DESENVOLVIMENTO DE INTERFACES ESPECÍFICAS PARA VISUALIZAÇÃO DE DADOS DE MODELOS DE PREVISÃO DE TEMPO

RELATÓRIO FINAL DE PROJETO DE INICIAÇÃO CIENTÍFICA
(PIBIC/CNPq/INPE)

Divani Carvalho Barbosa (UNITAU, Bolsista PIBIC/CNPq)
E-mail: divani@cptec.inpe.br

Dr. Prakki Satyamurty (INPE/LMO/CPTEC, Orientador)
E-mail: saty2@cptec.inpe.br

COLABORADORES

MSc. Eugênio Sper de Almeida (CPTEC/INPE)

Junho de 2002

ÍNDICE

DESENVOLVIMENTO DE INTERFACES ESPECÍFICAS PARA A VISUALIZAÇÃO DE DADOS DE MODELOS DE PREVISÃO DE TEMPO

Divani Carvalho Barbosa[1] (UNITAU, Bolsista PIBIC/CNPq)

Dr. Prakki Satyamurty[2] (CPTEC/LMO/INPE)

MSc. Eugênio Sper de Almeida[3] (CPTEC/INPE)

RESUMO

O Gridded Data Viewer (GDV) é um sistema de visualização de modelos meteorológicos desenvolvido pela "University Corporation for Atmospheric Research" que possibilita a visualização dos dados em três dimensões. O objetivo deste trabalho foi adaptar uma nova interface ao sistema GDV, tornando assim possível a entrada de dados gerados pelos Modelos Numéricos de Previsão de Tempo e Clima processados pelo supercomputador do CPTEC/INPE. Os Modelos Numéricos de Previsão de Tempo e Clima são programas complexos que representam o movimento e os processos físicos da atmosfera através de equações matemáticas. Para representação gráfica deste tipo de informação torna-se necessário a utilização de sistemas de visualização. Atualmente no ambiente operacional do CPTEC/INPE os meteorologistas utilizam-se dos sistemas de visualização GrADS, METVIEW e Vis5D. O sistema GDV, devido ao fato de ser desenvolvido utilizando a linguagem de programação orientada a objetos (Java), pode ser utilizado em diferentes plataformas. Atualmente o sistema possibilita visualizar dados somente no formato NetCDF (nc), formato padrão utilizado pela NASA. A partir do estudo da biblioteca de classes do VisAD, base do sistema e do código fonte do programa, foi realizada a adaptação da leitura do arquivo no formato Vis5D (v5d), um dos formatos utilizado pelo CPTEC/INPE. Portanto o intuito deste estudo foi adaptar esta nova interface chamada Gridded Data Viewer (GDV) no ambiente operacional do CPTEC/INPE para colaborar na previsão do tempo e clima gerada pelos meteorologistas.

[1] Aluna do Curso de Bacharelado em Computação Científica, UNITAU. **E-mail: divani@cptec.inpe.br**

[2] Pesquisador Titular do Centro de Previsões de Tempo e Estudos Climáticos, Laboratório de Meteorologia e Oceanografia. **E-mail: saty2@cptec.inpe.br**

[3] Tecnologista do Centro de Previsões de Tempo e Estudos Climáticos. **E-mail: eugenio@cptec.inpe.br**

1. Introdução

1.1 Modelos Numéricos de Previsão de Tempo

Atualmente a previsão de tempo está intimamente ligada a análise de resultados de modelos numéricos processados pelos supercomputadores, SX-3/12R e SX-4/8A, que possuem capacidade de processar até 16 bilhões de operações aritméticas em ponto flutuante por segundo (CPTEC-INPE, 2001). Os modelos numéricos para simulação de tempo e clima, integram informações atmosféricas, oceânicas e do solo.

Eles são programas complexos que representam o movimento e os processos físicos da atmosfera através de equações matemáticas. Estes modelos recebem como parâmetros de entrada dados observacionais, dados derivados de imagens de satélites e dados gerados por modelos de dias anteriores. Através das equações dos modelos, esses dados são processados gerando arquivos de previsão numérica de tempo. Estes arquivos são armazenados na forma de matrizes, sendo cada matriz relacionada a uma determinada variável física, a um nível atmosférico e um instante de tempo. Por exemplo, a variável "temperatura", possui 19 matrizes que correspondem aos níveis atmosféricos do modelo de superfície até a estratosfera. Atualmente no CPTEC/INPE, os modelos são executados duas vezes ao dia: 00 UTC e 12 UTC, gerando a partir destes horários arquivos com dados de previsão de tempo para diferentes intervalos (12, 24, 36, 48, 72, 96, 120, 144, 168 horas) (Satyamurty e Bitencourt, 1996).

A introdução de modelos numéricos tornou possível fazer previsão de tempo com melhor confiabilidade e maior prazo de antecedência. O modelo global do CPTEC processa dados globais com uma resolução horizontal de aproximadamente 100 km x 100 km, produzindo previsões de tempo com até 7 dias de antecedência e o modelo regional (ETA) processa dados regionais com uma resolução horizontal de 40 km x 40 km produzindo previsões de tempo com até 3 dias de antecedência.

1.2 Diagnósticos Estatísticos

Nas previsões numéricas do tempo existem erros sistemáticos, que aparecem por causa da carência de dados e parametrizações físicas inadequadas dos modelos numéricos. Os valores destes erros dependem da localização geográfica e da estação do ano. O conhecimento destes erros permite corrigir as previsões dos modelos numéricos. A visualização gráfica dos erros permite o georeferenciamento dos mesmos.

Para fazer previsões por estações meteorológicas e prever algumas variáveis que não são calculadas no modelo numérico físico (ex.: temperatura mínima e máxima), utilizam-se os conceitos Perfect Prog (PP) e Model Output Statistics (MOS) (Kim et al., 1998). Também, desenvolvem-se algoritmos específicos baseados nos processos físicos para a previsão dos fenômenos como a incidência de geada, granizo, nevoeiro, etc. Os mapas de distribuição geográfica desses fenômenos são preparados pelos sistemas de visualização.

As estatísticas de desempenho dos modelos são geralmente obtidas para previsões de variáveis simples, como Altura Geopotencial, Vento e Umidade Relativa.

Estas estatísticas que quantificam a confiabilidade das previsões são produzidas operacionalmente pelos centros de previsão numérica de tempo.

Os meteorologistas e pesquisadores precisam visualizar graficamente os campos de variáveis meteorológicos, diagnósticos, campos derivados e estatísticas de desempenho para verificar a consistência espacial e temporal das previsões.

2. Sistemas de Visualização Científica

A transformação de dados numéricos e alfanuméricos em gráficos, sempre foi um processo útil para que os cientistas desenvolvessem estudos e pesquisas meteorológicas eficientemente. A visualização consiste em interpretar os dados através de representação gráfica (ex.: mapas, tabelas, etc.). Para a visualização de dados dos modelos numéricos, informações derivadas do satélite e dados observacionais, os meteorologistas necessitam de ferramentas gráficas.

Cada software utiliza técnicas particulares desenvolvidas pelos seus fabricantes, porém todos tem o mesmo objetivo de interpretar variados tipos de dados e visualizá-los em formas gráficas. Como exemplos de softwares de visualização meteorológico, podemos citar: METVIEW, GrADS, Vis5D, SpreadSheet e GDV, sendo que os dois últimos possuem a caracteristica de manipular dados tridimensionais.

2.1 METVIEW

O sistema METVIEW foi desenvolvido conjuntamente pelo ECMWF (Centro Europeu de Previsão de Tempo) e CPTEC/INPE (Centro de Previsão de Tempo e Estudos Climáticos/Instituto Nacional de Pesquisas Espaciais). É um sistema flexível, modular e extensível, que permite aos meteorologistas acessar, manipular e visualizar dados meteorológicos, em plataforma UNIX (Workstations), nos formatos recomendados pela WMO ("Word Meterological Organization"): GRIB (GRIdded Binary) para imagens de satélite e grades provenientes de modelos numéricos e BUFR (Binary Universal Form for the Representation of meterological data) para dados observacionais (Cartaxo e Daabeck, 1998).

O sistema permite também visualização isolada ou combinada de imagens de satélites, dados numéricos de previsão de tempo/clima e dados observacionais. Possui aplicações que permitem a geração de produtos comumente utilizados pelos meteorologistas, como:

- Perfil Vertical - gráfico do perfil vertical dos campos do ar superior;
- Média - mostra a seção cruzada vertical da média dos parâmetros de ar superior;
- Seção Cruzada - gráfico da seção cruzada vertical do ar superior;
- Meteograma - mostra a evolução temporal de dados de previsão;
- Tefigrama - plota tefigramas (um diagrama termodinâmico da coluna atmosférica) de dados observacionais ou de campos sobre pontos selecionados;

• Filtragem de Observação - as variáveis dos dados observacionais podem ser plotados individualmente ou na sua totalidade, extraindo-se um subconjunto de dados (tempo, espaço e parâmetro);

• Matriz - faz a leitura de matrizes no formato ASCII e as converte em isolinhas;

• Interpolador de Dados Pontuais - converte dados observacionais em campos numéricos regularmente espaçados.

Com o METVIEW também é possível fazer animações, de dados meteorológicos dentro de um intervalo de tempo. Outro ponto de destaque é a ferramenta de recuperação de dados que permite o interfaceamento direto a um sistema de banco de dados, seja local ou distribuído, acessado de forma transparente ao usuário.

A Figura 1 apresenta a variável altura geopotencial, tendo como ponto de referência a Antártida, visualizado no METVIEW.

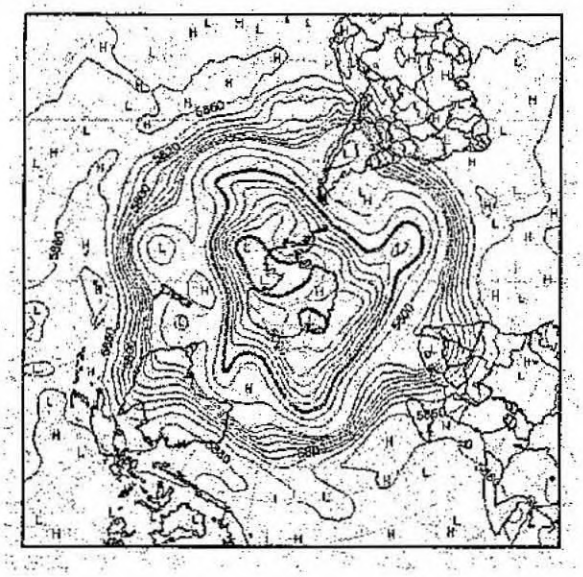


Fig.1 – Visualização da variável altura geopotencial

2.2 GrADS

O COLA (Center for Ocean-Land-Atmosphere Studies-USA) desenvolveu o Grid Analysis and Display System (GrADS) para ser um ferramenta de análise e visualização de dados de geociências. O Sistema GrADS (Doty,1995), é uma ferramenta de desktop interativa, atualmente em uso global, que permite a análise e exibição de dados de ciência da Terra. O GrADS está implementado em todas as plataformas de estações de trabalho UNIX disponíveis, igualmente em MSDOS e Windows95. Trabalha com dados de modelos em 4-D, que são linhas verticais, horizontais e tempo. Este sistema suporta dados em formato de grade não-linearmente espaçados (grades do Tipo Gaussiana e grades de modelagem oceanográfica de resolução variável) e representação de dados internos de arquivos binários ou GRIB.

As operações podem ser realizadas diretamente e interativamente nos dados, assim como executar programas em FORTRAN na linha de comando. O Sistema possui um rico conjunto de funções embutidos. Os gráficos tem saída em formato Postscript

para impressão em impressoras de postscript monocromática ou colorida. O usuário tem controle sobre todos os aspectos de saída de gráficos podendo escolher usar os parâmetros de visualização "default" do próprio sistema (exemplo: campo em contorno, projeção (lat x long), tipos de fonte, etc.).

Uma interface programável é fornecida na forma de uma interpretação de linguagem de script. Completas interfaces gráficas de dados podem e tem sido construídas. A linguagem do script pode também ser usada para automatizar cálculos multi-passo complexos ou exibições. Em versões futuras está previsto desenvolvimento de uma interface para dados em formato BUFR. O GrADS suporta dois tipos de dados meteorológicos básicos: pontos de grade e estações ou ponto observacionais.

O formato básico acessado pelo GrADS é composto de arquivos de dados (binários) e arquivos descritores (ctl's). O .ctl é construído para descrever os vários tipos de dados e estruturas, contendo todas as informações referentes aos arquivos de dados (binários e GRIB).

A Figura 2 apresentaa visualização pelo GrADS de uma imagem composta pelos satélites GOES/Meteosat, apresentando temperatura de brilho em escalas de cinza e o campo pressão ao nível médio do mar configurado em isolinhas.

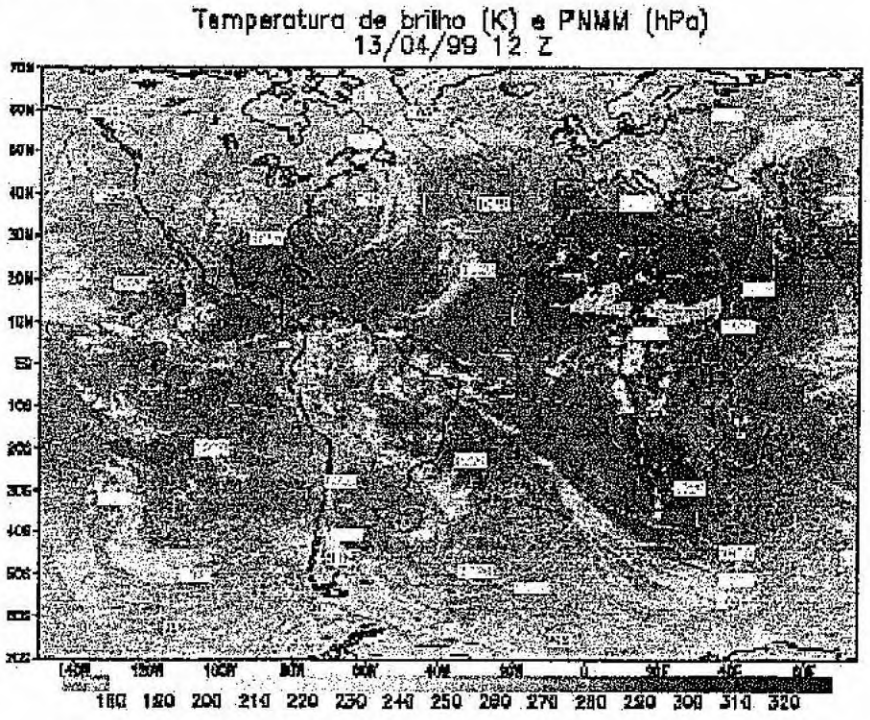


Fig.2 – Visualização da sobreposicao do campo de pressao sobre imagem composta GOES/Meteosat

2.3 Vis5D

Desenvolvido pelo Projeto de Visualização da Universidade de Wisconsin-Madison (Space Science and Engineering Center - SSEC), por Hibbard e Santek, em 1990, o sistema foi implementado em 1998, no CPTEC/INPE de Cachoeira Paulista (Brasil), onde auxilia os meteorologista a interpretarem os modelos numéricos.

Foi desenvolvido para visualização de dados gerados por modelos numéricos de tempo e similares. Opera com dados em cinco dimensões. Isto é, os dados são números reais em cada ponto da grade, o qual rotacionam em três dimensões espaciais (latitude, longitude e altura), uma dimensão temporal e uma dimensão para a enumeração de múltiplas variáveis físicas. Foi desenvolvida para trabalhar em plataforma UNIX/Linux (Hibbard e Santek, 1998).

Para seu funcionamento, este sistema necessita de uma biblioteca gráfica que é responsável pela visualização dos dados meteorológicos na tela do computador, esta biblioteca é distribuída originalmente com o software. A biblioteca gráfica original é a Mesa3-D, porém devido aos estudos sobre o Vis5D, no CPTEC/INPE, chegou-se a conclusão que utilizando a biblioteca gráfica OPENGL, obteve uma maior otimização, e explorou-se ao máximo as características da biblioteca OPENGL e da placa gráfica, no sistema.

Possui características de visualização e manipulação de dados no espaço tridimensional e nos domínios das variáveis e do tempo. Esses dados podem ser rotacionados, ampliados e animados em tempo real, além de se obter a representação de trajetórias de ventos, perfis de sondagens, isosuperfícies, planos de cortes de dados e edição de textos para publicação, entre outros.

Possuem duas janelas de trabalho:

- Painel de Controle, é responsável pelas funções primárias: as quais pode-se representar isosuperfícies, planos de cortes de dados, perfis de sondagens, traçado de trajetórias de vento, ativar/desativar topografia, box, relógio, animação, scritps, etc.
- A Janela 3-D e responsável pela apresentação de dados, permitindo ainda a rotação, animação e redução dos dados via mouse.

O sistema permite a utilização da linguagem interpretada Tcl que possibilita a geração de animações, modificação da representação dos campos meteorológicos, criação de menus e efeitos de computação gráfica. Para a inserção dos modelos meteorológicos gerados pelo supercomputador SX-3 e SX-4 instalado no CPTEC/INPE, foi desenvolvido um programa de conversão desses dados gerados no formato GrADS para o formato .v5d.

A Figura 3 apresenta o campo temperatura, visualizado no formato 3D no Vis5D.

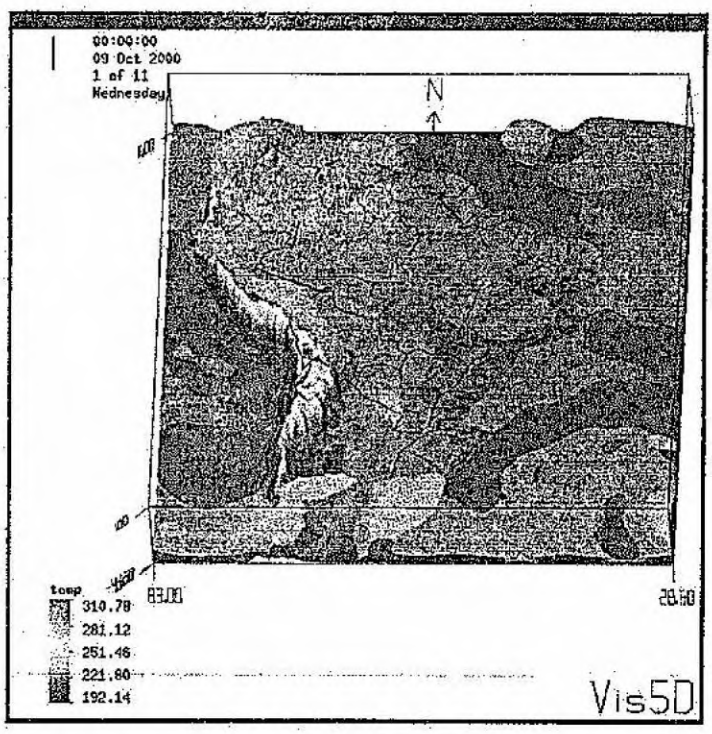


Fig.3 – Visualização de variável temperatura pelo Vis5D

2.4 VisAD

A biblioteca de classes VisAD ("Visualization for Algorithm Development") foi projetado com a intenção de permitir aos cientistas testar interativamente e visualmente resultados de seus cálculos computacionais (Hibbard e Santek, 1998).

Esta biblioteca funciona em Workstations e PC's (LINUX, WINDOWS NT e WIN95), utilizando a linguagem Java (apesar do sistema poder fazer chamadas de funções em C e Fortran). O sucesso da implementação do VisAD em Java deve-se a experiência dos projetistas nos desenvolvimentos de McIDAS e Vis5D, assim como troca de informações entre vários desenvolvedores de sistemas. O VisAD em Java possui as seguintes características:

- Independência de plataforma;
- Modelos de dados e visualização flexíveis e extensíveis;
- Suporte de programação para os usuários na análise algorítmica;
- Suporte para interfaces de usuários de disciplinas específicas e;
- Exploração da Internet através de computação distribuída e interfaces com o usuário colaborativas.

A idéia central do VisAD é oferecer grande flexibilidade ao usuário para que ele possa transformar seus dados em representação gráfica. Desta forma, a visualização é caracterizada pelo mapeamento dos elementos de um modelo de dados D nos elementos de um modelo de exibição E, sendo a função responsável por este mapeamento denominada de função de exibição M, ou seja, $M: D \rightarrow E$. Este tipo de mapeamento permite a visualização do comportamento dos dados internos de um algoritmo sem a

necessidade de tipos específicos para a lógica de exibição, ou seja, o VisAD faz a exibição dos dados internos de um algoritmo sem que seja necessário acionar rotinas gráficas que poderiam requerer tipos de dados particulares.

Os sistemas de gerenciamento de banco de dados tradicional são baseados em modelos de dados que envolvem entidades discretas e relações entre elas. Os dados científicos são diferentes porque tratam fenômenos contínuos. Modelos de natureza científicos utilizam-se de números reais de precisão infinita e funções com domínios infinitos. Contudo, esses objetos matemáticos são representados no interior do computador por dados objetos que são finitos e portanto aproximados.

O modelo de dados do VisAD é baseado na idéia que dados objetos (genericamente referenciados como tipos matemáticos) são aproximações de objetos matemáticos. Ele é dividido em coleções de tipos de dados. Os tipos de dados do modelo de dados do VisAD dividem-se em:

- Primitivos : aproximações de números reais, inteiros e textos (string);
- Complexos: tuplas finitas de outros tipos, representações de conjuntos sobre domínios definidos pelas tuplas de tipo real e amostragens finitas de funções de tipos de domínios para tipos de faixa (campos).

O modelo de dados do VisAD inclui outras formas de metadados (atribuições de dados fonte, nomes definidos por usuários e unidades para tipos primitivos de dados) e indicadores para tipos primitivos (usados como coordenadas espaciais e temporais). Esses indicadores permitem ao sistema troca entre diferentes sistemas de coordenadas espaciais e temporais (Hibbard et al., 1997).

Os dados objetos possuem um tipo de dado que é definido em termos de um conjunto de tipos primitivos de dados com nomes definidos de acordo com a aplicação específica do usuário (temperatura, pressão, latitude, longitude, tempo, etc.). Similarmente, a exibição do VisAD possui tipos complexos definidos em termos de um conjunto de tipos primitivos de exibição (coordenadas de exibição 2D e 3D, componentes de cor, transparência, índice para uma seqüência de animações, etc.). A exibição do VisAD é definida por um conjunto de mapeamentos de tipos primitivos de dados para tipos primitivos de exibição, denominados quadro de exibição de referência.

Uma seqüência temporal de imagens de satélite é usualmente exibida em um quadro de referência que mapeia as coordenadas da Terra para as coordenadas de exibição 2D, mapeia a radiância recebida pelo satélite em cores e mapeia o tempo em um índice de animação.

2.4.1 SpreadSheet

O sistema SpreadSheet é um subproduto da compilação da biblioteca de Classes VisAD e serve para a visualização genérica de dados científicos. Para a geração e utilização do sistema SpreadSheet, necessita-se do software JDK1.3.0 da Sun e da Biblioteca de Classes VisAD (Hibbard e Paul, 1998). O JDK1.3.0 é utilizado para realizar a compilação das Classes do VisAD, já que o sistema é escrito totalmente em Java, e para executar o Sistema de Visualização SpreadSheet.

O sistema SpreadSheet possui uma interface com o usuário, Painel de Visualização (Figura 4), onde são visualizados os dados. Toda vez que um dado é importado pelo VisAD, automaticamente o sistema oferece um Painel de Controle com as informações e atribuições que pertencem ao dado.

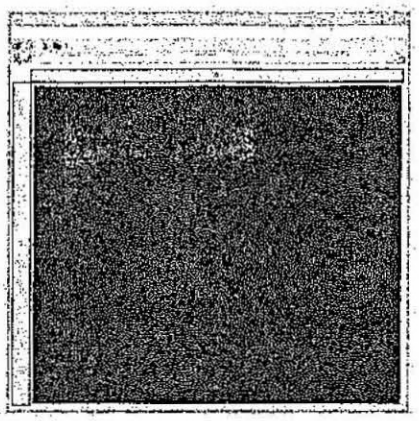
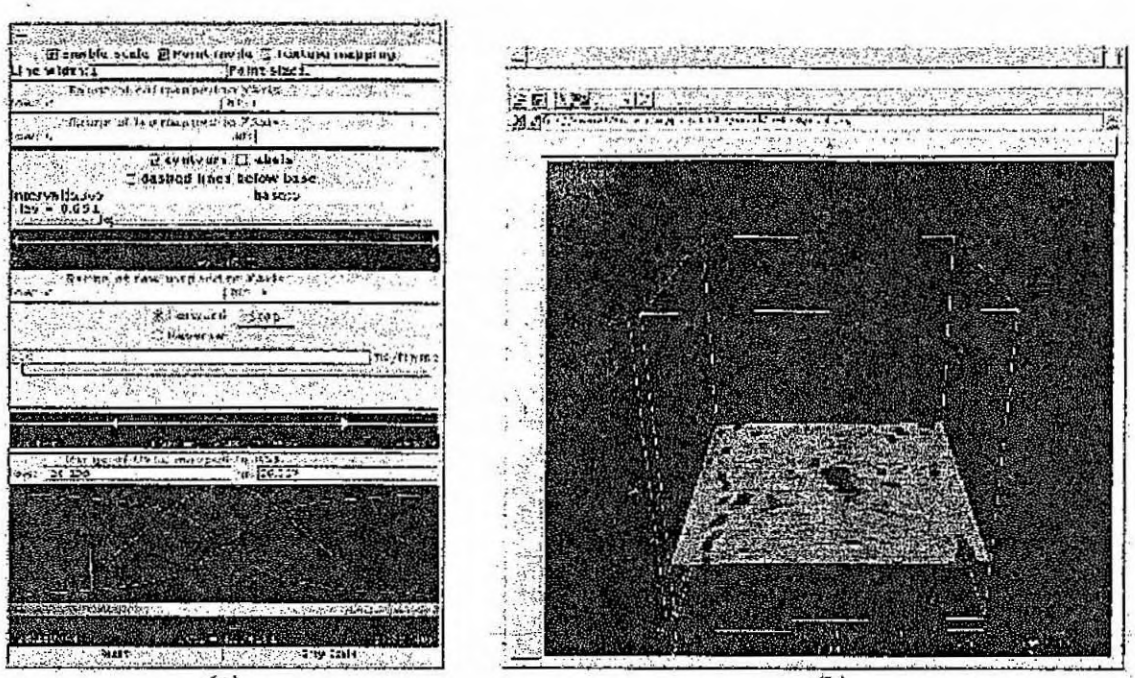


Fig.4 - Painel de Visualização do Sistema VisAD (SpreadSheet).

O Painel de Controle (Figura 5a) é gerado sempre que um dado é visualizado pelo SpreadSheet (Figura 5b). Ele possui a função de manipular interativamente o dado visualizado. Este painel é alterado toda vez que é alterado o mapeamento de uma variável através de funções de exibição do Editor de Mapeamento (Figura 6). O Editor de Mapeamento tem como característica a manipulação dinâmica dos dados, facilitando o trabalho do usuário, pois não há a necessidade de se fazer um programa para cada modificação na representação gráfica em estudo, por exemplo, transformação do espaço 2D para 3D, manipulação de cores, seleção de uma determinada faixa de dados, etc.



(a) *(b)*

Fig.5 – (a) Painel de Controle e (b) Painel de Visualização com um dado incorporado.

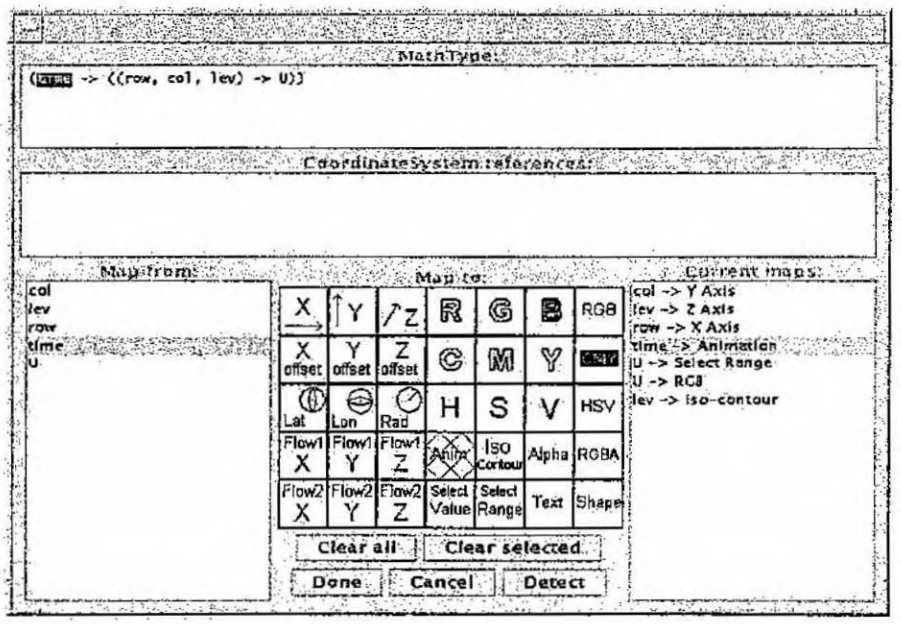


Fig.6 - Painel do Editor de Mapeamento.

3. Linguagem de Programação Orientada à Objetos

A indústria tem mostrado que as tecnologia de objetos tem sido adotadas, utilizadas e difundidas como as principais ferramentas para o desenvolvimento de softwares, onde o paradigma da orientação a objetos tem estado presente nos principais segmentos e tecnologia, incluindo processos de desenvolvimento, linguagens, arquiteturas e produtos.

O que a torna a orientação a objetos tão atraente é a possibilidade de gerenciar de forma mais simples problemas bastante complexos e de reusar códigos de forma mais eficiente. As vantagens da orientação a objetos estão relacionadas com a produção, reuso e manutenção de aplicações. Características como o alto desempenho da execução, eficiência e simplicidade serão encontradas nas linguagens Orientadas Objetos.

3.1 Conceitos Associados a Orientação à Objetos

Existem três conceitos chave na Orientação à Objetos: classe, objetos e herança.

- Classe: Classe é um tipo definido pelo usuário que contém o molde, a especificação. Ex: O tipo inteiro contém o molde para as variáveis declaradas como inteiros. A classe associa, funções e dados, controlando o acesso a estes, definí-la implica em especificar os seus atributos (dados) e seus métodos (funções). (Cesta e Rubira, 1997).
- Objetos: São a base da tecnologia orientada a objetos. Cada instância (variável) de uma classe é chamada de um objeto. Cada objeto possui seus próprios dados, mas todos os objetos de uma classe compartilham os códigos das funções membros (Carvalho e Erthal, 2000).
- Herança: É o mecanismo para compartilhar automaticamente métodos (funções) e atributos (dados) entre classes, subclasses e objetos, um poderoso mecanismo não encontrado nas linguagens tradicionais, fundamental para programação orientada objetos e um dos fatores de sucesso desta.

Desse modo uma classe define comportamento e forma padrão para a construção de objetos. Uma vez definida uma classe A, é possível definir uma nova classe B como uma extensão de A. Nesse caso, utiliza-se o conceito de herança, onde a subclasse B herda atributos e métodos da superclasse A. Uma subclasse pode rescrever métodos definidos na superclasse e criar novos atributos e métodos.

3.2 Fundamentos da Orientação à Objetos

A tecnologia baseada em objetos está fundamentada em características como:

- Abstração: Significa concentrar-se no que um objeto é e faz, antes de decidir como ele é implementado. O uso da abstração preserva a liberdade de se tomar decisões.
- Encapsulamento: É um ocultamento de informações. O conhecimento encapsulado em um objeto possibilita a separação dos seus aspectos externos dos

detalhes internos da implementação. O encapsulamento portanto, permite controlar o acesso de atributos (dados) e métodos (funções) em uma classe.

• Compartilhamento: Esse princípio dirige a criação de estruturas de herança onde procedimentos e dados que são compartilhados por um grupo de classes são descritos apenas uma vez, numa classe que é antecessor comum para cada classe no grupo.

• Polimorfismo: É a habilidade de adquirir mais de uma "forma". Um atributo (dado) de um objeto pode ter mais de um conjunto de valores e uma operação pode ser implementada por mais de um serviço.

3.3 A Linguagem Java

Java é uma linguagem simples, de fácil aprendizado ou migração, pois possui um reduzido número de construções. A diminuição das construções mais suscetíveis a erros de programação, tais como ponteiros e gerenciamento de memória via código de programação também faz com que a programação em Java seja mais eficiente. Contém um conjunto de bibliotecas que fornecem grande parte da funcionalidade básica da linguagem, incluindo rotinas de acesso à rede e criação de interface gráfica.

Baseada no paradigma da Orientação a Objetos - encapsulamento em um bloco de software dos dados (variáveis) e métodos de manipulação desses dados - a linguagem permite a modularização das aplicações, reuso e manutenção simples do código já implementado.

A linguagem Java é tanto compilada como interpretada, os programas Java são compilados em um formato intermediário chamado bytecodes. Assim, esses programas podem ser executados em qualquer sistema através de um interpretador Java. Com isso, o código precisa ser escrito e compilado apenas uma vez, pois os bytecodes gerados serão executados da mesma forma em qualquer plataforma de hardware e software.

3.4 Características da Linguagem Java

A seguir são citadas algumas características da linguagem Java:

• Orientado a Objetos: É um membro do paradigma orientado a objetos (OO) das linguagens de programação. As linguagens que aceitam este paradigma, como Java e C++, seguem a mesma filosofia básica, mas diferem em sintaxe e estilo. As linguagens orientadas a objetos oferecem muitas vantagens sobre as linguagens tradicionais. Como os objetos encapsulam dados e funções relacionados em unidades coesas, é fácil localizar dependências de dados, isolar efeitos de alterações e realizar outras atividades de manutenção, e talvez o mais importante, as linguagens (OO) facilitam a reutilização.

• É ao mesmo tempo compilado e interpretado: Os programas do Java são compilados em formato binário de código de bytes, que então são interpretados por um ambiente de execução do Java específico da plataforma em questão. Portanto é ao mesmo tempo compilado e interpretado.

• Portável: A característica de neutralidade da arquitetura Java é o grande motivo pelo qual os programas em Java são portáveis. Outro aspecto da portabilidade envolve a estrutura ou os tipos de dados inerentes da linguagem, como inteiro, string e

ponto flutuante. O compilador Java foi escrito com o próprio Java. Como seu ambiente de tempo e execução foi escrito em ANSI C, ele possui uma interface de portabilidade bem definida e concisa.

• Multitarefa: Os objetos binários de códigos de bytes do Java são formados por seqüências de execução múltiplas e simultâneas. Essas seqüências são conhecidas como contextos de execução ou processos leves. As linguagens C e C++ são membros de um paradigma de execução em seqüência única, por não oferecerem suporte a seqüências no nível de linguagem. O Java, no entanto, oferece suporte no nível de linguagem para multitarefa, resultando em uma abordagem de programação mais poderosa e de múltiplas facetas.

• Dinâmico: O projeto dinâmico permite que os programas Java se adaptem aos ambientes computacionais mutantes. Isso permite que os programadores tirem total proveito da orientação a objetos. É possível adicionar novos métodos e variáveis de instâncias em classes de bibliotecas, sem causar problemas aos programas aplicativos.

• Robusto: Quanto mais robusto um aplicativo, mais confiável ele será. Isso é desejável tanto para os desenvolvedores de software quanto aos consumidores. A maioria das linguagens (OO), como o C ++ e Java, possuem tipos bastante fortes. Isso significa que a maior parte da verificação de tipos de dados é realizada em tempo de compilação e não em tempo de execução. Isso evita muitos erros e condições aleatórias nos aplicativos. O Java, ao contrário do C++, exige declarações explícitas de métodos, o que aumenta a confiabilidade dos aplicativos.

• Simples: Um dos principais objetivos do projeto do Java foi criar uma linguagem o mais próxima possível do C ++, para garantir sua rápida aceitação no mundo do desenvolvimento (OO). Outro objetivo do seu projeto foi eliminar os recursos obscuros e danosos do C ++, que fugia da compreensão e aumentavam a confusão que poderia ocorrer durante as fases de desenvolvimento, implementação e manutenção do software. O Java é simples porque é pequeno. O interpretador básico do Java ocupa aproximadamente 40k de RAM, excluindo-se o suporte a multitarefa e as bibliotecas padrão, que ocupam outros 175k. Mesmo a memória combinada de todos esses elementos é insignificante, se comparada a outras linguagens e ambientes de programação.

• Oferece alto desempenho: Há muitas situações em que a interpretação de objetos de códigos de bytes proporciona desempenho aceitável. Mas outras circunstâncias exigem desempenhos mais altos. O Java concilia tudo isso oferecendo a tradução dos códigos de bytes para o código de máquina nativo em tempo de execução. O alto desempenho permite a implementação de aplicativos WEB em Java, na forma de programas pequenos e velozes, que podem ampliar significantemente os recursos tanto do cliente quanto do servidor. (Lemos, 1998).

4. Formatos de Dados

O Sistema VisAD foi projetado para suportar e interpretar os formatos, Netcdf, FITS, HDF-EOS, GIF, JPEG e Vis5D.

4.1 Formato FITS

O formato FITS (Flexible Image Transport System) foi adotada pela comunidade astronômica para facilitar a transferência de dados entre plataformas de

hardware e aplicações de software. O FITS é um formato capaz de armazenar vários tipos de dado, incluindo Bitmaps, arquivos ASCII, Matrizes multi-dimensionais e tabelas binárias (Murray e vanRyper, 1996).
A estrutura do arquivo contém o cabeçalho e o array de dados primário (HDU). A HDU (Header and Data Units) pode conter um cabeçalho seguido por registros de dados, ou pode conter somente um cabeçalho. Todos os dados no arquivo FITS é organizado com informações de registros lógicos de 2880 bytes de tamanho. Cada registro lógico contém 36 registros, chamado de cartões de imagem. O Cartão de imagem é um campo lógico similar ao campo binário no cabeçalho da imagem. Cada cartão contém 80 bytes de dados ASCII, que descrevem alguns aspectos de organização do dados do arquivo de dados de imagem FITS. A maioria dos cartões de imagem podem aparecer em ordem dentro do cabeçalho. O primeiro cartão de imagem deve ser o SIMPLE, seguido por BITPIX em segundo, NAXIS em terceiro e END por último. Na Figura 7 é demonstrado um exemplo de cabeçalho FITS.

SIMPLE → Palavra chave que sempre aparece em primeiro no cabeçalho do arquivo FITS. É um valor boolean que indica a conformidade do nível do arquivo para especificar o FITS.

T : FITS é padrão.
F : o arquivo é diferente e requer uma especificação do FITS padrão.

BITPIX → O cartão de imagem Bitpix contém o valor inteiro que especifica o número de bits usado para representar cada valor de dados. Para os dados de imagem, este é o números de bits por pixel.

NAXIS → Contém valores inteiros com uma extensão que varia de 0 à 999, que indicam os números do eixo de array de dados. Convencionalmente os bitmaps tem um NAXIS de valor 2. Um valor 0 significa que não há dados no cabeçalho, embora a extensão possa estar presente.

NAXISn → Cartão de imagem que indica o comprimento de cada eixo da unidade BITPIX. O valor do campo indexado pela palavra chave, contendo números não negativos inteiros, representa o número de posições ao longo do eixo n de um array de dados. Este cartão de imagem deve estar presente para todos os valores n=1,..., NAXIS.

EXTEND → O cartão de imagem EXTEND pode ser incluído se houver extensões do arquivo FITS. Se não há extensões, não haverá o cartão de imagem EXTEND.

END → Palavra chave que indica o fim de um cabeçalho e é sempre o último cartão de imagem do cabeçalho. Não possui valor. Este cartão de imagem contém espaços em branco das colunas de 9 por 80 e é preenchido com espaços de forma que o comprimento do cabeçalho seja múltiplo de 2880 bytes.

```
        1         2         3         4         5         6         7
12345678901234567890123456789012345678901234567890123456789012345678901
123456
SIMPLE  =    T
BITPIX  =    8/ 8 bits por pixel
NAXIS   =    2/ tabela da matriz 2D
NAXIS1  =    168/ número de colunas em bytes
NAXIS2  =    5/ número de linhas
DATE    =    '09/17/93'
ORIGIN  =    'O' 'Reilly & Associates' / Publisher
AUTHOR  =    'James D. Murray' / Creator
REFERENC =   'Graphics File Formats' / Where refenced
COMMENT =    'Sample FITS header'
END
```

Fig.7 - Cabeçalho do Arquivo FITS

Os dados da imagem FITS, segue imediatamente o cabeçalho em dados binários. Estes dados são armazenados em bytes que nunca são comprimidos.

A presença ou ausência do array de dados primário é indicado pelo valor de cada NAXIS ou NAXISn. Os dados de um arquivo FITS pode ser armazenados em bytes, palavras de 16 ou 32 bits, e valores de pontos flutuantes de 32 ou 64 bits conforme o padrão ANSI/IEEE-754.
O número do bits do dados de imagem, não incluindo a adição de brancos no final do dados de imagem, pode ser calculado pelos valores dos cartões de imagem: BITPIX, NAXIS e NAXISn.

NumberOfBits = BITPIX * (NAXIS1 * NAXIS2* . . . * NAXIS[NAXIS])

4.2 Formato GIF

O formato GIF tem como objetivo principal, tornar disponível um conjunto de arquivos pequenos que possam ser acessados por qualquer usuário, mesmo sob um acesso à Internet lento. Um outro propósito é colocar disponível produtos para usuários que usam softwares como o NETSCAPE. Alguns destaques do GIF:

- ganho no tempo de transmissão, e espaço na memória;
- armazena informações de 8 bits/pixel (256 cores ou menos).

O GIF utiliza o compressor LZW, que possui uma alta velocidade para este tipo de formato. O compressor cria uma tabela que será inserido os padrões encontrados à medida que o arquivo é percorrido. Quanto maior o número de padrões capturados melhor ficará a compressão (Murray e vanRyper, 1996).

A estrutura do formato GIF é determinado pelo cabeçalho e a área de dados, demonstrado pela Figura 8, informações (Hearder, Logical Screen Descriptor e Global Color Table), Image "n" (Local Image Descriptor, Local Color Table, Image Data e Trailer)

multidimensionais para cada variável para um apropriado "offset". O cabeçalho não tem nenhum espaço extra utilizável, seu tamanho é determinado através da programação, onde se determina as dimensões, variáveis e atributos do arquivo Netcdf.

A vantagem deste cabeçalho é que se torna mais compacto, pois utiliza apenas o espaço de memória necessário. Porém há uma desvantagem em termos de se modificar qualquer arquivo Netcdf, pois para realizar esta alteração ficará tão caro quanto copiar um arquivo.

O formato possui um número de ID pelo qual é identificado. Estes componentes podem ser usados juntos para capturar o significado dos dados e relações entre a área de dados e array orientados em banco de dados. A biblioteca Netcdf permite o acesso simultâneo para múltiplos arquivos Netcdf que são identificados pelo número do arquivo ID, em acréscimo para um nome comum de arquivo. O Netcdf possui uma tabela de símbolos para variáveis contendo o nome da variável, tipos de dados, rank, dimensões e o endereço inicial do disco.

Dimensões → Representa uma dimensão física real, por exemplo, tempo, latitude, longitude ou altura. Pode também ser utilizada para indexar outras quantidades, por exemplo; número da estação ou número de modelo rodado. A dimensão pode ser determinada em, estática (record dimension) ou dinâmica (unlimited dimension), possuindo um nome e tamanho (no caso da record dimension), que será sempre positivo inteiro.

Variáveis → Utilizadas para guardar os dados de um arquivo Netcdf. Representa uma quantidade de valores do mesmo tipo. Contém: nome, tipo de dado e uma forma descrita pela lista de dimensão especificada onde a variável é criada. A variável pode também ser atributos associados que podem ser adicionado, deletado ou alterado mesmo depois de ter sido criada.

Atributos → Usados para armazenar dados sobre os dados, similar a vários esquemas convencionais de sistemas de banco de dados. A maioria dos atributos fornecem informações sobre uma variável específica. São associados as variáveis como; nome, comprimento e valores. Os atributos podem ser mais dinâmico que a variável e dimensão, pois podem ser deletados e tem o tipo, comprimento e valores alterados.

A parte de tamanhos fixos de dados, segue o cabeçalho contendo todos os dados da variável para as que não empregam dimensões ilimitadas. Os dados de cada variável é armazenada continuamente nesta parte do arquivo.

A parte de registro de dados, consiste em um número de variáveis do registro, cada qual contém dados para todas as variáveis do registro. Os dados registrados para cada variável é armazenada continuamente em cada registro.

4.6 Formato Vis5D

O Vis5D é utilizado para armazenar e renderizar os dados numéricos em três dimensões. A semelhança dos dados tipicamente adquiridos de fontes científicas provenientes de dados de tempo e medidas topográficas (Murray e vanRyper, 1996). Todos os arquivos Vis5D contém três sessões de dados:

- Cabeçalho do arquivo 5D;
- A seqüência de cabeçalho de informações de grade 3D;
- Seqüência de array de dados de grade 3D.

O arquivo do cabeçalho, contém informações sobre o conteúdo inteiro do arquivo. O cabeçalho de informações de grade são diretórios com uma seqüência de dados em grades armazenados em um arquivo. Estes dados tem uma seqüência de uma ou mais grades em 3D, cada quais definidos em ponto de coordenadas em dados numéricos.

O cabeçalho de informações de grade armazena o tamanho e a localização, o tempo e data do modelo, nome da variável e a unidade descritora de grade ou ponto de grade. Cada ponto de grade é constituída de cinco pontos flutuantes que requerem valores para dados de cinco dimensões. Os cinco valores são armazenados em três localizações de coordenadas em ponto de grade, em diferença do valor de tempo e no nome da variável física do ponto de grade.

As três posições de coordenadas armazenam as dimensões espaciais (latitude, longitude e altitude), do ponto de grade 3D. O time step é a diferença entre a análise e previsão. O nome da variável é o rótulo usado para referir-se ao ponto.

Os pontos de grade são divididos em informações lógicas agrupadas de seqüência de grades. A seqüência de grades é um ou mais pontos de grade existentes no mesmo momento e tempo. Uma seqüência de grade é similar a um quadro de animação. A visualização de seqüências de grade em sucessão ordenadas formam a animação dos dados numéricos.

Toda a estrutura do cabeçalho 5D dos arquivos Vis5D tem entrada com 256 bytes que segue um formato, demonstrado na Figura 9.

```
Typedef struct_Vis5Dherader
{
   Char  Identifier [32];      /* File description field */
   Long  ProjectNumber;        /* Project number */
   Long  CreationDate;         /* Date file was created */
   Long  MaximumSize;          /* Nº of data points in largest 3D grid */
   Long  NumberOfGrids;        /* Nº of 3D grids in the data */
   Long  FirstGrid;            /* Location of first frid */
   Long  Padding [51];         /* Alignment padding */
} Vis5Dheader;
```

Fig.9 - Cabeçalho 5D do Arquivo Vis5D

Identifier → Área de 32-byte usada para armazenar um caracter nulo no final que é utilizado para identificar o arquivo e o conteúdo.

ProjectNumber → Valor utilizado para identificar o projeto no qual um arquivo Vis5D pertence.

CreationDate → Cria a data de um arquivo Vis5D no formato YYDDD. Um valor de 01h indicam que não foi criado uma data específica.

MaximumSize → Número de pontos de dados de uma grande grade 3D. Este valor é produto de um número de linhas, colunas e níveis em uma larga grade.

NumberOfGrids → Total de números de grades 3D desses dados. Esse valor é produto do número de diferenças do tempo e os parâmetros desses dados.

FirstGrid → Localização do "Offset" da posição da primeira grade de dados. Este valor de "Offset" é um número de 4-byte de valores LONG a partir da primeira grade até o início do arquivo. A primeira grade usualmente segue a última informação da grade 3D do cabeçalho do arquivo.

Padding → Utilizado para preencher o bloco do cabeçalho se necessário, por exemplo; caso um cabeçalho tenha 204 bytes o Padding irá preencher o espaço até que seja obtido o valor de 256 bytes.

Toda a estrutura do cabeçalho 3D dos arquivos Vis5D tem entrada com 256 bytes que segue um formato, demonstrado na Figura 10.

```
typedef struct _3DGridInformationHeader
{ Long Size;                    /* Number of data points */
Long NumberOfRows;              /* Number of rows */
Long NumberOfColumns;           /* Number of columns */
Long NumberOfLevels;            /* Number of levels */
Long DataLocation;              /* Location of grid data */
Long Date;                      /* Grid date stamp */
Long Time;                      /* Grid time stamp */
Long Paddingl;                  /* Alignment padding */
Long ParamName[4];              /* 4-character variable or parameter name*/
Char UnitsDesc14];              /* 4-character units description */
Long Padding2[11];              /* Alignment padding */
Long IType;                     /* Always 04h */
Long NorthLatitude;             /* North latitude * 10000 */
Long WestLongitude;             /* West longitude * 10000 */
Long LatitudeIncrement;         /* Latitude increment * 10000 */
Long LongitudeIncrement;        /* Longitude increment * 10000 */
Long Padding3[41;               /* Alignment padding */
Long IhType;                    /* Always 0lh */
Long TopAltitude;               /* Top altitude * 1000 */
Long AltitudeIncrement;         /* Altitude increment * 1000 */
Long Padding4[31];              /* Alignment padding */
} 3DGRIDINFOHEADER;
```

Fig.10 - Cabeçalho 3D do Arquivo Vis5D

Size → É igual para os números de dados de 4 bytes em pontos de grade. Este valor é sempre igual para: **NumberOfRows (latitude); NumberOfColumuns (longitude); NumberOfLevels (altitude)**, em pontos de grade.

DataLocation → As localizações dos dados na grade do arquivo armazenado como número de 4 byte para valores longos na entrada do arquivo.

Date → Marca a data dos dados da grade no formato de YYDDD.

Time → Marca o tempo da grade no formato HHMMSS.

Padding1 → Tamanho de 4 bytes, usado para alinhar os sete primeiros campos do cabeçalho, preenchendo 32 bytes de limite.

ParamName → É uma string de 4-caracter ASCII que é um nome de variável física usada para representar um ponto de grade. Este campo não termina com valor NULL e o bloco possui um espaço de 20h de caracter se necessário.

UnitsDesc → É uma string de 4-caracter ASCII que descreve a unidade de medida usado pelo ponto de grade. Este campo não termina com valor NULL e o bloco possui um espaço de 20h de caracter se necessário.

Padding2 → Tamanho de 44 bytes, usado para alinhar os 10 campos anteriores. Este campo é configurado para o valor 04h.

Itype → Sempre configurado para o valor 04h.

NorthLatitude → Latitude mais ao norte em dados de grade multiplicado por 1000.

WestLongitude → Longitude mais ao oeste em dados de grade multiplicado por 1000.

LatitudeIncrement → Incremento na latitude multiplicado por 1000.

LongitudeIncrement → Incremento na longitude multiplicado por 1000.

Padding3 → Tamanho de 16 bytes, usado para alinhar os 16 campos anteriores. Este campo é configurado para o valor 00h.

IhType → Sempre configurado para o valor 01h.

TopAltitude → Maior altitude multiplicada por 1000.

AltitudeIncrement → Incremento na altitude multiplicado por 1000.

Padding4 → Tamanho de 124 bytes, usado para os blocos de informações do cabeçalho de saída para o comprimento de 256 bytes. Este campo é configurado para o valor 00h.

5. Materiais e Métodos

Para desenvolvimento das atividades previstas na proposta inicial foi utilizado um computador Pentium 166Mhz com 64Mb de memória que possuia o sistema operacional Linux Red Hat 6.0. Inicialmente atualizou o sistema operacional para Linux Red Hat 7.2, em seguida foram instalados os softwares JDK1.3.1 (Java Development Kit - extraído site da SUN), Mesa Library, Java 3D, Java Help Software, Java WebStart Version 1.0.1, código fonte GDV (fornecido pela equipe de desenvolvimento da UNICAR) e a biblioteca de classes do VisAD (extraída do site da Universidade de Wisconsin).

Afim de visualizar e manipular os dados gerados pelo Modelo de Previsão de Tempo e Clima do CPTEC/INPE no sistema GDV, foram iniciados estudos no código fonte da biblioteca de classes do VisAD, que está totalmente escrita em linguagem de

programação orientada a objetos (Java). Utilizando a classe para manipulação de dados no formato Vis5D existente nesta biblioteca foi desenvolvido um programa para leitura de informações deste formato. O código deste programa foi posteriormente incorporado ao GDV.

Para geração deste relatório foi adquirido um conversor do arquivo no formato do GrADS para Netcdf, este conversor utiliza do pacote Netcdf e Udunits, ambos desenvolvidos pela "University Corporation for Atmospheric Research", estes pacotes fazem parte das bibliotecas geradas pelos pesquisadores da Universidade com o intuito de promover o intercâmbio entre usuários dos sistemas desenvolvidos por eles. Através da compilação destes pacotes foram gerados os arquivos da biblioteca utilizada pelo conversor.

A Figura 11 apresenta-se o GDV, interpretando e visualizando um dado no formato Netcdf, e na Figura 12 apresenta-se a interface agora implementada visualizando um dado gerado pelo modelo regional (ETA) do CPTEC/INPE.

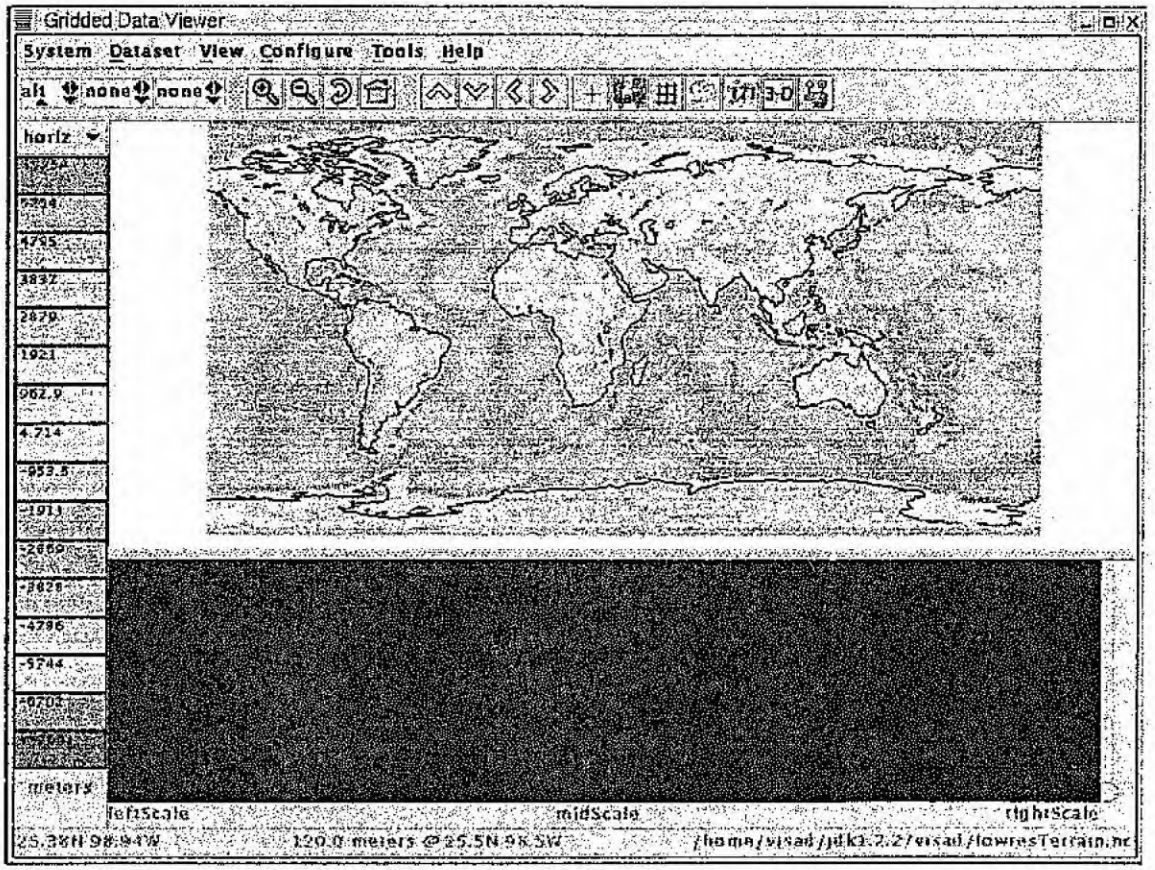


Fig.11 – Visualização da variável "PS" (Pressão na Superfície), dado fornecido com o GDV.

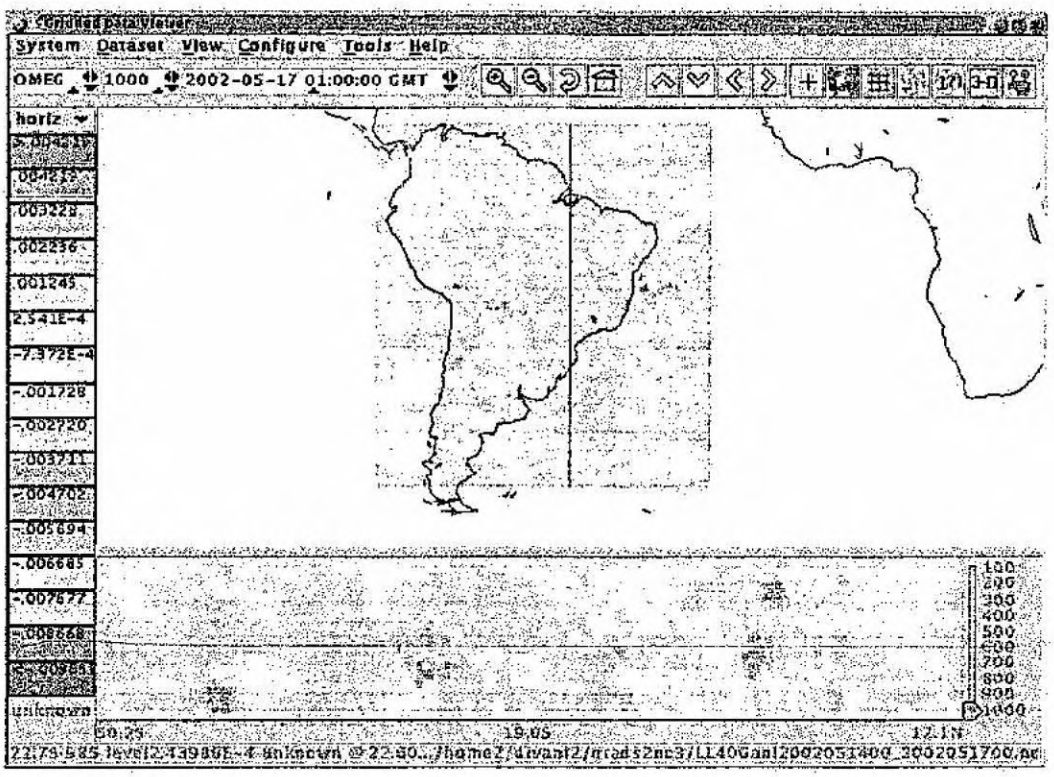


Fig.12 - Visualização da variável "OMEG" (componente vertical de vento), Modelo Regional (ETA), dado gerado em 14/05/02 previsão para 17/05/02

6. Resultados e Discussões

O sistema **GDV** é uma ferramenta de visualização de dados representados através de valores em grades regulares. Esta nova tecnologia em três dimensões visa ajudar os meteorologistas na interpretação e visualização de dados do CPTEC. Foram realizados algumas manipulações com dados gerados pelo Modelo Regional (ETA) processado pelo super computador do CPTEC/INPE. Na Figura 13, visualizamos o dado selecionando a variável "UVEL" (componente horizontal do vento). Através das funções fornecidas pela interface definem-se o nível, o tempo, a região geográfica, e a secção desejada. Na Figura 13 escolheu-se como a forma de visualização o traçado de linhas de contorno (iso contour) utilizando uma função da interface disponível no "menu Tools".

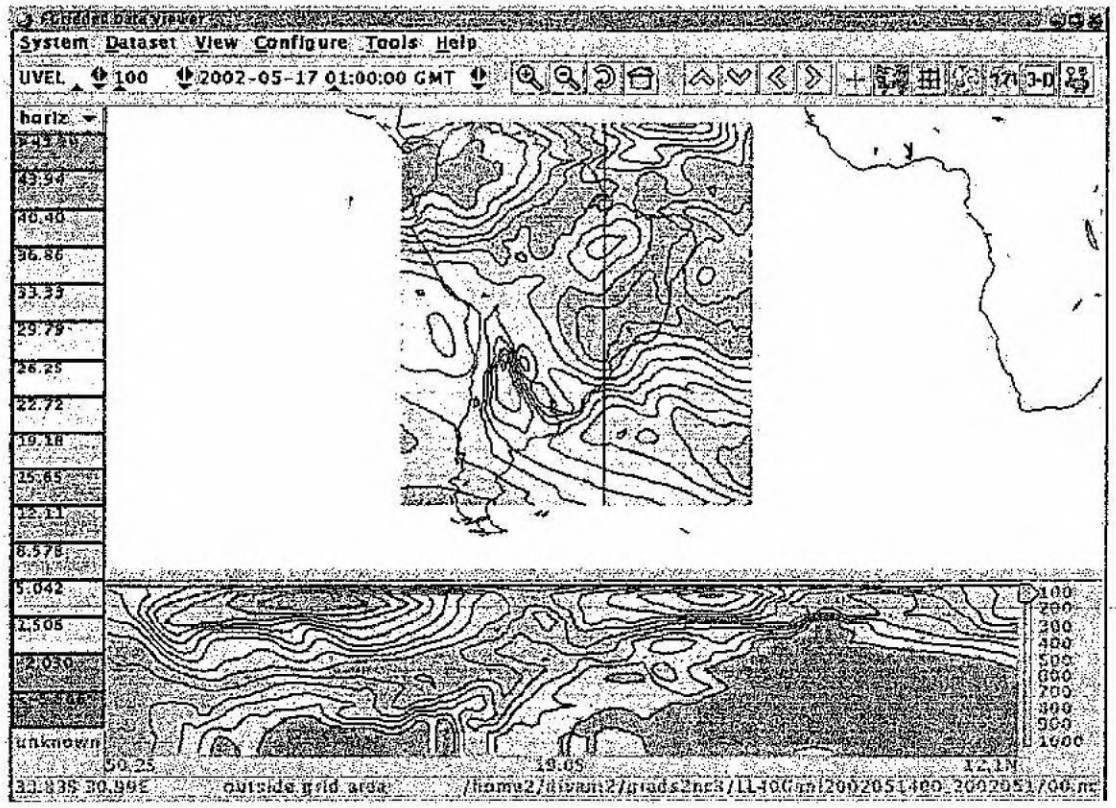


Fig.13 – Visualização da variável "UVEL", Modelo Regional (ETA), dado gerado em 14/05/02 previsão para 17/05/02.

Na Figura 14, alterou-se a variável a ser visualizada para "TEMP" (Temperatura), e acrescentou-se uma outra função, que determina os valores da linhas de contornos (iso contour), obtido também através do "menu Tools", que é chamada de Contours Labels. O painel mostra a distribuição de temperatura em 1000 hPa em 17 de maio de 2002 às 00:00 UCT. O painel de baixo mostra um corte vertical do mesmo campo em 45° W.

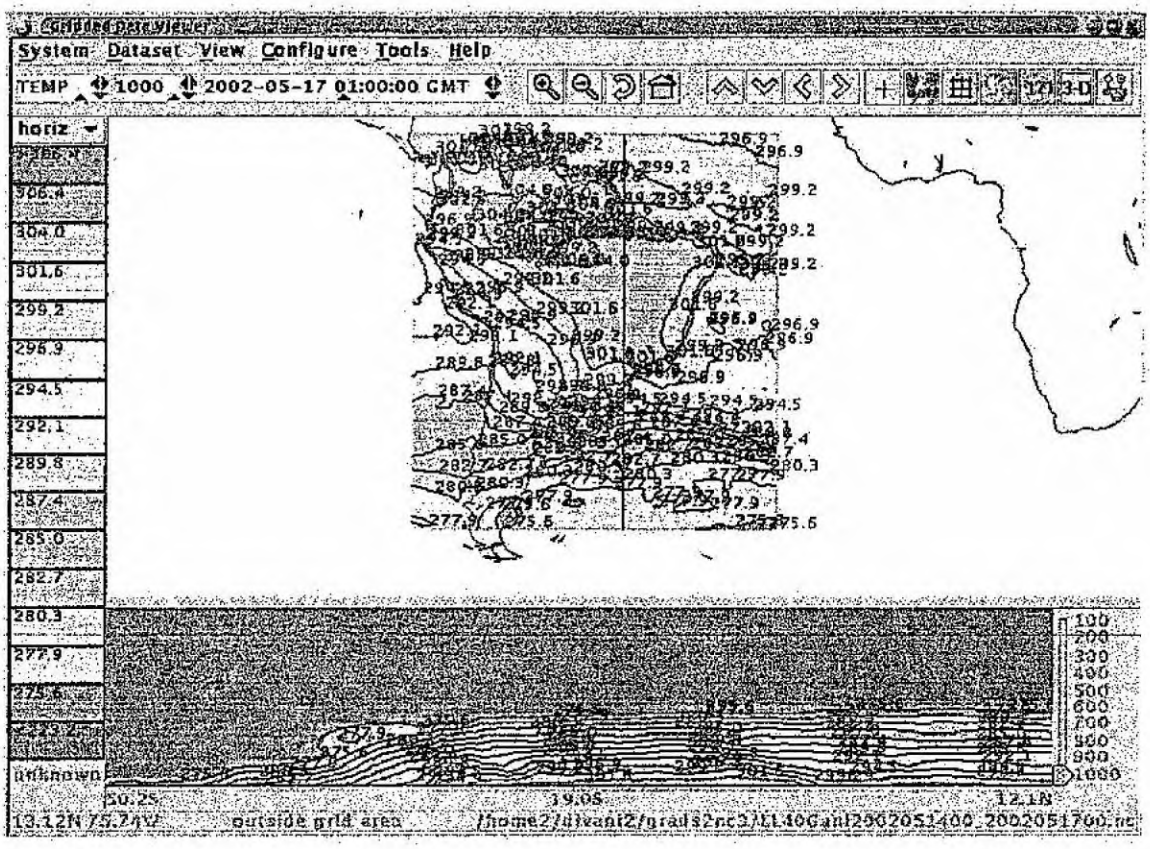


Fig.14 – Visualização da variável Temperatura, , Modelo Regional (ETA), dado gerado em 14/05/02 previsão para 17/05/02.

7. Conclusão

No CPTEC, já existem algumas ferramentas para a análise de dados científicos, porém um novo sistema foi implementado utilizando o sistema operacional LINUX, com a intenção de disponibilizar aos meteorologistas do CPTEC, mais uma ferramenta de interpretação e visualização de dados meteorológicos.

A implementação desta nova ferramenta foi possível, pois esta interface utiliza da Biblioteca de Classes do VisAD, um sistema de visualização já implementado no CPTEC, dando continuidade aos estudos iniciais. Mas a maior dificuldade foi a adaptação desta interface para visualização dos dados do Modelo de Previsão de Tempo e Clima do CPTEC/INPE, uma vez que quando foi adquirida só visualizava dados no formato padrão utilizado pela NASA

Durante os estudos e as manipulações na interface, observou-se uma grande facilidade no manuseio da mesma, podendo agilizar a previsão e facilitando o trabalho do meteorologista. No entanto é necessário dispor de uma máquina com bons recursos computacionais, visto que em se tratando de visualização gráfica o volume de dados costuma ser grande.

O bolsista adquiriu, no desenvolver das atividades descritas nesse relatório, um crescimento nos conhecimentos gerais, particularmente nas áreas de programação orientada objetos, linguagem de programação Java, previsão numérica de tempo e sua visualização gráfica. Dificuldades que venham a ser encontradas serão mais facilmente resolvidas, devido a experiência obtida no decorrer do trabalho. De grande valia está sendo a elaboração desse relatório, pois traz novos conhecimentos na redação e formatação de trabalhos científicos.

8. Referência Bibliográfica

Cartaxo, J.; Daabeck, et al. **METVIEW Home Page.** [online] <http://www.cptec.inpe.br/products/METVIEW>. nov. 1998.

Carvalho, S. V.; Erthal, G.J. **Programação em C++.** Instituto Nacional de Pesquisas Espaciais, jan. 2000.

Castro, M. A. S. **Tutorial de Html da USP.** [online] <http://www.icmsc.sc.usp.br/manuals/HTML/exgifjpeg.html>. set. 1998.

Centro de Previsão de Tempo e Estudos Climáticos. [online] <http://www.cptec.inpe.br>. abr. 2001. (CPTEC-INPE).

Cesta, A. A.; Rubira, C. M. **Tutorial a linguagem de programação Java.** [online] <http://www.dcc.unicamp.br/~cmrubira/aacesta/java/>. jun. 1997.

Doty, B. **Center for Ocean-Land-Atmosphere Studies.** [online] <http://GrADS.iges.org/GrADS/head.html>. nov. 1995. (COLA).

HDF-EOS Standards and Tools Information Center. [online] <http://hdfeos.gsfc.nasa.gov/hdfeos/workshop.html>. nov. 1998. (NASA)

Hibbard, B.; Santek, D. **Vis5D Home Page.** [online] <http://www.ssec.wisc.edu/~billh/Vis5D.html>. out.1998.

Hibbard, B.; Paul, B.**VisAD Home Page.** [online] <http://www.ssec.wisc.edu/~billh/VisAD.html>. out. 1998.

Hibbard, W. J.; Anderson, J.; Paul, B. E. **Java and World Wide Web Implementation of VisAD:** Conf. Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology, 174-177, 1997,

Kim,I. S.; Quadro,M. F. L.; Marengo, J. A. **Sobre Interpretação Estatistíca da Saída do Modelo Numérico.** X Congresso Brasileiro de Meteorologia. Brasília: CD ROM., 1998.

Lemos, I. **Apostila Introdução sobre o Java.** [online] <http://www.geocities.com/Pentagon/9286/Java.htm>. nov. 1998.

Murray, J. D.; vanRyper, W. **Encyclopedia of Graphics File Formats, Second Edition**, Editora: O'Reilly & Associates, Inc., 103 Morris Street, Suite A., Sebastopol, CA 95472, abr. 1996.

Rew, R.; Davis, G.; Emmerson, S. **Netcdf User's Guide**. [online] <http://www.atmos.colostate.edu/html/Netcdf/guide.txn_toc.html>. nov. 1998.

Satyamurty, P.; Bitencourt, D. P. **Previsão Numérica de Tempo no CPTEC**. [online] <http://www.cptec.inpe.br/products/climanalise/cliesp10a/daniel.html>. out. 1996.

Sun Microsystems, Inc. **The Java Tutorial:** A practical guide for programmers. [online] <http://java.sun.com/docs/books/tutorial/index.html>. jun. 2001. (SUN).