4
Implementação do Gerador de arquivos VRML - VRMLGer

Neste capítulo são apresentados o desenvolvimento do Gerador de arquivos VRML - VRMLGer, a linguagem de programa utilizada, a estrutura da entrada de dados e dados resultantes da simulação, informações geométricas, de configuração do Gerador, de escolhas de geração e o que é gerado em cada escolha.

É apresentada também a linguagem de programação em scripts do software de FEA com a qual foram criados os macros para geração dos arquivos texto, que servem de entrada para o VRMLGer.

4.1.
A linguagem de programação utilizada

A linguagem de programação escolhida para o desenvolvimento do VRMLGer foi a linguagem C++ (Object Management Group, 2003) e particularmente utilizou-se o ambiente de programação do software Visual C++ 6.0 (MSDN, 2001). O desenvolvimento deste gerador não está restrito à utilização da linguagem C++. Poderia ter sido utilizada qualquer outra linguagem de programação que gerencie eficientemente a manipulação de arquivos texto e tenha a facilidade de manipular estruturas de dados dinâmicas.

Na sua programação foram utilizados conceitos de arrays dinâmicos para otimizar a utilização da memória do computador onde funcionará o VRMLGer e também para ser adaptável a qualquer quantidade de dados numéricos, como dados de entrada ou saída.

Foram utilizadas características de orientação a objetos, como programação com Classes para o armazenamento dos dados e na manipulação destas estruturas de dados utilizaram-se seus respectivos Métodos de cada Classe.

No gerenciamento dos arquivos textos de entrada utilizaram-se bibliotecas de programação que facilitaram a leitura dos arquivos de entrada e a gravação nos arquivos de saída, com extensão wrl.

4.2.
A estrutura dos dados de entrada

Os dados de entrada são arquivos no formato texto e com extensão txt contendo as seguintes informações:

ALLNODES.txt:

Arquivo que contém as informações das coordenadas de pontos no espaço, estes pontos em FEA são denominados Nós.

Na figura 9 está mais apresentada a estrutura do arquivo. Cada linha contém os dados de um ponto ou Nó. Do lado direito do índice do Nó, separados por vírgulas, estão os valores das coordenadas x, y e z. No caso de se tratar de uma estrutura em 2D, bastaria zerar os valores de um dos três eixos.

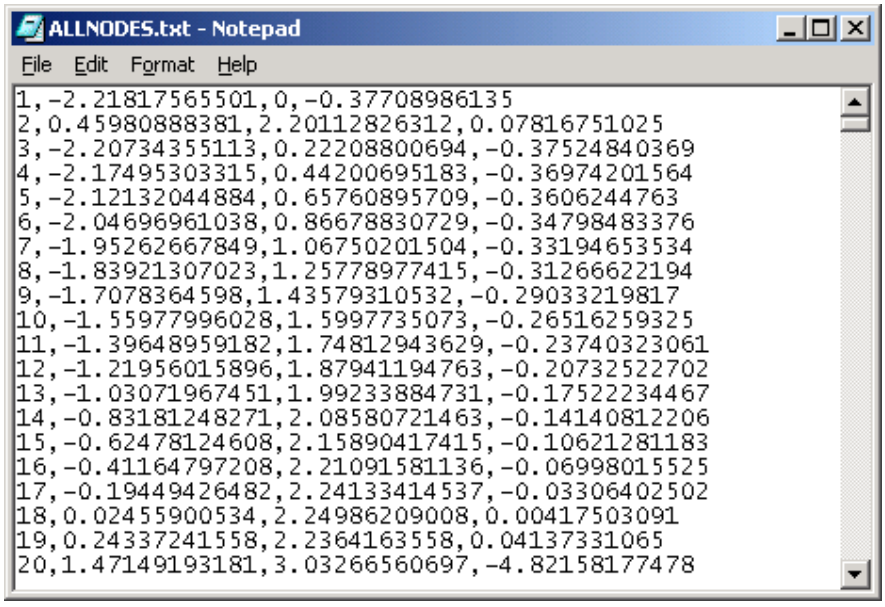

Figura 9 - Modelo do arquivo de entrada ALLNODES.txt

ALLELEM.txt:

Arquivo que contém as informações dos Nós que fazem parte de cada Elemento. Os Elementos podem ser planos quadrangulares, triangulares ou a união de 2 pontos que é chamada de uma linha. Usualmente têm-se até 3 identificadores numéricos como atributos de cada Elemento, a saber: tipo de material, tipo de Elemento e espessura do Elemento.

Na figura 14 está apresentada a estrutura do arquivo. Cada linha contém os dados de um Elemento. Do lado direito do índice do Elemento, separados por vírgulas, estão os valores dos 3 identificadores e logo depois os valores dos 4 Nós que fazem parte do Elemento.

Segundo o exemplo da figura 10 os Elementos 1 até 5 teriam um tipo de material indicado com o número 1, enquanto que os Elementos 6 até 10 teriam um outro tipo de material indicado com o número 2.

Na terceira coluna encontra-se o identificador do tipo do Elemento. No exemplo da figura 14 os Elementos de 1 a 3 são do tipo 4.

Na quarta coluna está identificador de espessura dos Elementos. No caso todos os elementos do exemplo da figura 14 têm a espessura indicada com o número 2.

As colunas 5, 6, 7 e 8 contêm os índices dos Nós que fazem parte da estrutura do elemento.

No exemplo os elementos 1, 2, 3 e 4 têm uma estrutura quadrangular, pois os quatro índices de Nós aos quais faz referência são diferentes; já os elementos 5, 6 e 7 têm os dois últimos valores dos seu respectivos Nós repetidos, indicando assim que é um elemento com uma estrutura triangular. Finalmente temos o exemplo dos elementos 8, 9 e 10, aonde os dois zeros no final indicam que estes elementos são a união só 2 Nós.

Deve-se observar que:

1. Os índices dos Nós de cada Elemento seguem o sentido anti-horário ao olhar-se para os Elementos de frente.

2. Os caracteres não numéricos são descartados no momento da leitura do arquivo.

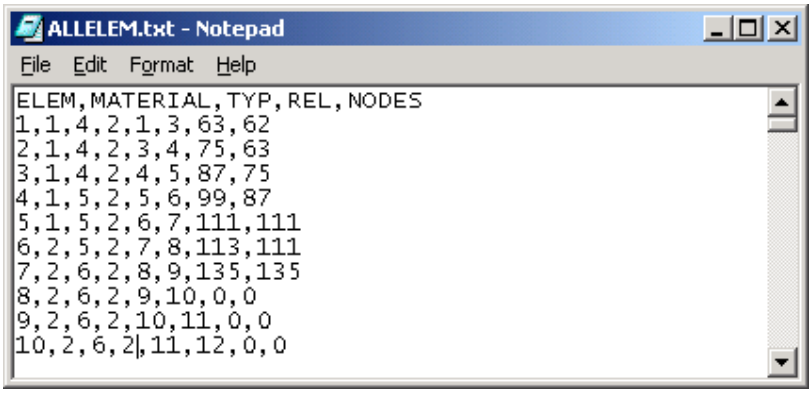

Figura 10 - Modelo do arquivo de entrada ALLELEM.txt

KPLIST.txt:

Arquivo que contém as informações dos KeyPoints, que, no software de FEA usado, são os pontos mais importantes que o usuário definiu na estrutura no momento da modelagem da malha.

Na figura 11 mostra-se um exemplo de 12 KeyPoints e logo depois dos índices de cada ponto, separados por virgulas, estão os valores das coordenadas x, y e z destes pontos.

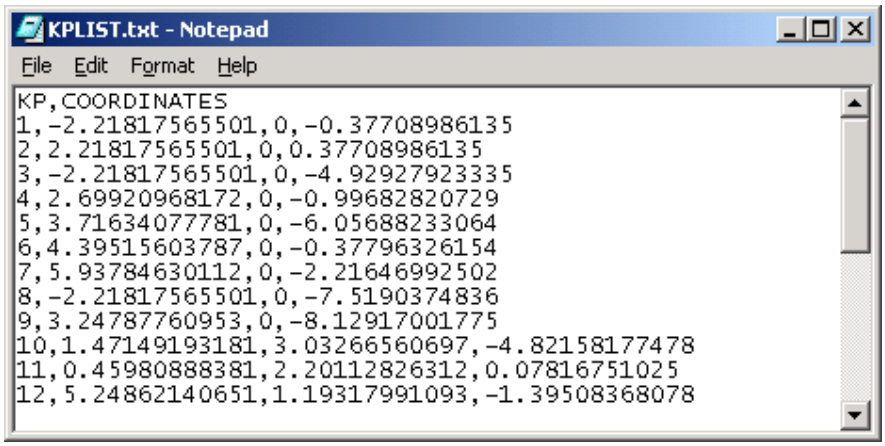

Figura 11 - Modelo do arquivo de entrada KPLIST.txt

LINES.txt:

Arquivo que contém as informações das Linhas, que são linhas no espaço, que o usuário definiu na estrutura no momento da modelagem da malha. Estas Linhas são compostas por uma série de Nós.

Este arquivo texto contém na primeira linha dois valores separados por uma vírgula. O primeiro valor é o índice da Linha e o segundo é o número de Nós que compõem a Linha. Nas linhas seguintes estão os índices dos Nós que fazem parte desta Linha. Entre uma Linha e outra há comentários que são ignorados na leitura do arquivo.

Os dois primeiros índices representam os Nós do início e final da Linha. Os demais índices representam os nós intermediários da Linha.

Na figura 12 mostra-se um exemplo de um arquivo LINES.txt. A primeira Linha tem 6 Nós, o Nó 1 é seu Nó inicial e o Nó 33 é seu Nó final. Os Nós 60, 61, 62 e 63 são os Nós intermediários desta Linha. Após o texto define-se a composição da Linha número 2 que tem 4 Nós e o restante da informação é análogo ao explicado para Linha 1.

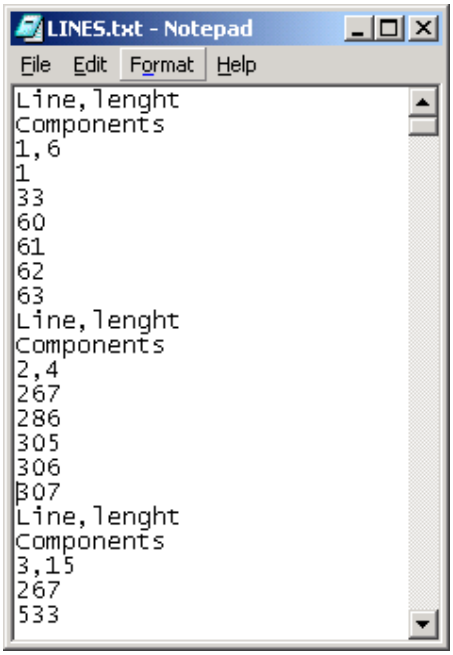

Figura 12 - Modelo do arquivo de entrada LINES.txt

AREAS.txt:

Arquivo que contém as informações das Áreas, que são na realidade superfícies no espaço, que o usuário definiu na estrutura no momento da modelagem da malha. Estas superfícies são compostas por uma série de pequenas superfícies, que são os Elementos. Cada Elemento pertencente a somente uma Área em particular.

Este arquivo contém, na primeira linha, dois valores separados por uma vírgula. O primeiro valor é o índice da Área e o segundo valor é o número de Elementos que compõem esta Área. Nas linhas seguintes estão os índices dos Elementos que fazem parte desta Área.

Uma nova Área será definida ao aparecer dois novos valores separados por uma vírgula.

Na figura 13 mostra-se um exemplo de um arquivo AREAS.txt. A primeira Área tem 234 Elementos, cujos índices estão nas linhas subsequentes.

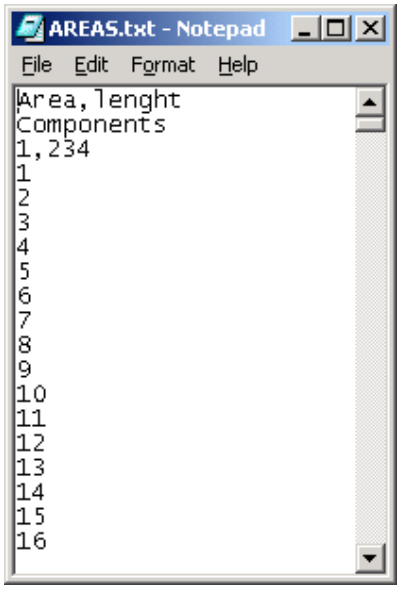

Figura 13 - Modelo do arquivo de entrada AREAS.txt

ALLDOF.txt:

Arquivo que contém informações sobre os graus de liberdade (DOF) de cada Nó. Armazenam os deslocamentos x,y,z das posições originais de cada Nó da estrutura depois de ter sido realizado o processamento de FEA.

Este arquivo texto tem uma estrutura parecida com a estrutura do arquivo ALLNODES.txt. Cada linha contém o índice do Nó seguido das informações de quanto se deslocou um Nó no espaço x, y, z desde sua posição original.

Na figura 14 mostra-se um exemplo de um arquivo ALLDOF.txt. Cada linha contém informação do DOF de cada Nó correspondente do arquivo ALLNODES.txt. Por exemplo, pegando a primeira linha desde arquivo texto, o Nó 1 tem os seguintes deslocamentos: de 0.00318452085 em X, de 0.0 em Y e de –.211202656x10^{-5} em Z.

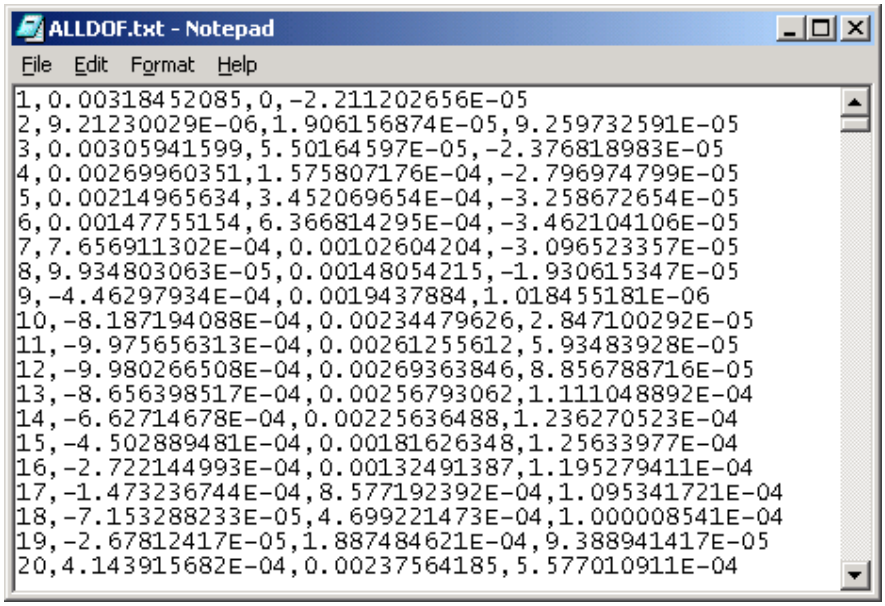

Figura 14 - Modelo do arquivo de entrada ALLDOF.txt

RES.txt:

Arquivo que contém os resultados do processamento, uma malha modelada pelo usuário, em uma ferramenta computacional de FEA.

Este arquivo tem uma estrutura parecida com a estrutura do arquivo ALLNODES.txt. Cada linha contém três valores de resultados e um primeiro de indice, separados por vírgulas, que são correspondentes a um Nó.

Na figura 15 mostra-se um exemplo de um arquivo RES.txt, com dados correspondentes aos valores de S1, S2 e S3 para cada Nó. S1, S2 e S3 representam os valores das tensões principais. No exemplo, o Nó 1 tém os valores 37749778.9, 12571649.4 e 0.08288429776 correspondentes a S1, S2 e S3, respectivamente.

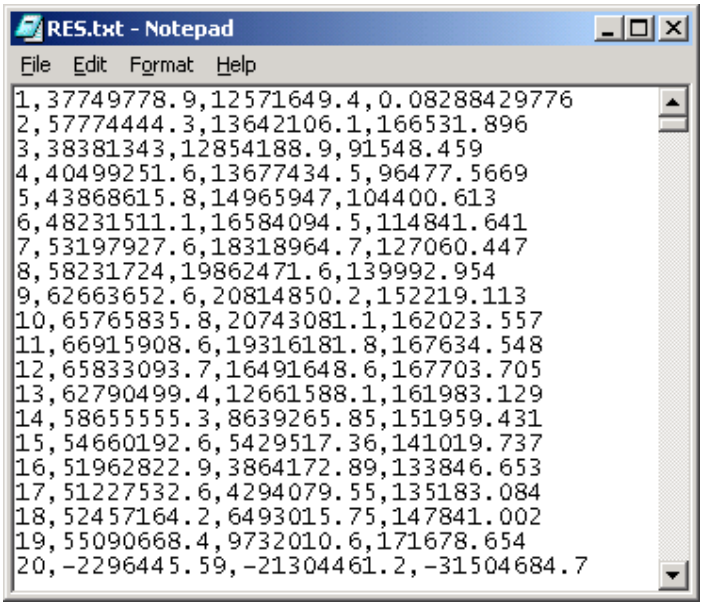

Figura 15 - Modelo do arquivo de entrada RES.txt

4.3.
A configuração do VRMLGer

A configuração do Gerador é feita por meio de arquivos texto São fornecidas informações sobre a paleta de cores usada na visualização dos resultados, as funções que devem ser ativadas no gerador de VRML, a transparência com que devem ser mostrados os objetos, a saturação das cores do material do objeto etc.

O arquivo paletaCores.txt:

Arquivo que contém 128 linhas e em cada linha de texto tem 3 valores, separados por vírgulas, correspondentes aos valores de Vermelho(R), Verde(G) e Azul(B). Estes valores estão entre 0 e 1 e a combinação destes três valores geram uma cor especifica, sendo que 0 indica ausência da componente e 1 é o valor máximo da componente.

Por default, em FEA, a paleta de cores para mostrar dados entre 128 cores diferentes começa com a cor Azul (R=0, G=0, B=1), passa gradativamente até a cor Verde (R=0, G=1, B=0) e passando gradativamente termina na cor Vermelha (R=1, G=0, B=0). Na figura 16 mostra-se a escala de cores default que começa do o Azul o vai até o Vermelho.

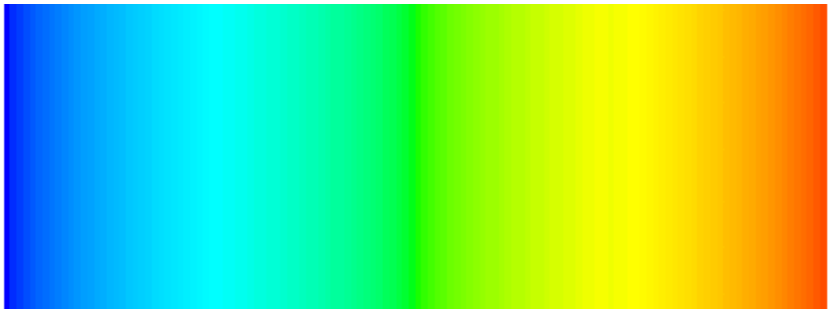

Figura 16 – Paleta de Cores de 128 valores

Na figura 17 mostra-se o arquivo dos dados da paleta de cores de 128 valores, na primeira linha se descreve o valor da cor Azul absoluta. Nas linhas seguintes vai se aumentando o valor da componente Verde até atingir os valores de R=0, G=1 e B=1, logo depois vai se diminuindo os valores de Azul até chegar ao Verde absoluto (R=0, G=1, B=0). Faz-se a mesma operação de passo gradativo entre as cores Verde e Vermelho.

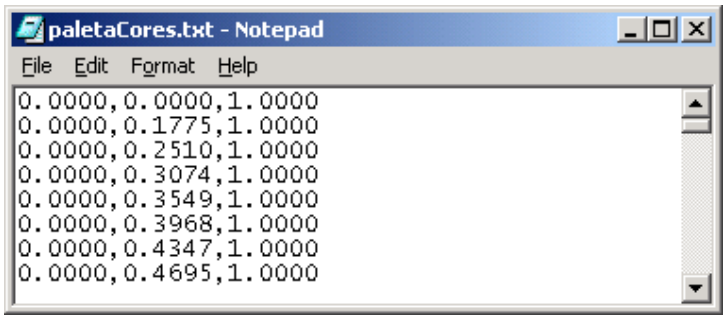

Figura 17 - Modelo do arquivo de configuração da paleta de cores.

O arquivo transparency.txt:

Arquivo que contém um único valor numérico que varia entre 0 e 1, onde 1 é a total transparência dos objetos a serem colocados dentro dos arquivos VRML e 0 não mostra efeito nenhum de transparência nos objetos.

O arquivo diffuseColor.txt:

Arquivo que contém um valor numérico que pode ser 0 ou 1, o valor de 1 ativa a ação e o valor de 0 a desativa. A ação consiste em calcular a componente difusa (Watt, 2000) da iluminação gerando um sombreamento mais realista.

Na figura 18 mostra-se um exemplo de um objeto gerado com esta ação ativado mediante o valor 1 no arquivo diffuseColor.txt. Na figura 19 mostra-se um

exemplo do mesmo objeto da figura 18 só que desta vez com a ação desativada mediante o valor 0 no arquivo diffuseColor.txt.

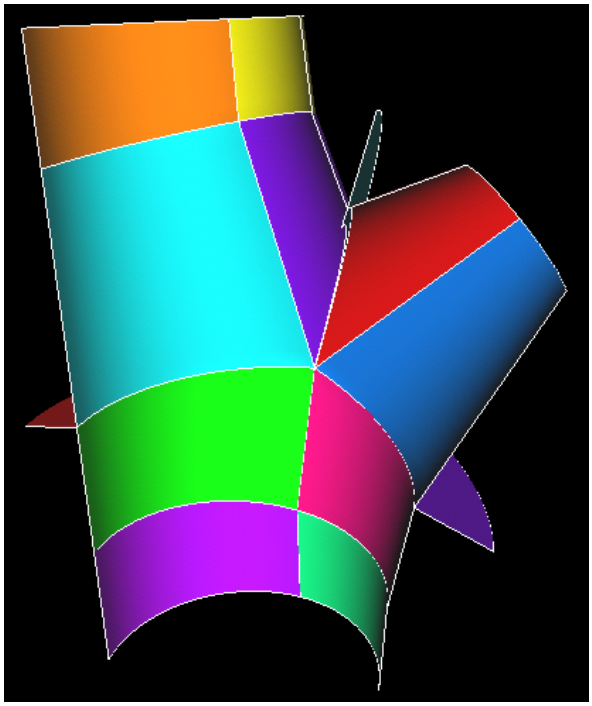

Figura 18 - Objeto gerado com o valor 1 (ativado) no arquivo diffuseColor.txt.

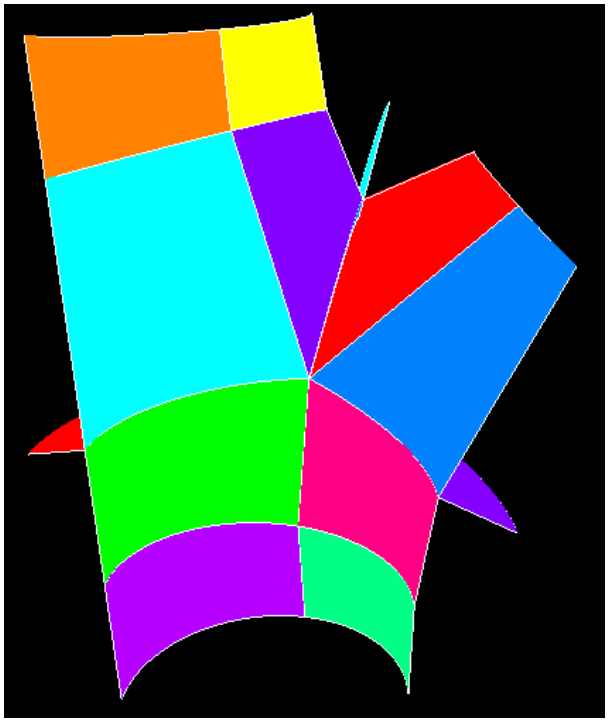

Figura 19 – Objeto gerado com o valor 0 (desativado) no arquivo diffuseColor.txt.

O arquivo creaseAngle.txt:

Arquivo que contém um valor numérico entre 0 e 6.283185, correspondendo a valores de ângulos em radianos.

Suaviza as bordas das faces e superfícies adjacentes toda vez que o ângulo entre elas for igual ou menor que o especificado no arquivo creaseAngle.txt.

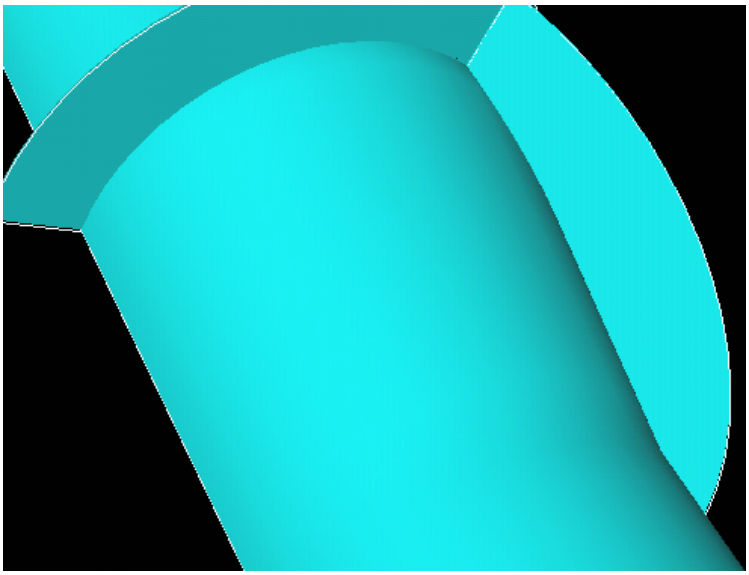

Figura 20 - Objeto gerado com valor de 0.5 (com suavização) no arquivo creaseAngle.txt.

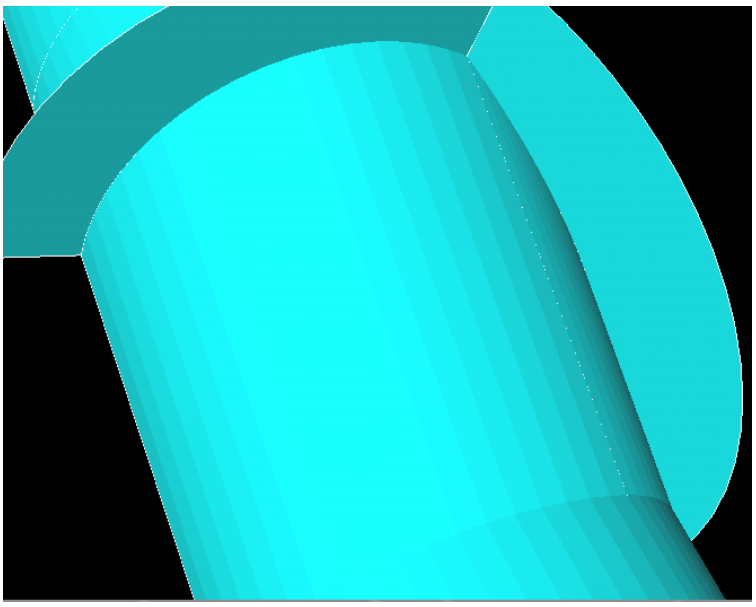

Figura 21 - Objeto gerado com valor de 0.0 (sem suavização) no arquivo creaseAngle.txt.

Na figura 20 mostra-se um exemplo de um objeto gerado com um creaseAngle de 0.5. Na figura 21 se mostra um exemplo do mesmo objeto da figura 20 só que desta vez gerado com um creaseAngle de 0, onde o objeto é visualizado facetado.

Uma observação importante é que o valor colocado no arquivo creaseangle fica sem efeito caso o valor do arquivo difuseColor.txt seja 0.

O arquivo funções.txt:

Arquivo que contém uma lista de valores relacionados com determinadas funções de geração a serem realizadas dentro do VRMLGer. Os valores possíveis são: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 140, 141, 142, 143, 150, 151 e 152. Cada um destes números ativará uma função de geração que resulta em um arquivo VRML com extensão wrl.

Na figura 22 mostra-se um arquivo exemplo de entrada, ativando todas as funções do gerador.

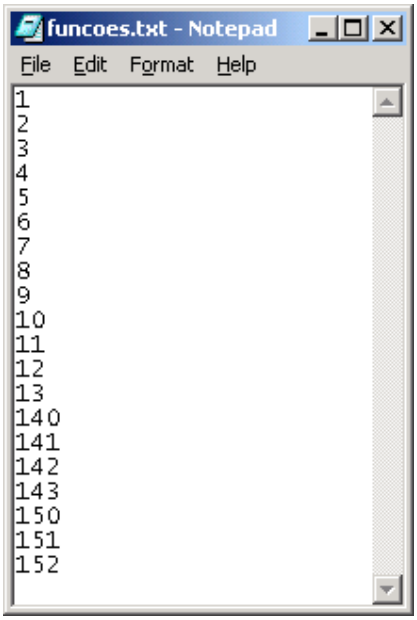

Figura 22 – Modelo do arquivo de entrada que ativa as funções do Gerador.

4.4. Dados de saída em VRML das funções de geração

Cada função dentro do VRMLGer gera um VRML diferente. A seguir serão apresentadas cada uma das funções de geração dentro do VRMLGer

1 : Gera em um arquivo VRML a malha de Elementos, fornecida no arquivo ALLELEM.txt, traçada somente por linhas brancas.

2 : Gera em um arquivo VRML a malha de Elementos, com seus planos preenchidos e sem desenhar as linhas, o que faz parecer todo o objeto como se fosse um bloco inteiro.

3 : Gera em um arquivo VRML a malha de Elementos, com linhas traçadas e seus planos preenchidos.

4 : Gera em um arquivo VRML os Elementos, representados por uma coloração diferente, relativos ao tipo de material, fornecidos no arquivo ALLELEM.txt.

5 : Gera em um arquivo VRML os Elementos, representados por uma coloração diferente, relativos ao tipo de elemento, fornecidos no arquivo ALLELEM.txt.

6 : Gera em um arquivo VRML os Elementos, representados por uma coloração diferente, relativos a espessura do elemento, fornecidos no arquivo ALLELEM.txt.

7 : Gera em um arquivo VRML as linhas fornecidas no arquivo LINES.txt. Estas linhas são traçadas em diferentes cores.

8 : Gera em um arquivo VRML as áreas fornecidas no arquivo AREAS.txt. Estas áreas são preenchidas em diferentes cores.

9 : Gera em um arquivo VRML os pontos no espaço fornecidos no arquivo KPLIST.txt.

10 : Gera em um arquivo VRML os pontos no espaço fornecidos no arquivo ALLNODES.txt.

11 : Gera em um arquivo VRML os deslocamentos (DOFs) dos Nós fornecidos no arquivo ALLDOF.txt.

12 : Gera em um arquivo VRML a malha de elementos considerando os deslocamentos dos Nós, isto é a estrutura deformada.

13 : Gera em um arquivo VRML as duas malhas da estrutura, a malha inicial não deformada junto com a malha final deformada. Cada uma delas com um preenchimento de cor diferente. Este arquivo VRML tem interação do usuário com os objetos mediante *picks* nas estruturas. A programação da interação foi desenvolvida em linguagem JavaScript. A interação consiste em virar transparente, invisível ou visível cada uma das malhas independentemente uma da outra.

140 : Gera em um arquivo VRML a malha de elementos considerando os deslocamentos dos Nós, como no caso da função 12, mas desta vez os elementos têm uma cor diferente em cada Nó. Esta cor depende do valor armazenado na segunda coluna do arquivo ALLDOF.txt. É feita uma correlação entre estes valores e as cores da paleta de cores, onde o menor valor de DOFs corresponde à cor Azul e o maior valor corresponde à cor Vermelha. Os valores da segunda coluna do arquivo ALLDOF.txt são os valores de UX. UX é a distancia que se deslocou só no eixo X cada Nó desde seu ponto original antes da análise.

141 : Gera em um arquivo VRML a malha de elementos considerando os deslocamentos dos Nós, como no caso da função 12, mas desta vez os elementos têm uma cor diferente em cada Nó. Esta cor depende do valor armazenado na terceira coluna do arquivo ALLDOF.txt. É feita uma correlação entre estes valores e as cores da paleta de cores, onde o menor valor de DOFs corresponde à cor Azul e o maior valor corresponde à cor Vermelha. Os valores da terceira coluna do arquivo ALLDOF.txt são os valores de UY. UY é a distancia que se deslocou só no eixo Y cada Nó desde seu ponto original antes da análise.

142 : Gera em um arquivo VRML a malha de elementos considerando os deslocamentos dos Nós, como no caso da função 12, mas desta vez os elementos têm uma cor diferente em cada Nó. Esta cor depende do valor armazenado na quarta coluna do arquivo ALLDOF.txt. Os valores da quarta coluna do arquivo ALLDOF.txt são os valores de UZ. UZ é a distancia que se deslocou só no eixo Z cada Nó desde seu ponto original antes da análise.

143 : Gera em um arquivo VRML a malha de elementos considerando os deslocamentos dos Nós, como no caso da função 12, mas desta vez os elementos têm uma cor diferente em cada Nó. Esta cor depende dos três valores das colunas dois, três e quatro, armazenados no arquivo ALLDOF.txt, UX, UY e UZ, respectivamente. O módulo do vetor deslocamento é chamado em FEA de USUM.

150 : Gera em um arquivo VRML a malha de elementos considerando os deslocamentos dos Nós, como no caso da função 12, mas desta vez os elementos têm uma cor diferente em cada Nó. Esta cor depende

do valor armazenado na segunda coluna do arquivo RES.txt. Como nas funções acima é feita uma correlação entre estes valores e a paleta de cores. Para nosso estudo de caso corresponde com os valores de S1, que é correspondente à primeira tensão principal.

151 : Gera em um arquivo VRML a malha de elementos considerando os deslocamentos dos Nós, como no caso da função 12, mas desta vez os elementos têm uma cor diferente em cada Nó. Esta cor depende do valor armazenado na terceira coluna do arquivo RES.txt. Como nas funções acima é feita uma correlação entre estes valores e a paleta de cores. Para nosso estudo de caso corresponde com os valores de S2, que é correspondente à segunda tensão principal.

152 : Gera em um arquivo VRML a malha de elementos considerando os deslocamentos dos Nós, como no caso da função 12, mas desta vez os elementos têm uma cor diferente em cada Nó. Esta cor depende do valor armazenado na quarta coluna do arquivo RES.txt. Como nas funções acima é feita uma correlação entre estes valores e a paleta de cores. Para nosso estudo de caso corresponde com os valores de Tensão Equivalente – Von Mises para o Modelo 1 e 2 e corresponde com os valores da terceira tensão principal para os Modelos 3 e 4.

4.5. Macros na linguagem APDL do ANSYS

Para extrair os dados necessários, em arquivos texto, que servem como entrada no VRMLGer se utilizou uma linguagem de programação criada pela mesma ANSYS, chamada de APDL (ANSYS, 2000). Os dados necessários se encontram nos arquivos dbs (em formato proprietário da ANSYS) onde só se pode ter acesso mediante rotinas na linguagem APDL. Na realidade APDL é uma linguagem de Scripting que serve para automatizar procedimentos na construção de estruturas ou modelos (ANSYS, 2000).

Foram desenvolvidas algumas macros que servem para obter os arquivos texto, entrada do VRMLGer. O código pode ser visto no Apêndice B.