TIPOS ABSTRATOS DE DADOS EM COMPUTAÇÃO GRÁFICA E PROCESSAMENTO DE IMAGENS

Gilberto Câmara *
João Ricardo de Freitas Oliveira **
Fernando Yutaka Yamaguchi ***
Flávio Roberto Dias Velasco ****
Ricardo Cartaxo Modesto de Souza *****

DEPARTAMENTO DE PROCESSAMENTO DE IMAGENS
INSTITUTO DE PESQUISAS ESPACIAIS
Caixa Postal 515
12.201 - São José dos Campos (SP)

SUMÁRIO

Este trabalho discute a utilização dos conceitos de tipos abstratos de dados no desenvolvimento de software para aplicações em Computação Gráfica e Processamento de Imagens. É feita uma análise das características gerais do desenvolvimento de programas nestas áreas, com alguns exemplos. A seguir, propõe-se uma metodologia de projeto utilizando tipos abstratos de dados. Finalmente, é apresentado um exemplo mais completo do uso de abstrações de dados no desenvolvimento de uma norma GKS.

ABSTRACT

This paper describes the use of abstract data types on software development for applications in the areas of Computer Graphics and Image Processing. An analysis of the general characteristics of software development in this area is made, and some examples given. A project methodology for these areas is also proposed, using abstract data types. Finally, a design example on the development of a GKS standard using data abstractions is given.

---

* Engenheiro Eletrônico, ITA, 1979. Ms.C., Computação Aplicada, INPE, 1982. Áreas de Interesse: Processamento de Imagens, Computação Gráfica, Engenharia de Software, Bancos de Dados Não-Convencionais.
** Engenheiro Eletrônico, ITA, 1977. Ms.C., Ciência Espacial, INPE, 1981. Áreas de Interesse: Computação Gráfica, Engenharia de Software.
*** Engenheiro Elétrico, FVE, 1984. Áreas de Interesse: Computação Gráfica, Processamento de Imagens.
**** Engenheiro Eletrônico, ITA, 1970. Ms.C., Engenharia de Sistemas, UFRJ, 1973. Dr., Computação Aplicada, INPE, 1977. Áreas de Interesse: Processamento de Imagens, Inteligência Artificial, Engenharia de Software.
***** Engenheiro Eletrônico, ITA, 1975. Áreas de Interesse: Processamento de Imagens, Computação Gráfica, Arquitetura de Computadores.

1. INTRODUÇÃO

Este trabalho analisa a utilização de conceitos de tipos abstratos de dados nos ambientes de Computação Gráfica e Processamento de Imagens. Sua motivação foi o estabelecimento de metodologia de desenvolvimento de software no ambiente do Departamento de Processamento de Imagens do Instituto de Pesquisas Espaciais - INPE.

A metodologia proposta utiliza conceitos de tipos abstratos de dados (TAD), e pretende ser eficaz para uma variada gama de projetos nas áreas de Computação Gráfica e Processamento de Imagens (GCPI).

O uso de TADs representa um avanço recente na Engenharia de Software (Shaw, 1984; Brooks, 1987) e permite organizar os programas em módulos de tal forma a esconder decisões de projeto e os detalhes de implementação. No DPI/INPE, a metodologia de TADs é utilizada de maneira sistemática no desenvolvimento de software, com grandes benefícios, incluindo maior produtividade, transportabilidade e reusabilidade do software.

2. CARACTERÍSTICAS DE "SOFTWARE" PARA IMAGENS E GRÁFICOS

2.1 DA NATUREZA DO PROCESSO

Numa formulação bastante geral, os sistemas de processamento de imagens e de gráficos (PICG) tratam com objetos pictóricos e seus atributos. Estes objetos podem estar representados tanto na forma de varredura (imagens digitais), quanto na forma vetorial (cartas digitalizadas), como nas duas maneiras. Exemplos de objetos pictóricos seriam uma imagem de satélite, um descrição de um automóvel, uma carta de uso do solo, ou uma representação de uma imagem.

O procedimento típico de uma sessão de trabalho num sistema deste tipo é a aplicação de uma sequência de operações a um ou mais objetos pictóricos, de forma a obter, ao final, um novo objeto ou uma descrição do objeto manipulado. Nesta sequência, os objetos em estudo estão sendo continuamente transformados, combinados ou sintetizados. Este tipo de procedimento difere sobremaneira do paradigma de um sistema de aplicações comerciais, onde são feitas operações paralelas e assíncronas, tais como consulta, inserção e exclusão a uma base de dados.

Métodos propostos na área de Engenharia de Software, desenvolvidos para sistemas comerciais, devem ser analisados com cuidado, antes de se fazer uma transposição pura e simples para o campo de processamento pictórico. Como exemplo, a metodologia de Análise Estruturada, de grande sucesso e larga aplicação na área comercial, não se adequa de forma natural aos problemas acima mencionados. A escolha de uma metodologia para o campo de CGPI não pode, assim, deixar de levar em conta que o seu paradigma de trabalho é fundamentalmente diferente da área comercial.

Um aspecto relevante neste tipo de sistema, que o difere de um sistema comercial, é a grande variedade de objetos presentes. Enquanto que os objetos de um sistema comercial apresentam uma regularidade previsível e representável por estruturas simples (tais como tabelas de empregados e departamentos), os sistemas de CGPI podem conter objetos complexos, de formatos e tamanhos bastante distintos entre si.

A rápida evolução do hardware para CGPI coloca uma dificuldade adicional, pois é preciso garantir a portabilidade do software para uma vasta gama de dispositivos gráficos.

2.2 USO DE TIPOS ABSTRATOS DE DADOS EM CGPI

Os conceitos de tipos abstratos de dados aplicam-se de forma natural à área de CGPI. Cada um dos objetos pictóricos complexos presentes será descrito por um TAD; caso o objeto seja composto de um ou mais sub-objetos, estes também podem ser representados por TADs.

Os objetos pictóricos definidos por TADs (e suas operações) compõem um ambiente de programação de grande poder expressivo, o que simplifica a tarefa do programador de aplicação. O desenvolvimento de aplicações complexas se torna mais confiável e mais rápido.

Cada TAD será definido de forma a isolar decisões de projeto, seguindo o princípio de "ocultamento de informação" (Parnas, 1972). Deste modo, a descrição dos objetos pictóricos será separada de sua estrutura e implementação internas. Isto permite ao programador de aplicação manipular as entidades pictóricas sem necessidade de preocupar-se com a maneira como tais entidades são representadas internamente no sistema.

Com os conceitos de TADs, pode-se abordar de forma adequada os dois principais problemas do desenvolvimento de software para CGPI: complexidade e transportabilidade.

3. EXEMPLOS DE ABSTRAÇÕES DE DADOS EM CGPI

Em Processamento de Imagens, a abstração mais importante é a imagem, que pode ser vista externamente como uma fotografia digital, que congela uma realidade instantânea. Assim, a descrição de uma imagem deve conter uma identificação além dos valores digitais que correspondem a cada elemento de imagem (pixel).

Como exemplo do uso de TADs, será considerado o conjunto de operações necessário para copiar uma imagem de um dispositivo de entrada para outro local. Estes dispositivos podem ser gráficos ou não. Para tal fim, é necessário cunhar as seguintes abstrações adicionais:

- Janela: indica a região do dispositivo de entrada (ou saída) onde a imagem deve ser lida (ou escrita).

- Dispositivo: definidos logicamente (isto é, para cada periférico do sistema pode estar associada uma identificação).

Uma imagem vista num dispositivo gráfico pode ser o resultado do acesso a dados que estão em outro local (em disco, por exemplo). Assim, para cada imagem, pode-se definir um imagem-mãe (onde estão os dados de origem).

No que segue, o TAD imagem é descrito e são dados exemplos de operações possíveis. A notação utilizada é derivada de Veloso (1987).

TAD Imagem

- Domínio :

```
IMAGEM      imagem_mãe, imagem_trab;
JANELA      jan_leitura, jan_escrita;
DISPOSITIVO disp_ent, disp_saida;
```

- Operações:

Imag_seleciona_mãe (imagem_mãe, imagem_trab)

Efeito : seleciona a imagem de referência.

Imag_escol_amb_ler (imagem_trab,disp_ent,jan_leitura)

Efeito : Escolha do ambiente de leitura da imagem (dispositivo de entrada e janela de leitura).

Imag_escol_ambien_esc
(imagem_trab, disp_saida, jan_escrita)

Efeito : Escolha do ambiente de escrita da imagem

Imagem_transf (imag)

Efeito : Transfere os dados de imagem do ambiente de entrada (imagem-mãe, coordenadas e dispositivo), para o ambiente de saída.

Embora isto seja transparente para o programador de aplicação, haveriam profundas diferenças na implementação desta última operação, dependendo do dispositivo. Para exemplificar, são considerados tres casos distintos:

- leitura e escrita em disco: a imagem é implementada como uma matriz 2D de linhas e colunas (acessível pelas linhas). As linhas da imagem de entrada são lidas e escritas sequencialmente.

- leitura e escrita em dispostivo dotado de controlador gráfico sem "inteligência" (como o GDC NEC 7220): é feita uma movimentação de blocos na memória do dispositivo gráfico, no modo pixel-a-pixel (ponto a ponto).

- leitura e escrita em dispositivo dotado de processador gráfico com "inteligência" (como o TI 34010): esta operaçào é implementada em uma única instrução (PXLBLT - "pixel bloc transfer").

Além de auxiliar o desenvolvimento, o uso desta abstração nos ajudaria a garantir independência de dispostivos, e deste modo, assegurar a transportabilidade do software em uma área onde a evolução do hardware é extremamente rápida.

Um exemplo de uso de TADs em Computação Gráfica é o desenvolvimento do Sistema de Informações Geográficas - SIG. O SIG é um banco de dados geográficos, que permite adquirir, armazenar, combinar, analisar, e recuperar informações codificadas espacialmente.

Um sistema de informações geográficas contém uma grande variedade de representações de dados cartográficos, e para tratar dados de formatos distintos de forma unificada, foi cunhada a abstração plano de informação. Um Plano de Informação (PI) reúne todas as representações possíveis para um mesmo dado geográfico, o que simplifica em muito o tratamento pelo sistema. A altimetria, o uso do solo, a hidrografia e a rede elétrica são exemplos de quatro possíveis PI.

No SIG, um PI correspondente à altimetria pode conter as representações: vetorial (isolinhas), amostras 3D (amostras esparsas), árvore 2D (amostras ordenadas para facilitar a busca na operação de interpolação), grade regular (resultado da interpolação) e imagem (arquivo no formato de varredura).

4. DISCIPLINA DE PROGRAMAÇÃO NA LINGUAGEM C

Os ambientes do DPI/INPE usam a linguagem C, de uso corrente em CGPI. O correto uso da linguagem C requer o estabelecimento de uma disciplina de programação, de vez que esta linguagem suporta apenas parcialmente o uso de TADs. Pode-se dividir o uso de TADs em duas partes: o programador de aplicação e o programador básico.

O programador de aplicação deve utilizar-se apenas de duas ferramentas : o comando include (que torna conhecido ao programa a definição do TAD) e as operações constantes numa biblioteca de TADs. O acesso a estruturas internas do TAD, embora permitido na linguagem C, deve ser visto como violação dos princípios de abstração de dados e evitado.

O programador básico deve implementar os TADs fazendo uso do construtor typedef; cada função definida sobre um TAD deve ter um valor de retorno, distinguindo-se dois casos: operações, que podem retornar abstrações e predicados, que retornam valores booleanos.

5. USO DE TADs NA IMPLEMENTAÇÃO DO GKS

A norma GKS consiste em rotinas de interface gráfica que dão ao programador de aplicação a habilidade de criar saídas e aceitar entradas gráficas, de uma variedade de dispositivos. O programa de aplicação é independente de dispositivo, e as rotinas do GKS comunicam-se com cada periférico por meio de "drivers" (controladores). A complexidade dos "drivers" é dependente da capacidade de processamento de cada equipamento gráfico.

Para implementar o padrão GKS no DPI/INPE, optou-se por utilizar TADs, pelas razões abaixo:

- Evolução: como o GKS comporta vários níveis de implementação de complexidade crescente, os TADs definidos para um determinado nível podem ser utilizados como elementos básicos para TADs mais complexos, no desenvolvimento de níveis superiores.

- Transportabilidade: o produto final será utilizado tanto para dispositivos simples (tais como plotadoras de pena com controle ponto-a-ponto), como para placas gráficas inteligentes (com processadores gráficos como o TI 34010).

Para exemplificar a utilização de TADs na implementação do GKS, será mostrada parte da rotina básica de traçado do GKS (chamada traçar_polilinha ou gpl em FORTRAN). Esta rotina utiliza o conceito do GKS de "transformation pipeline" (sequência de transformação).

Para garantir a independência do dispositivo, os dados são inicialmente transformados da coordenadas de referência do usuário (chamadas de coordenadas do mundo) para um sistema de referência normalizado. Esta transformação é chamada transformação de normalização. A seguir, os dados são mapeados para o sistema de referência físico do dispositivo, num processo chamado transformação de estação.

Esta rotina foi desenvolvida tomando por base, entre outras, tres abstrações e suas operações, descritas a seguir:

A) **CAIXA**

Descrição :

A abstração **CAIXA** é utilizada para representar um retângulo, definido através das coordenadas de seu vértice inferior esquerdo e de seu vértice superior direito. Utilizações típicas desta abstração incluem retângulos que definem janelas, viuportes e recortes.

Estrutura :

```
typedef struct
{
 float      x_inf,
            y_inf,
            x_sup,
            y_sup;
}CAIXA_TIPO, *CAIXA;
```

B) **TRANSF**

As transformações de normalização e de estação são definidas a partir de duas abstrações **CAIXA** : uma janela e um viuporte.

```
typedef struct
{
    CAIXA       janela;
    CAIXA       viuporte;
}TRANSF_TIPO, *TRANSF;
```

```
g_transf_cria(transf)

TRANSF   transf;
```

modifica : transf.

efeito :
Carrega em TRANSF todas as informações necessárias que definem uma transformação (de normalização ou de estação).

C) **PONTOS**

A abstração **PONTOS** é utilizada para representar uma sequência de pontos, através de dois vetores, x e y, contendo as coordenadas dos pontos a serem traçados, e de um escalar num_pontos, representando o número de pontos dados.

```
typedef struct
{
    int       num_pontos;
    float     *x,
              *y;
}PONTOS_TIPO, *PONTOS;

g_pontos_aloca(num_pontos)
int      num_pontos;
```

devolve : **PONTOS**.

efeito : Inicialmente aloca memória para a abstração **PONTOS**, em seguida, baseado no valor num_pontos, aloca memória para x e y.

```
g_pontos_cria(pontos,x,y)
PONTOS   pontos;
float    x[];
float    y[];
```

modifica : pontos.

efeito : carrega os vetores de entrada x e y na abstração PONTOS.

```
g_pontos_transf_exec(pontos,transf)

PONTOS    pontos;
TRANSF    transf;
```

modifica : pontos.

efeito :
Executa transformação nas coordenadas pertencentes a pontos, armazenando o resultado na mesma abstração.

No que segue, o trecho (na linguagem de programaçàо C) que implementa a função traçar_polilinha é mostrado. Para efeitos de simplificaçàо, apenas a parte relevante do código foi incluída. Também supõe-se a existência de um procedimento g_driver_pontos, que escreve diretamente uma sequência de pontos num determinado dispositivo.

```
gpl(num_pontos,x,y)
int     num_pontos;          /* numero de pontos  */
float   x[],                 /* array em x */
        y[];                 /* array em y */
{
    PONTOS          pontos_poli;
    TRANSF          transf_n,transf_e;
...
    /* ALOCA ESPAÇO PARA PONTOS */
    pontos_poli = g_pontos_aloca(num_pontos);

    /* COLOCA X E Y EM PONTOS */
    g_pontos_cria(pontos_poli,x,y);

    /* CRIA TRANSF NORM */
    g_transf_cria(transf_n);

    /* NORMALIZA */
    g_pontos_transf_exec(pontos_poli,transf_n);

    /* CRIA TRANSF. DE ESTACAO */
    g_transf_cria(transf_e);

    /* TRANSFORMACAO DE ESTACAO */
    g_pontos_transf_exec(pontos_poli,transf_e);

    /* CHAMA O DRIVER */
    g_driver_pontos(num_est,pontos_poli);
...
}
```

6. CONCLUSÕES

A metodologia de tipos abstratos de dados tem sido usada com sucesso em vários projetos de desenvolvimento de software e de pesquisa no Departamento de Processamento de Imagens do INPE. As vantagens observadas na adoção desta metodologia num ambiente de produção foram:

- Aumento de produtividade.
- Melhor qualidade e rapidez na documentação do software.
- Reutilização de tipos (e operações básicas) em vários projetos.
- Facilidade na implementação nos programas de aplicação.
- Garantia maior de independência de dispositivo.
- Divisão de trabalho mais adequada nas equipes de projeto, pois é mais fácil atribuir aos analistas mais experientes a responsabilidade sobre partes críticas do software.

Com base na experiência adquirida, planeja-se adotar a metodologia de TADs em todos os projetos do DPI/INPE. Como a linguagem utilizada é "C", que não prevê mecanismos que suportem diretamente TADs, foi necessário utilizar certas construções especiais e adotar uma disciplina de programação para suprir as deficiências da linguagem.

Para o futuro, espera-se que as bibliotecas de TADs construídas venham a se tornar o cerne de uma biblioteca de software que possibilite a reutilização sistemática de TADs (Meyer, 1987). Em termos de evolução de prática de programação em ambientes de produção, os TADs são um passo natural para utilização do paradigma de programação orientada para objetos (Goldberg and Robson, 1983; Stroustrup, 1988; Halbert and O'Brien, 1987).

7. REFERÊNCIAS BIBLIOGRÁFICAS

Brooks,F. "No Silver Bullet: Essence and Accidents of Software Engineering". IEEE Computer, April 1987, pp.10-20.

Goldberg,A.; Robson,B. Smalltalk-80: The Language and its Implementation. Addison-Wesley, New York, 1983.

Halbert,B.; O'Brien, P. "Using Types and Inheritance in Object-Oriented Programming", IEEE Software, Vol.4, No.5, September 1987, pp. 71-79.

Meyer,B. "Reusability: The case for Object-Oriented Design". IEEE Software, March 1987,pp. 50-64.

Parnas,D. "On the Criteria to be Used in Decomposing Systems into Modules", Communications of the ACM, December 1972, pp. 1053-1058.

Shaw,M. "Abstraction Techniques in Modern Programming Languages". IEEE Software, Vol.1, No.4, October 1984, pp.10-28.

Stroustrup, B. "What is Object-Oriented Programming ?", IEEE Software, Vol.5, No.3, May 1988, pp. 10-20.

Veloso,P.A.S. Estruturação e Verificação de Programas com Tipos de Dados, São Paulo, Edgard Blucher, 1987.