APLICAÇÃO DE FERRAMENTAS DE VISUALIZAÇÃO CIENTÍFICA A RESULTADOS DE MODELOS METEOROLÓGICOS COM ADAPTAÇÃO ESPACIAL

Margarete Oliveira Domingues (margaret@cptec.inpe.br)
Eni Alvim de Oliveira (eni@cptec.inpe.br)

Laboratório Associado de Meteorologia e Oceanografia (LMO)
Centro de Previsão de Tempo e Clima (CPTEC)
Instituto Nacional de Pesquisas Espaciais (INPE)
Caixa Postal 515
12201-970 - São José dos Campos, SP, Brasil

PALAVRAS CHAVE: Adaptação Espacial; Visualização Científica; Open DX; Vis5D; "Free Software"
GRUPO TEMÁTICO: 11 – Serviços e Produtos Meteorológicos

RESUMO

Em Meteorologia são comuns as situações em que o escoamento apresenta estruturas com variações bruscas em certas regiões. Para uma representação adequada dessas estruturas são necessárias grades muito refinadas. Em muitos casos, essas estruturas possuem uma localização espacial reduzida em comparação com o domínio do escoamento. Por essa razão, há um interesse por métodos numéricos adaptáveis no espaço que usem grades menos refinadas nas regiões em que o escoamento seja mais suave. Trabalhar apenas nas malhas refinadas nas regiões de interesse significa economia de custo computacional, que pode chegar a valores altamente significativos. Além disso, com a variação temporal da posição dessas estruturas, é importante que seja possível redefinir as grades, dinâmica e automaticamente, durante as simulações numéricas. As estruturas de dados associadas a esse tipo de método são muito complexas, e demandam técnicas específicas para serem visualizadas. Existem alguns sistemas de Visualização Científica dirigidos a grandes volumes de dados multidimensionais, sendo que alguns deles possuem código aberto, são gratuitos e multiplataformas. Um desses sistemas é o OpenDX e outro é o Vis5D, especialmente voltado para aplicações meteorológicas. Este trabalho apresenta uma breve discussão do problema científico da aplicação de métodos adaptáveis no espaço, bem como uma análise comparativa do uso desses sistemas nesse tipo de visualização. São discutidas questões como tempo investido na aprendizagem do uso das ferramentas, documentação, facilidade de instalação, apoio técnico e procedimentos de visualização.

INTRODUÇÃO

As situações em que o escoamento apresenta estruturas com variações bruscas necessitam grades muito refinadas para uma representação adequada. Em muitos casos, essas estruturas possuem uma localização espacial reduzida, em comparação com o domínio do escoamento. Por essa razão, há um interesse por métodos numéricos adaptáveis no espaço, que usem grades mais refinadas apenas nessas regiões, e menos refinadas nas regiões em que o escoamento seja mais suave. Além disso, com a variação temporal da posição dessas estruturas, é importante que seja possível redefinir as grades, dinâmica e automaticamente, durante as simulações numéricas. Nesse tipo de técnica, a estrutura da malha adaptável resultante é totalmente dependente da função analisada, podendo apresentar uma composição bem heterogênea: esparsa em regiões de suavidade e densa nas regiões de pouca suavidade. Esse fato aumenta a complexidade da estrutura de dados, principalmente em dimensões superiores. Por isso, para o caso bidimensional, pode ser preferível malhas que sejam compostas por sub-malhas localmente regulares. Isso facilita as operações de acesso e reduz a complexidade das estruturas para armazenar esses dados, o que, em geral, reduz o custo computacional total. Uma discussão mais detalhada desse tema é obtida em Domingues (2001).

A motivação para este trabalho é apresentar a saída de modelos adaptáveis espacialmente utilizando dois sistemas de representação 3D. Inicialmente a visualização foi desenvolvida com o sistema OpenDX. Apesar de ser poderoso e gratuito, esse sistema é pouco divulgado na comunidade meteorológica brasileira, e sua distribuição, até o momento, não segue as normas GNU/GPL

(http://www.gnu.org). Como alternativa para esse sistema, utiliza-se o sistema Vis5D, mais conhecido nessa comunidade, e com licença GNU/GPL.

DESCRIÇÃO DOS DADOS A SEREM VISUALIZADOS

Os dados a serem visualizados estão apresentados em um domínio composto de sub-malhas regulares, que podem apresentar diferentes resoluções espaciais. Tais dados pertencem a uma malha irregular com uma estrutura de dados bem definida, na qual as sub-matrizes têm sempre o mesmo número de elementos e estão dispostas nesse domínio fixo. Na prática, a malha regular refinada que representaria todo o domínio não precisa ser construída explicitamente para ser visualizada. Isso reduz o custo computacional do cálculo da interpolação dos pontos ausentes, nas sub-malhas menos refinadas, mas que compõem a malha principal que representa o domínio, e portanto elimina também o custo de armazenamento de todas essas informações. Portanto, o desafio dessa visualização é construir uma superfície de visualização 3D no domínio usando apenas as informações das sub-malhas. É importante destacar que essa construção de sub-malhas é elaborada de forma a reproduzir adequadamente a informação.

VISUALIZAÇÃO NO OPENDX

OpenDX é um programa gratuito semelhante ao Data Explorer da IBM (maiores informações podem ser obtidas na página http://www.opendx.org). Esse programa tem como finalidades principais a visualização de informações formatadas em 2D e 3D, de forma interativa ou automática - por meio de "scripts" ou de inclusão no código fonte de funções em linguagem C ou Fortran - e a criação de filmes de visualização temporal de uma base de dados em formato "mpeg". Uma das grandes vantagens desse recurso é sem dúvida a facilidade de programação em ambiente visual, conhecido pela sua sigla em inglês VPE. A Figura 1 apresenta o VPE utilizado na visualização de uma saída do modelo adaptável com três opções de visualização: a superfície 3D gerada, a malha com os pontos utilizados (Figuras 2 e 3).

Visualizar dados em malhas irregulares é um recurso disponível no OpenDX. Por outro lado, dada a estrutura de sub-malhas a serem representadas, verificou-se ser mais simples e menos dispendioso computacionalmente fazer visualização por meio do conjunto de sub-matrizes. O maior desafio encontrado nesta parte do trabalho é preparar a entrada das sub-matrizes no padrão ".dx" para visualizar os dados no domínio sem precisar compor a malha refinada correspondente, e descobrir um meio de visualizar esses dados 2D de forma 3D. No Apêndice encontra-se a função utilizada para gerar esses dados e um exemplo dessa saída. Basicamente cada sub-malha é representada por um trio de objetos: *gridposition, gridconnection, array type.* O primeiro deles é o responsável por determinar quantos pontos existem na sub-malha, quais são suas coordenadas iniciais (ponto mais ao sul e mais a oeste) e seus espaçamentos na direção norte-sul e na leste-oeste. O segundo define como serão conectados os pontos da malha; neste caso se utiliza uma conexão de malha regular. Para simular a conexão entre as sub-malhas, usa-se o seguinte truque: cada sub-malha compartilha suas fronteiras direita e superior (internas ao domínio) com suas respectivas sub-malhas vizinhas. O terceiro objeto define como a matriz de dados está organizada; neste caso ela está obedecendo uma variação do tipo linha/coluna, i.e., $(x_0,y_0)(x_0,y_1)(x_0,y_2)$ $(x_0,y_3)....(x_1,y_0)(x_1,y_1)(x_1,y_2)(x_1,y_3)...$. O nome do arquivo de dados que será lido fica definido neste objeto. Após definir esses três objetos, cria-se o campo de dados a ser lido, com a instrução *field*. Nessa instrução informa-se quais trios de objetos estão sendo incluídos nesse campo. No final do processamento, agrupa-se os campos na instrução *group*, em que o número a seguir da instrução *value* determina o campo de dados definido. A função utilizada para a representação 3D é a *rubber sheet*, em que os dados são tratados como se fossem dados de topografia, sobrepondo um mapa de cores a esse relevo.

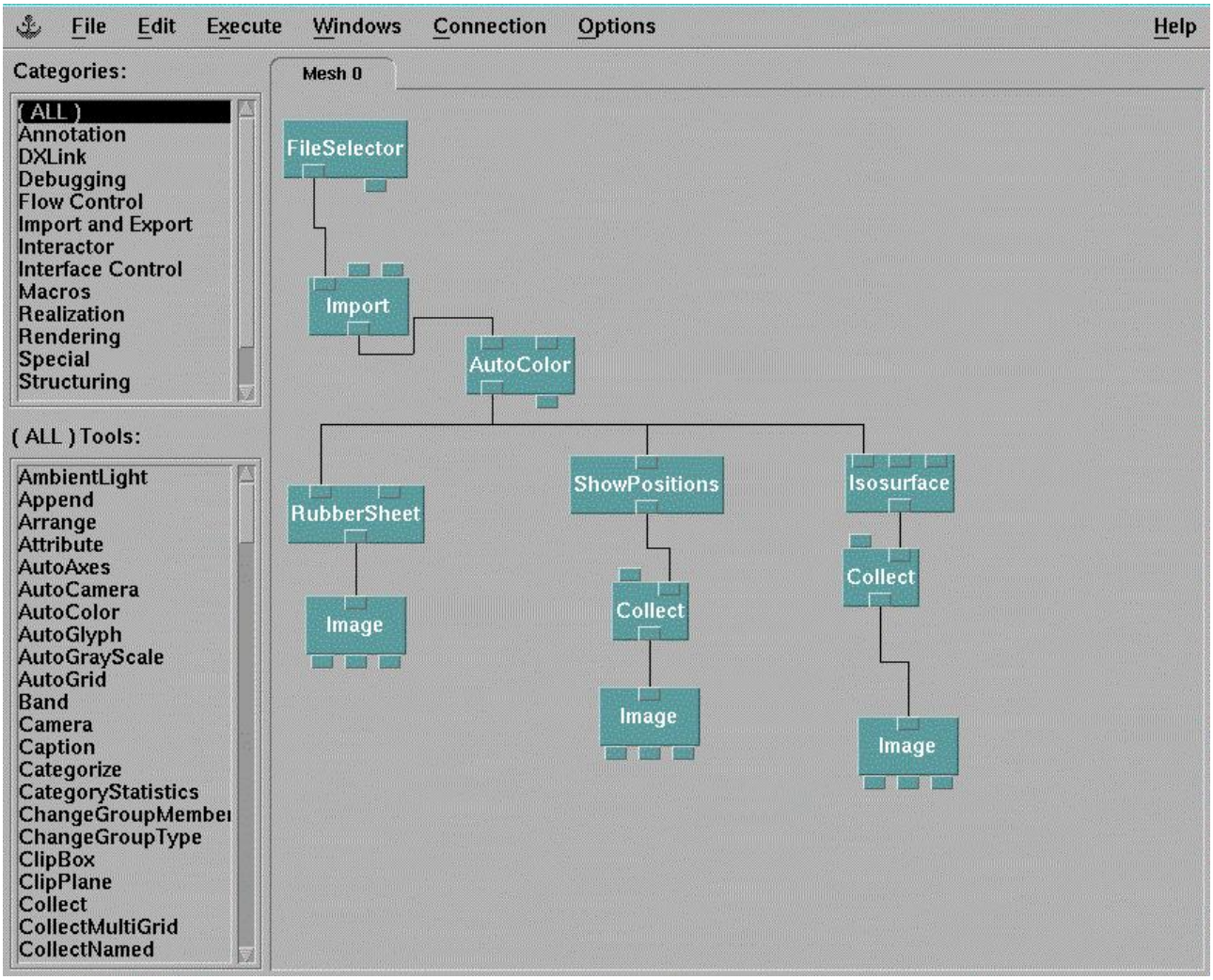

Fig. 1 – Exemplo de um programa no VPE.

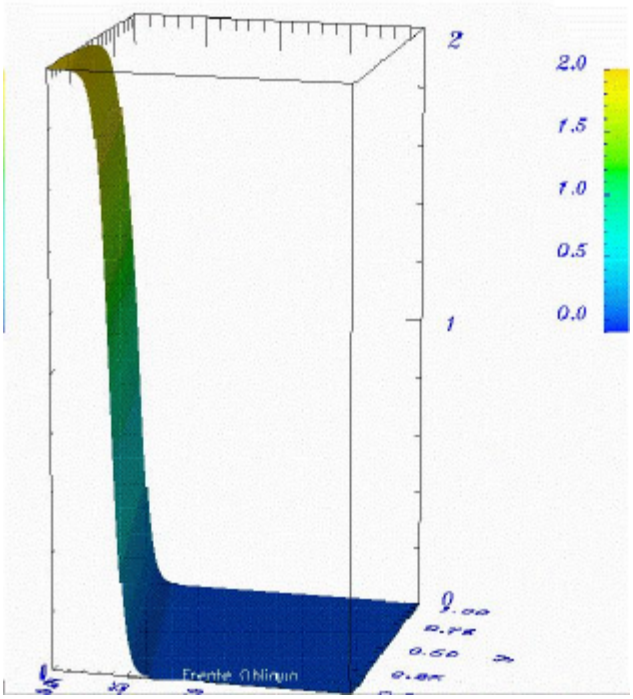

Fig. 2 – Exemplo de visualização 3D.

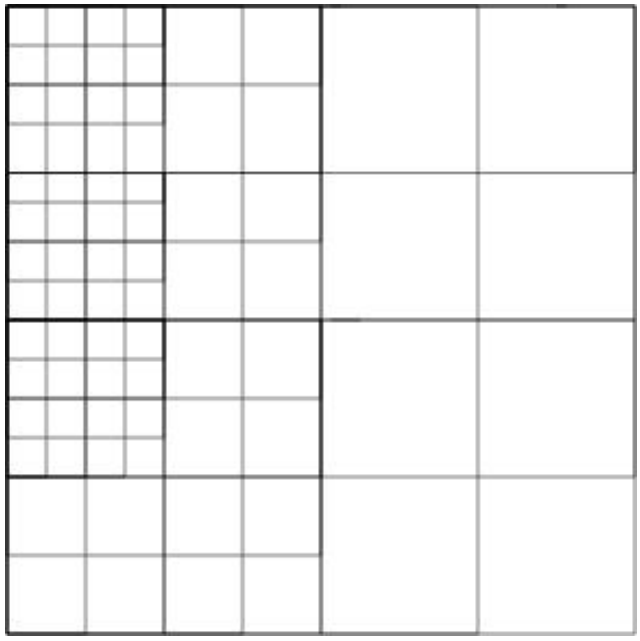

Fig. 3 – Sub-matrizes utilizadas na visualização da Figura 2.

VISUALIZAÇÃO NO VIS5D

A **descrição do sistema Vis5D** bem como as instruções para a instalação e o uso podem ser encontradas na página http://www.ssec.wisc.edu/~billh/vis5d.html. A versão utilizada neste trabalho é a 4.3.

O **sistema Vis5D** é destinado especialmente à visualização de grandes arquivos de dados 5D, tais como os produzidos por modelos numéricos de previsão do tempo e similares. Os dados são localizados em uma malha 3D, e considera-se a quarta dimensão como o tempo, e a quinta como a enumeração de diversas variáveis físicas. Os recursos do sistema permitem o traçado de isosuperfícies, isolinhas, planos de corte, volumes, etc., bem como a animação das imagens geradas, porém não permite a visualização de dados em grades irregulares.

A **preparação dos dados** a serem visualizados consiste em uma conversão de formato e uma alteração na sua organização. Os dados de entrada são os contidos nos arquivos gerados para o sistema OpenDX, e a conversão é feita com base em um programa exemplo que acompanha o sistema Vis5D. A partir deles, gera-se um arquivo no formato e na organização estabelecidos pelo Vis5D. No caso em questão, os dados estão ordenados por linha para o tratamento pelo OpenDX, e devem ser ordenados por coluna no Vis5D; além disso, a origem de cada sub-malha é definida como o ponto superior esquerdo (posição Norte-Oeste), enquanto que no OpenDX, neste caso, é o ponto inferior esquerdo (posição Sul-Oeste).

Considera-se uma única variável, ou seja, tem-se apenas uma superfície de dados. A projeção escolhida é a linear retangular, em unidades genéricas, fornecendo-se as coordenadas da origem, e os espaçamentos entre as linhas e entre as colunas. O sistema de coordenadas verticais é o de níveis igualmente espaçados, em unidades genéricas, sendo que, no caso deste trabalho em particular, tem-se um único nível, ou seja, a superfície é definida por apenas uma variável.

A representação visual dos dados apresenta, pois, apenas uma superfície na qual os valores da variável estão associados a cores. Para a obtenção de uma superfície no espaço tridimensional, opta-se por um subterfúgio, criando-se um arquivo de topografia no qual o dado em cada ponto da malha é considerado como uma cota. Isto possibilita a representação 3D com cores associadas aos valores da variável, e permite o traçado de isolinhas e de perfis de corte. A criação de um arquivo próprio de topografia é possível através de um programa exemplo que está incluído no Vis5D. Um exemplo de visualização utilizando o Vis5D pode ser visto nas Figuras 4 e 5.

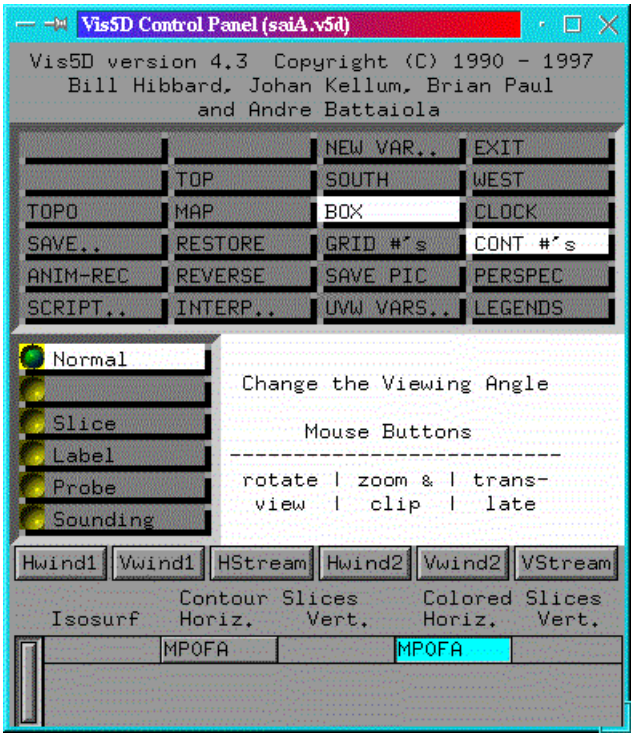

Fig. 4 – Painel do sistema Vis5D.

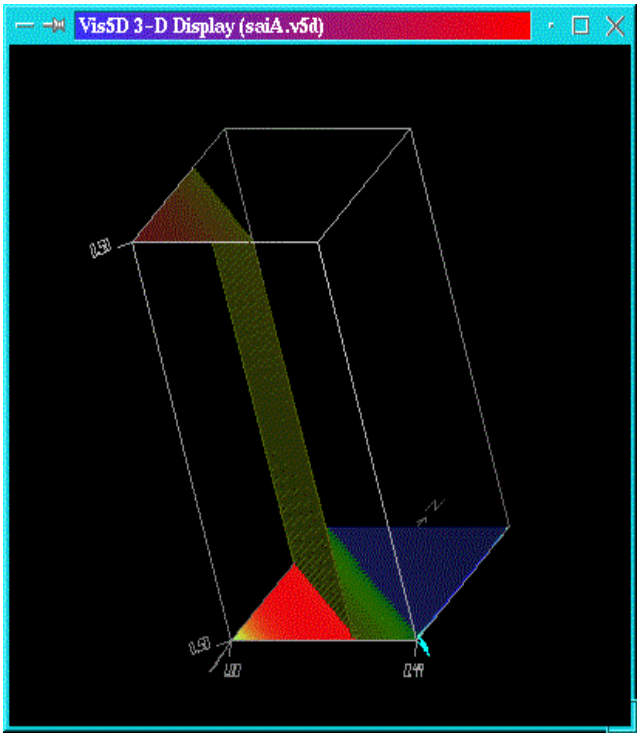

Fig. 5 – Exemplo de visualização com o Vis5D.

CONCLUSÕES

Os principais comentários sobre o uso desses sistemas são:

a) a instalação com sucesso de ambos em ambiente GNU/Linux depende de experiência do instalador, por exemplo, ver no Apêndice 1 a descrição da instalação do OpenDX sob o sistema operativo GNU/Linux; por outro lado, no caso de estações Sun as instruções de instalação são suficientes;

b) o tempo investido na aprendizagem inicial do OpenDX é menor, pois este sistema dispõe do VPE, além da forma usual de comandos;

c) a documentação de ambos é suficiente para o desenvolvimento de visualizações para as quais os sistemas foram originalmente projetados; o apoio técnico é proveniente de listas de discussão e tem se mostrado satisfatório;

d) no OpenDX a visualização é obtida pela composição das sub-malhas, formando a representação 3D, com a utilização da função *rubber sheet*. A solução proposta com o uso do Vis5D consiste na geração de imagens correspondentes às diferentes resoluções, compondo-as posteriormente para efeitos de visualização, de modo a poder apresentar os resultados obtidos, diferenciando as sub-malhas menos refinadas das sub-malhas mais refinadas. Esta proposta implica em um gasto computacional adicional em relação à solução adotada utilizando o OpenDX. Portanto, este é um tópico ainda em desenvolvimento, e está sendo iniciado o uso do sistema VisAD (http://www.ssec.wisc.edu/~billh/visad.html) para a visualização dessas malhas regulares por blocos.

BIBLIOGRAFIA BÁSICA PARA A IMPLEMENTAÇÃO

Domingues, M. O. Métodos Adaptativos (a ser publicado como parte integrante de Tese de Doutorado) DMA/IMECC, **UNICAMP**. 2001.

IBM **Visualization Data Explorer. QuickStart Guide**. IBM. SC34-3262-02, sl. sd. (http://www.opendx.org)

IBM **Visualization Data Explorer. User's Reference**. IBM. SC38-0486-03, sl. sd. (http://www.opendx.org)

IBM **Visualization Data Explorer. Programmer's Reference**. IBM. SC38-0486-03, sl. sd. (http://www.opendx.org)

Universidade de Wisconsin. **Vis5D README** sl. sd. (http://www.ssec.wisc.edu/~billh/vis5d.html).

Universidade de Wisconsin. **VisAD.** sl. sd. (http://www.ssec.wisc.edu/~billh/visad.html).

Silva, E.C. **Introdução ao IBM Data Explorer**. CENAPAD-SP, SP, agosto, 1997.

Treinish, L. **Data Explorer Tutorial for Earth, Space and Environmental Sciences**. IBM Thomas J. Watson Research Center, sl., sd. (http://www.tc.cornell.edu)

AGRADECIMENTOS

As autoras agradecem às Dras. Sônia M. Gomes (IMECC/UNICAMP) e Lilliam M. Alvarez Diaz (Instituto de Cibernética, Matemática e Física de Havana, Cuba) e à Fapesp (processo 94/ 2016-9) pelo apoio à realização deste trabalho. Agradecimentos também são devidos ao Dr. Odim Mendes Jr. e à analista Angela Y. Harada, pela instalação do OpenDX.

APÊNDICE

1) Etapas de Instalação do OpenDX

O OpenDX está disponível para várias plataformas. Na plataforma UNIX as instruções contidas nas páginas do OpenDX são suficientes. Não ocorreram problemas na instalação em uma Sun/Solaris. Por outro lado no caso do GNU/LINUX(Red Hat e/ou Conectiva) percebeu-se que cada micro, na verdade, é um universo distinto de qualquer outro, embora possa parecer terem a mesma estrutura eletrônica e os mesmos pacotes computacionais. Outrossim, em GNU/Linux, a instalação depende da experiência da pessoa que está instalando. Para muitos PCs a instalacao é simples e ocorre da seguinte maneira:

a)Instale inicialmente os arquivos:

- bzip2-0.9.5d-1cl (que provê o libbz2.so.0 necessário ao ImageMagick)
- libstdc++-2.9.0-14cl (necessário para o OpenDx funcionar)
- libstdc++-2.95.1-2.10.0-3 (que complementa as bibliotecas)

b)Instale então o OpenDx.

A seguir, para que o OpenDx funcione de qualquer lugar, é conveniente estabelecer um link simbólico para o comando de execução "dx". Para isso execute o seguinte comando: cp -l /usr/local/dx/bin/dx /usr/bin. Neste ponto, se tudo tiver transcorrido tranquilamente, é so testar o OpenDx com um dos exemplos contidos na pasta dxsamples-4.0.8. O comando IMPORT do OpenDx não encontra esse diretório diretamente e é necessário utilizar o comando FILESELECTOR para selecionar o diretório corretamente.

Caso haja placas controladoras de vídeo fora do comum, é possível que seja necessário instalar duas bibliotecas que lidam com o ambiente gráfico:

- libGL.so -- que está contida no jogo Quake
- libglide2x.so -- requerida em placas de vídeo com acelerador

Elas devem ser copiadas nos diretórios:

- /lib/libGL.so
- /usr/lib/libglide2x.so

2)Função Membro de Formação de Dados ".dx"em C^{++}:

```
void TMesh::PrintDataExplorer(int kmin,  int kmax, int lmin, int lmax, char *idName){
int var,block,mother,k,l, bmax, kk,ll,kp,lp,object;
```

```
char aux[140], auxdx[140];
double x,y,level,dx,dy,data;

kk =  (*this).kN;  ll =   (*this).lN;
kp=kmax + abs(kmin) +1;  lp=lmax + abs(lmin) +1;
object =0;
if( (bmax = (*this).fkblock) == 0) bmax=1;
  for(block=0;block<bmax;block++){
       level= Data[block].WhichLevel();
       dx=1.0f/(kk   * pow(2.0f,level) );
       dy= 1.0f/(ll  * pow(2.0f,level) );
        x=Data[block].WhereIsX()+kmin*dx;
        y=Data[block].WhereIsY()+lmin*dy;
        fstream out;
        memset(aux,0,sizeof(aux));
        memset(aux,0,sizeof(aux));

         sprintf(aux,"PF%d.%s.dat",block+1,idName);
         sprintf(auxdx,"MeshP.%s.dx",idName);
          out.open(aux,ios::out);
                for(k=kmin;k<=kmax;k++){
                      for(l=lmin;l<=lmax;l++){
                                data=Data[mother][block].ff00[k][l];
                                          if(fabs(data) < 1E-30) data=0.0f;
                                    out<<data<<" ";
                                    }
                                    out<<endl;
                                }
             out.close();
        if(block==0){
          (*this).DxFormatDataBegin(x, dx, y,dy, kp, lp,aux, auxdx,object);
                    object+=5;
           }
        else{
            if( (block==bmax-1)){
             (*this).DxFormatDataInside(x, dx, y,dy, kp,lp,aux,auxdx,object);
                           object+=3;
           (*this).DxFormatDataEnd(auxdx,object);
            }
               else{
           (*this).DxFormatDataInside(x, dx, y,dy, kp,lp, aux, auxdx,object);
                           object+=3;
             }

        }

}

//---------------------------------------------------------------------------
void TMesh::DxFormatDataBegin(double x0,double dx,double y0,double dy,int  nx,int ny,char* filedata, char* filedx,
int object)
{
fstream poutdx;
int items = nx*ny;
poutdx.open(filedx,ios::out);
            poutdx << "# Definition of regular positions and conections"<<endl;
            poutdx << "object 1 class gridpositions counts "<<nx<< " "<<ny<< " "<<endl;
            poutdx <<"origin "<<x0<< " "<<y0<<"  "<<endl;
            poutdx <<"delta  "<<dx<< " 0"<< "  "<<endl;
            poutdx <<"delta  0  "<< dy<<"  "<<endl;
            poutdx<<endl;
            poutdx<<"object 2 class gridconnections counts "<<nx<<" "<< ny <<endl;
            poutdx<<"# attribute \"element type\" string \"quads\""<<endl;
            poutdx<<"# attribute \"ref\" string \"positions\""<<endl;
            poutdx<<endl;
            poutdx<<"object 3 class array type float rank 0 items " << items<<" data file "<< filedata<<endl;
           poutdx<<"attribute \"dep\" string \"positions\" "<<endl<<endl;
```

```
        poutdx<< "object 4 class field"<<endl;
        poutdx<< "component \"positions\" value 1" <<endl;
        poutdx<< "component \"connections\" value 2" <<endl;
        poutdx<< "component \"data\" value 3" <<endl;


        poutdx.close();
        return;
}

//---------------------------------------------------------------------------
void TMesh::DxFormatDataInside(double x0,double dx,double y0,double dy,int nx,int ny,char*  filedata, char* filedx,
int object)
{
fstream poutdx;
int items = nx*ny;
poutdx.open(filedx,ios::app); //Para unir as diversas sub-malhas
poutdx <<endl<<"# Definition of regular positions and conections"<<endl;
poutdx << "object "<<object<<" class gridpositions counts "<<nx<< " "<<ny<<"  "<<endl;
poutdx <<"origin "<<x0<< " "<<y0 << "  "<<endl;
poutdx <<"delta "<<dx<< " 0  "<<endl;
poutdx <<"delta  0 "<<dy<<"  "<<endl<<endl;
poutdx<<"object "<< object+1<<" class array type float rank 0 items " << items<<" data file "<<
        filedata<<endl<<endl;
poutdx<<"attribute \"dep\" string \"positions\" "<<endl<<endl;
poutdx<< "object "<<object+2<< " class field"<<endl;
poutdx<< "component \"positions\" value "<<object <<endl;
poutdx<< "component \"connections\" value 2 " <<endl;
poutdx<< "component \"data\" value "<< object+1 <<endl<<endl;
poutdx.close();
}
//---------------------------------------------------------------------------
void TMesh::DxFormatDataEnd(char* filedx, int object)
{
fstream poutdx;
int  i,k;
poutdx.open(filedx,ios::app); //Citar os membros para visualização
poutdx<<"object 980 class group"<<endl;
poutdx<<"member 0 value  4"<<endl;
if(object>6){
         for(i=1, k=7;k<object;k+=3,i++)
             poutdx<<"member "<<i <<" value "<<k<<endl;
             }
poutdx<<"end"<<endl;
poutdx.close();
}
```

4)Exemplo do Formato de Dados ".dx":

```
# Definição da sub-malha regular 33x33 pontos, com coordenadas iniciais (x_0,y_0)=(0,0) e  dx=dy=0,015625
object 1 class gridpositions counts 33 33
origin 0 0
delta  0.015625 0
delta  0  0.015625

# Atribuindo conexões para malhas regulares
object 2 class gridconnections counts 33 33
attribute "element type" string "quads"
attribute "ref" string "positions"

#Indicando o nome do arquivo de dados, o formato e o número de elementos esperados
object 3 class array type float rank 0 items 1089 data file P0.dat
attribute "dep" string "positions"

#Definindo o campo. Os valores 1, 2 e 3 estão associados aos objetos definidos anteriormente.
```

object 4 class field
component "positions" value 1
component "connections" value 2
component "data" value 3

Iniciando um novo campo
object 5 class gridpositions counts 33 33
origin 0 0.5
delta 0.0078125 0
delta 0 0.0078125

object 6 class array type float rank 0 items 1089 data file P1.dat

#Observe que a conexão dos pontos, descrita no object 2, é mantida.
object 7 class field
component "positions" value 5
component "connections" value 2
component "data" value 6

#As demais sub-malhas são representadas da mesma maneira.
...
#Após todas as sub-malhas já estarem definidas, cria-se o grupo, em que cada value do membro se refere ao campo (field) definido anteriormente.

object 980 class group
member 0 value 4
member 1 value 7
member 2 value 10
member 3 value 13