetadas.

Toda vez em que um ponto é retirado, a lista de vizinhos desse ponto é gerada a partir da lista de faces da malha, esta lista de vizinhos é necessária para que se saiba quais faces foram afetadas. Após esta lista de vizinhos ser gerada, são retiradas todas as faces que fazem referencia ao ponto que vai ser retirado e o ponto é então apagado da lista de pontos da malha. Utiliza-se então a biblioteca descrita na seção 3.2 para a retriangularização dos pontos da lista de vizinhos.

Exemplo de aplicação

Na figura 2.6, o ponto 5 será retirado da malha, gerando um quadrado na malha, como pode ser visto na figura 2.7, esse quadrado deve ser retirado.

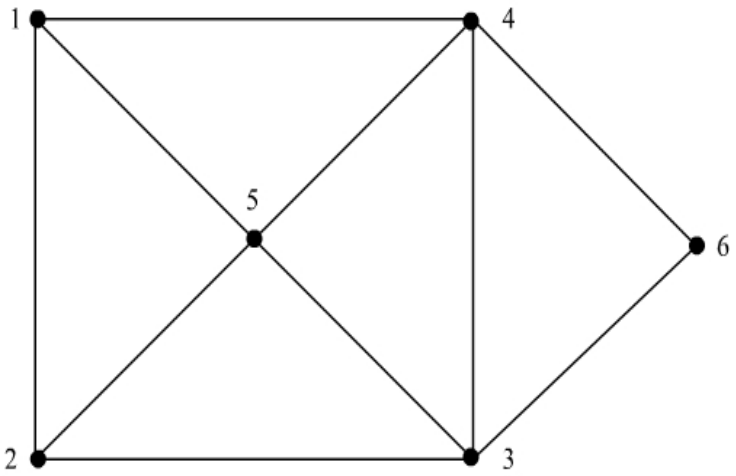


Figura 2.6: Ponto 5 a ser retirado

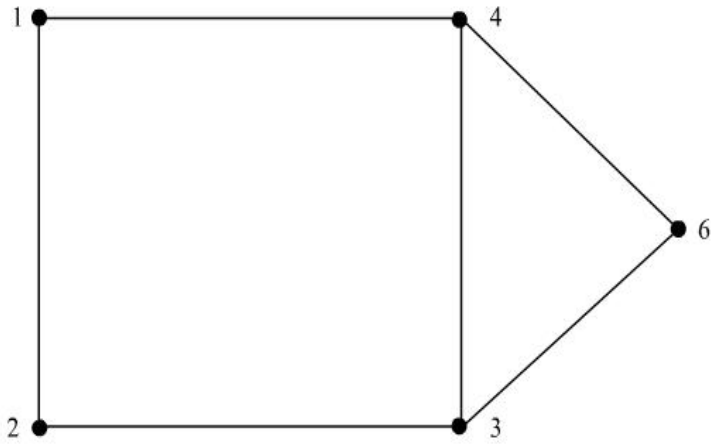


Figura 2.7: Quadrado gerado

Os pontos 1,2,3 e 4 fazem parte da lista de vizinhos do ponto 5, e todas as faces que contém o ponto 5 serão retiradas da malha, como mostrado na figura 2.8. O ponto 6 não faz parte da lista de vizinhos do ponto 5 e conseqüentemente sua face será mantida.

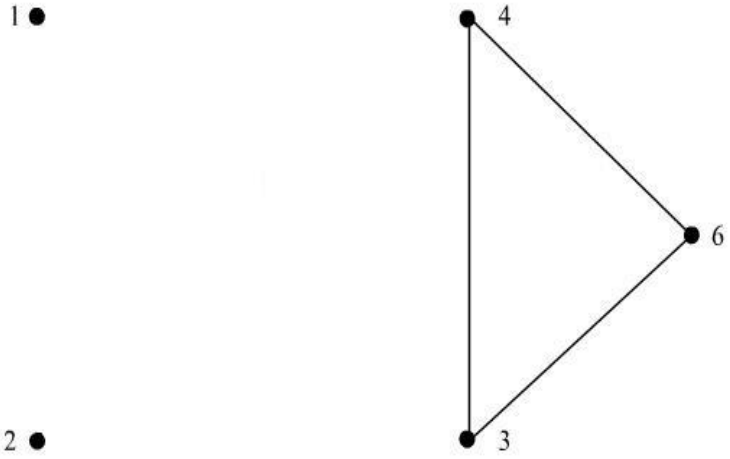


Figura 2.8: Faces retiradas

Com a lista de vizinhos, gera-se então uma nova triangulação, como mostrado na figura 2.9.

2.3.2 Triângulo

A remoção de um triângulo da malha acontece de forma similar a retirada de um ponto. Quando se retira um triângulo, cada um dos 3 pontos referentes ao triângulo é retirado e a

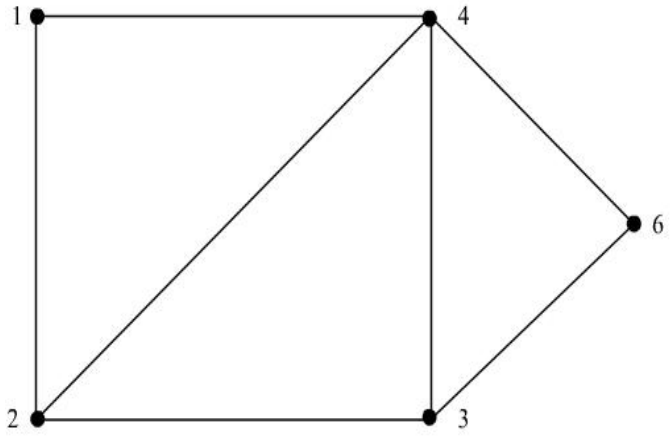


Figura 2.9: Malha retriangulada

malha retriangulada. Isto garante que não haja buracos não preenchidos na malha e que a retriangulação não gere o mesmo triângulo já retirado.

A escolha da ordem de retirada dos pontos do triângulo é aleatória. Para a retirada de cada ponto, utilizamos o algoritmo de retirada de pontos da malha descrito na seção 2.3.1.

Exemplo de aplicação

Na figura 2.10, o triângulo preenchido será retirado da malha, para isso, cada um dos 3 pontos será retirado um de cada vez, demonstraremos isso passo a passo. O primeiro ponto a ser retirado está em vermelho na imagem 2.10, a escolha do primeiro ponto a ser retirado não segue nenhum critério.

A figura 2.11 mostra o resultado da retirada do primeiro ponto e o ponto vermelho indica qual o próximo que será retirado.

Na figura 2.12 temos o resultado da retirada do segundo ponto e em vermelho qual será o terceiro e ultimo ponto a ser retirado da malha.

A 2.13 mostra o resultado final da retirado do triângulo da malha, pode-se ver que a retirada de um triângulo pode modificar de forma significativa a malha.

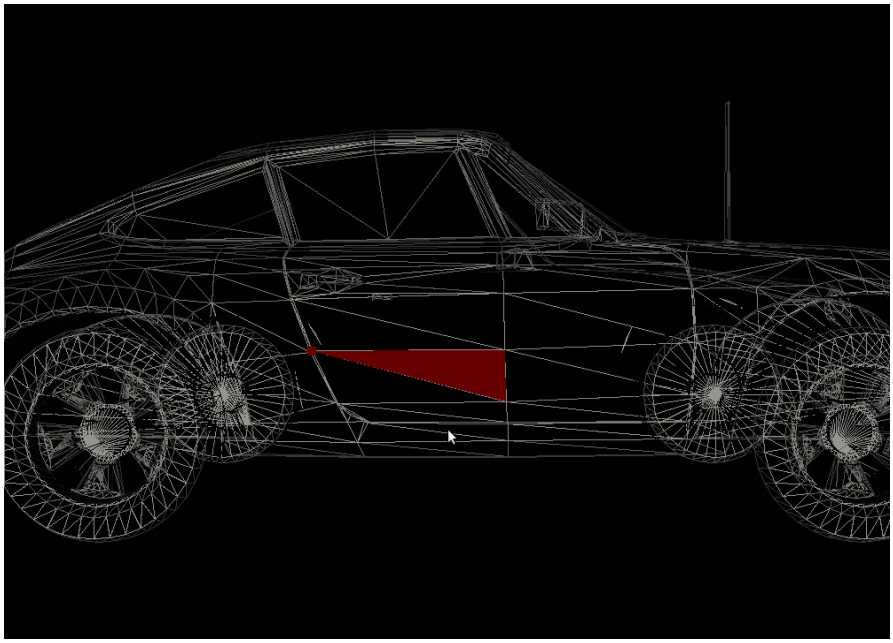


Figura 2.10: Triângulo a ser retirado

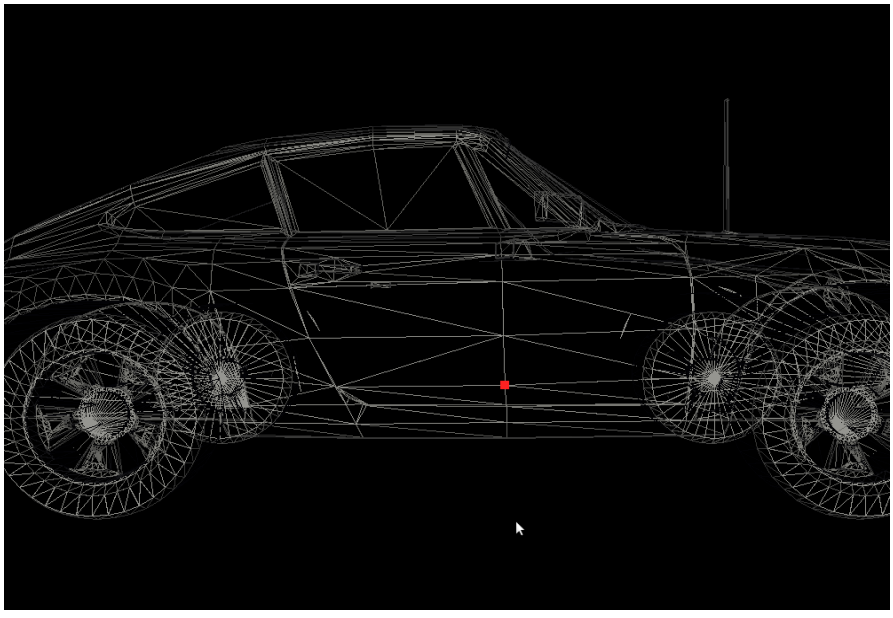


Figura 2.11: Segundo ponto do triângulo

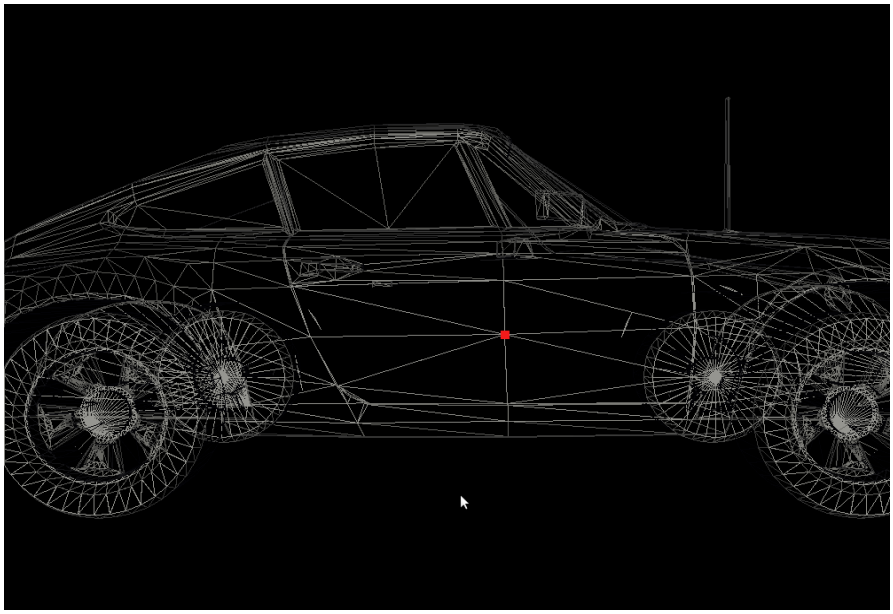


Figura 2.12: Terceiro ponto do triângulo

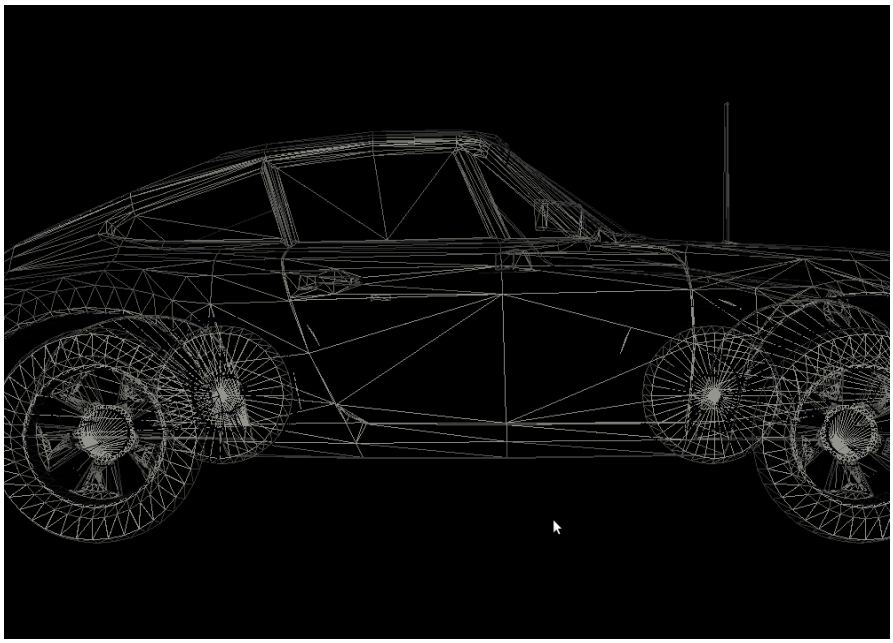


Figura 2.13: Resultado final da retirada

3 Bibliotecas utilizadas

3.1 Ply

PLY é a sigla para Polygon File Format, ou Arquivo de Formato de Polígonos, também conhecida como Stanford Triangle Format, ou Formato de Triângulo de Stanford. Tanto o formato quanto a biblioteca PLY foram originalmente desenvolvidos em The Leland Stanford Junior University e licenciados para uso livre pela própria universidade.

Segundo o idealizador do formato e da biblioteca, Greg Turk, o objetivo "...é criar um único formato de arquivo que é tanto flexível o suficiente para antecipar futuras necessidades como é simples o suficiente para não afastar potenciais usuários."[THE. . . ,]. O formato PLY foi utilizado neste trabalho para armazenar as malhas triangulares para que fossem lidas utilizando a biblioteca ply.

Os arquivos no formato PLY são compostos por duas partes essenciais: o cabeçalho e a descrição do objeto. O cabeçalho é definido sempre no início do arquivo, começando com a linha "ply". Em seguida, nele são definidos o formato do arquivo sendo utilizado (ASCII ou binário) , seguidos de uma lista com as propriedades do objeto. Utiliza-se então a linha "end_header"para indicar o fim do cabeçalho. Segue-se então a descrição do objeto, que é uma lista com as propriedades do mesmo, seguindo a sequência definida no cabeçalho. O formato PLY também aceita comentários, utilizando a linha "comment". Prosseguimos com um exemplo de um PLY simples, para facilitar o entendimento do formato:

ply

comment a linha seguinte define o formato do arquivo ply

format ascii 1.0

comment as próximas linhas definem quais as propriedades do objeto

comment nesse caso, vértices com 3 coordenadas e faces com uma lista de vértices

element vertices 8

property float x

property float y

property float z

element face 12

property list uint int vertices_indices

comment a linha seguinte indica o fim do cabecalho e o inicio da descricao do objeto

end_header

comment as próximas 8 linhas definem os 8 vértices declarados no cabeçalho

1 0 0

0 1 0

0 0 1

1 1 0

1 0 1

0 1 1

1 1 1

0 0 0

comment as próximas 12 linhas definem as 12 faces declaradas no cabeçalho

3 0 7 1

3 1 3 0

3 1 3 5

3 5 3 6

3 0 7 2

3 4 0 2

3 7 1 2

3 1 5 2

3 2 4 5

3 5 4 6

3 0 4 6

3 3 0 6

3.2 Tinlib

O escopo deste trabalho é a manipulação e visualização de malhas e não sua geração, por este motivo utilizamos uma biblioteca pronta e não implementamos um algoritmo apróprio.

A biblioteca Tinlib [ATKESON, 2005] foi usada neste trabalho para geração de uma malha triangular a partir de certos pontos passados para a sua função de geração de malha.

A Tinlib foi utilizada na remoção de pontos, remoção de triângulos e na adição de ponto a face.

3.3 OpenGL

OpenGL trata-se de um ambiente completo e de alta performance para desenvolvimento de aplicações gráficas tanto em duas quanto em três dimensões, buscando manter o foco em questões de portabilidade e adotando o sistema de licenciamento SGI [SGI. . . ,], deixando livre de exigências de licenciamento os desenvolvedores de software que utilizem openGL em seus programas [OPENGL. . . ,].

Estes são fatores importantes que nos levaram a utilizar OpenGL para renderizar as malhar triangulares em nosso trabalho.

3.4 GTK-Glade

Para criarmos as interfaces deste trabalho utilizamos o software Glade.

Glade [GLADE,] é uma ferramenta de desenvolvimento de interface ao usuário, distribuído pela licença GNU GPL, que garante o desenvolvimento de janelas GTK+ [GTK+...,] para o ambiente GNOME [GNOME...,].

As interfaces criadas a partir do Glade são salvas em arquivos XML, e podem ser carregadas usando a biblioteca libglade [LIBGLADE,].

O Glade é mantido pelos desenvolvedores Glade e por voluntários da comunidade GNOME.

4 Manual

4.1 Compilação, Execução e Encerramento

Neste trabalho, fizemos uso da ferramenta GNU Make. Desta forma, a compilação do programa fica simplificada à execução do comando 'make' na pasta raiz dos códigos fontes.

Uma vez compilados os códigos-fontes, é gerado o arquivo binário executável de nome 'mt'. Este programa deve ser executado passando-se como primeiro e único parâmetro qualquer arquivo contendo uma malha triangular no formato PLY, a explicação sobre este formato é detalhada na seção 3.1.

Ao iniciar, serão apresentadas ao usuário duas janelas, descritas nos próximos tópicos. Para encerrar o programa, o usuário deve optar por fechar a janela de renderição ou pressionar a tecla de escape, também na janela de renderização.

4.2 Janela de Renderização da Malha

A primeira das janelas exibidas, intitulada "Malhas Triangulares", exibe a renderização em openGL do arquivo recebido como parâmetro na linha de comando. Também são marcados na renderização qual vértice e qual triângulo está selecionado na janela de manipulação de malha. Nesta janela estão habilitadas diversas interações do usuário através de teclado. Seguem as teclas e os respectivos efeitos proporcionados.

- Efeitos de rotação
  - Tecla 'j': rotaciona a imagem à esquerda no eixo horizontal

  - Tecla ’l’: rotaciona a imagem à direita no eixo horizontal
  - Tecla ’k’: rotaciona a imagem abaixo no eixo vertical
  - Tecla ’i’: rotaciona a imagem acima no eixo vertical

- Efeitos de zoom
  - Tecla ’+’: aproxima a imagem de forma rápida
  - Tecla ’=’: aproxima a imagem de forma mais lenta
  - Tecla ’-’: afasta a imagem de forma rápida
  - Tecla ’_’: afasta a imagem de forma mais lenta

- Efeitos de preenchimento
  - Tecla ’w’: ativa ou desativa o preenchimento dos triângulos com a cor cinza. Por padrão, optamos por iniciar o programa com o preenchimento desativado.

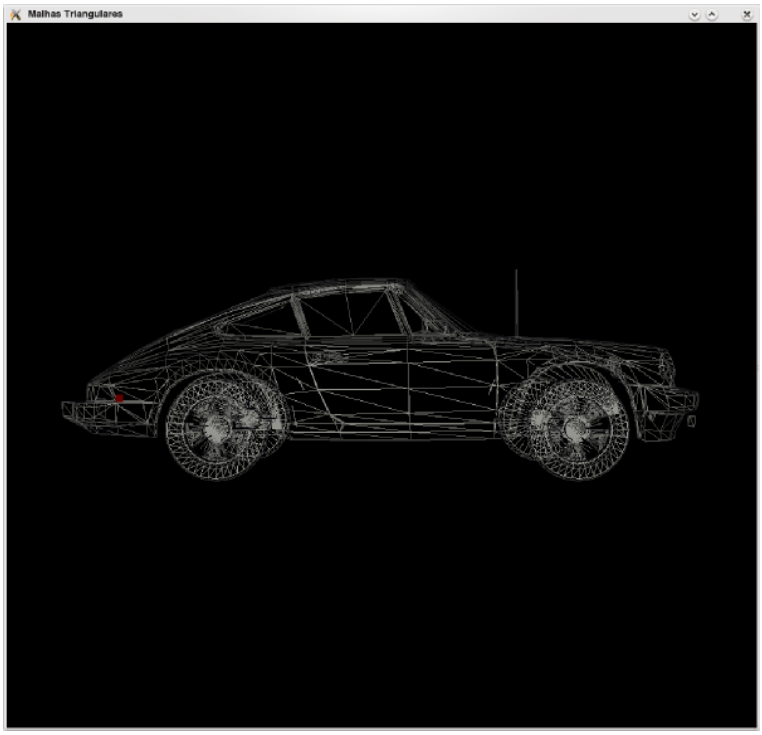


Figura 4.1: *Janela de Renderização da Malha*

4.3 Janela de Manipulação da Malha

A segunda janela exibida recebe o título de ”Operadores em Malhas Triangulares”. Foi desenvolvida utilizando as bibliotecas de janelas GTK+ [GTK+...,], desenhadas utilizando Glade [GLADE,]. Permite que o usuário interaja com a malha carregada do arquivo fornecido, de forma que as alterações se reflitam na janela de renderização.

A biblioteca Glade é descrita na seção 3.4.

As operações se dão em três áreas distintas nesta janela:

1. Lista de vértices
    - Seleção de vértices
    - Edição de vértices
    - Remoção de vértices da malha
2. Lista de triângulos
    - Seleção de um triângulo
    - Edição de um triângulo
    - Remoção dos triângulos da malha
    - Adição de um novo vértice em um triângulo selecionado
3. Simplificação de malha
    - permite realizar sucessivas operações de redução da malha, optando por reduções de 5% ou 20%

A operação de edição de triângulos e pontos é acessada com um duplo clique no valor que se quer modificar.

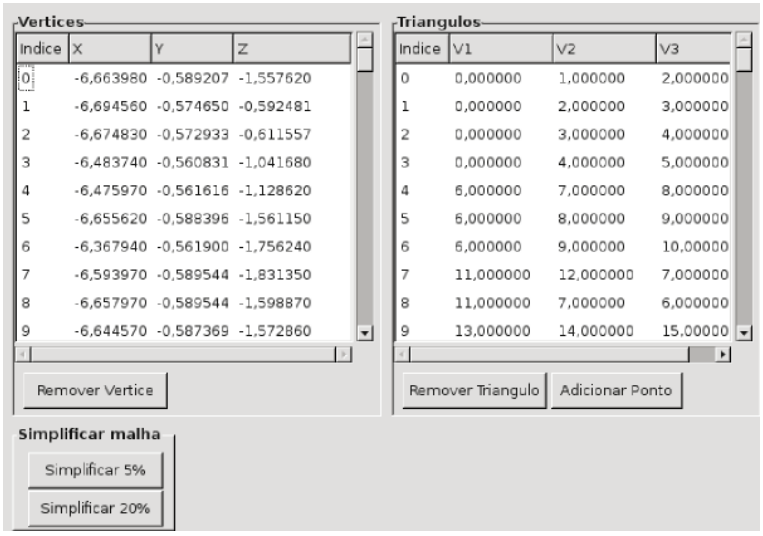


Figura 4.2: *Janela de Manipulação da Malha*

Conclusão

Malhas triangulares são estruturas de simples manuseio e de grande poder representativo, mas o material sobre o assunto é muito escasso, dificultando sua utilização e desenvolvimento. Com este trabalho, acreditamos que estamos contribuindo para a proliferação deste assunto, facilitando sua utilização por outras pessoas e servindo como um começo para alavancar novas soluções.

Neste trabalho foram desenvolvidos diversos operadores sobre as malhas triangulares, as malhas são obtidas através de arquivos no formato PLY, tendo como principal vantagem sua flexibilidade. As malhas então são exibidas em uma janela OpenGL dentro da ferramenta e suas possíveis manipulações são mostradas em uma janela GTK.

O operador de simplificação implementado reduz a complexidade da malha diminuindo o número de triângulos existentes. Implementamos também a adição e remoção de pontos e ainda a remoção de triângulos da malha.

Como uma possível forma de continuar este trabalho, uma sugestão seria pesquisar formas mais eficientes de armazenamento, otimizar a manipulação dos dados e desenvolver não apenas a manipulação das malhas, mas também suas diversas formas geração, inclusive o aumento do detalhamento da malha.

Referências Bibliográficas

ALGORRI, M.-E.; SCHMITT, F. Mesh simplification. *Computer Graphics Forum*, v. 15, n. 3, p. 77–86, ago. 1996.

ATKESON, J. C. *Free Triangulated Irregular Network library*. 2005. Website. `http://jca3.freeshell.org/tinlib/tinlib.html`.

CIGNONI, P.; MONTANI, C.; SCOPGNO, R. A comparison of mesh simplification algorithms. *Computers and Graphics*, v. 22, p. 37–54, 1998.

DEHAEMER, M. J.; ZYDA, M. J. Simplification of objects rendered by polygonal approximations. *Computers and Graphics*, v. 15, p. 175–184, 1991.

GLADE. Website. `http://glade.gnome.org/`.

GNOME: The Free Software Desktop Project. Website. `http://www.gnome.org/`.

GTK+ Project. Website. `http://www.gtk.org/`.

GUÉZIEC, A. *Surface Simplification Inside a Tolerance Volume*. [S.l.], Mar. 1996. IBM Research Report RC 20440.

HINKER, P.; HANSEN, C. Geometric optimization. In: *In Proc. Visualization ˘201993*. [S.l.: s.n.], 1993. p. 189–195.

LIBGLADE. Website. `http://www.jamesh.id.au/software/libglade/`.

LOW, K.-L.; TAN, T.-S. Model simplification using vertex-clustering. In: *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*. New York, NY, USA: ACM, 1997. p. 75–ff. ISBN 0-89791-884-3.

OPENGL Licensing. Website. `http://www.opengl.org/about/overview/#11`.

RONFARD, R.; ROSSIGNAC, J.; ROSSIGNAC, J. Full-range approximation of triangulated polyhedra. In: ROSSIGNAC, J.; SILLON, F. (Ed.). *Proceeding of Eurographics, Computer Graphics Forum*. Blackwell, 1996. v. 15(3), p. C67–C76. Disponível em: <http://perception.inrialpes.fr/Publications/1996/RRR96>.

SGI Licensing. Website. `http://www.sgi.com/products/software/opengl/license.html`.

THE PLY Polygon File Format. Website. `http://www.cs.virginia.edu/~gfx/Courses/2001/Advanced.spring.01/plylib/Ply.txt`.

TURK, G. Re-tiling polygonal surfaces. In: *Computer Graphics*. [S.l.: s.n.], 1992. p. 55–64.