

Trabalho Prático n. 2

Programação com *Quadrees*

Métodos de Programação I – 2003/04

Preâmbulo

Este trabalho prático é constituído por duas secções. A primeira contém questões obrigatórias, e a segunda questões para valorização. Sempre que possível, deverá escrever as funções pedidas utilizando os padrões de recursividade (anamorfismos, catamorfismos, hilmorfismos) adequados.

Considere como origem das coordenadas numa imagem rectangular o seu vértice superior esquerdo.

Introdução

Uma *quadtree* é simplesmente uma árvore quadrática etiquetada nas folhas, mas contendo normalmente (no contexto que consideraremos) também informação nos nós. Um tipo de aplicação muito em voga destas árvores concerne a representação de informação espacial (por exemplo no domínio dos Sistemas de Informação Geográfica, mas também nos motores de animação utilizados em jogos 3D de última geração).

Por informação de natureza espacial entende-se uma distribuição de um conjunto de objectos numa determinada representação de um *terreno* (por exemplo) Um dos métodos possíveis para a representação deste tipo de informação utiliza uma divisão recursiva do terreno em sub-terrenos, divisão esta que pode ser representada por uma árvore.

Neste trabalho prático ocupar-nos-emos com a aplicação de *quadrees* na representação e processamento de imagens.

I – *Quadrees* na representação de imagens

Existem dois métodos essenciais para a representação de imagens a cores:

1. Como *bitmaps*, matrizes bidimensionais contendo em cada posição informação relativa um pixel (poderá ser um booleano (bit) caso se trate de uma imagem a preto e branco; um valor inteiro caso se trate de uma imagem em *grayscale* ou uma lista de 3 inteiros no caso das imagens a cores).
2. Como *mapas vectoriais*, sendo então cada imagem descrita por uma lista dos objectos que a constituem (por exemplo, quadrado de cor azul de lado 30 e coordenadas (20, 10) para o vértice inferior esquerdo).

As *quadrees* proporcionam um terceiro método: uma optimização dos *bitmaps* em que se representa de forma atómica informação relativa a todo um bloco (rectangular) de pixels. A ideia é considerar a imagem (que se considera ter uma forma rectangular) dividida em quatro quadrantes iguais; cada um destes é em si uma imagem rectangular que se divide de igual forma. A imagem assim dividida corresponde a um nó da *quadtree*, em que se armazena informação geométrica, nomeadamente as coordenadas de dois vértices opostos. Os seus descendentes são (as árvores correspondentes a) os seus 4 quadrantes.

Quadrantes (ou *secções*) constituídos apenas por pixels da mesma cor não precisam de ser divididos: basta representá-los por folhas da árvore, onde é guardada informação relativa à cor. Naturalmente, em imagens com grandes manchas da mesma cor, é possível poupar espaço considerável por comparação com a representação por *bitmap*. No caso limite, a secção mais pequena é um pixel.

Tarefas

Pretende-se escrever as seguintes funções:

1. (`mkQTree :: Bitmap -> QTree`) criação de uma *quadtree* a partir de um *bitmap* fornecido
2. (`scale :: Int -> QTree -> QTree`) operação de ampliação de uma imagem, segundo o factor de multiplicação fornecido. Esta função poderá ser generalizada para admitir reduções de imagens (factor negativo) e ainda factores de ampliação fraccionais (por exemplo 1.5 ampliará a imagem aumentando ambos os seus lados em 50%).
3. (`mirrorV, mirrorH :: QTree -> QTree`) operações de rebatimento vertical e horizontal

4. (`rotateD, rotateR :: QTree -> QTree`) operações de rotação de 90 graus nos dois sentidos
5. (`mapColour :: (Colour -> Colour) -> QTree -> QTree`) mapeamento uniforme de cor em toda a imagem

II – Sobreposição de Imagens e Utilização de Informação Vectorial

Escreva agora as seguintes funções:

1. (`sp :: QTree -> QTree -> QTree`) operação de sobreposição de árvores. Deverá alterar o tipo de dados para lidar com a noção de cor transparente (i.e. uma cor visualizada como branco, mas que quando sobreposta a outra permite “ver” a cor subjacente).
2. Funções para desenhar formas geométricas sobre uma imagem. Deverá ser possível, por exemplo, a seguinte invocação:

```
drawRectangle r pos t
```

que permite acrescentar à *quadtrees* `t` o rectângulo `r`, sendo o seu vértice superior esquerdo colocada nas coordenadas dadas por `pos`.

Comece por considerar apenas o caso dos rectângulos com lados paralelos às margens da imagem; para isso defina um tipo adequado (contendo as dimensões e cor do rectângulo).

A estrutura das *quadtrees* permite efectuar esta operação de forma muito optimizada; deverá ter isto em conta na função que escrever. Em seguida poderá generalizar a função para lidar com outras formas geométricas.

Tipos de Dados a Utilizar

- Tipo de dados a utilizar para a representação de *bitmaps*:

```
type Comp = Int
type Colour = (Comp, Comp, Comp)
data Pixel = Pix Colour
data BitMap = Bmp (Int, Int, [[Pixel]])
```

As componentes (R, G, B) de uma cor são inteiros com valor entre 0 e 255. Num *bitmap* `Bmp(r, c, l)`, `l` é uma lista de comprimento `r`, contendo listas de comprimento `c`. Cada uma destas listas representa uma linha da imagem.

- Tipo de dados *Quadtree*

```
type Point = (Int, Int)
type Coords = (Point, Point)
type Section = (Coords, Colour)
data QTree = Empty | Leaf Section | Node Coords QTree QTree QTree QTree
```

A cada nó e cada secção (folha) de uma árvore estão associadas duas coordenadas, dos vértices *superior esquerdo* e *inferior direito*. Observe-se que a presença do constructor `Empty` deve-se à eventual necessidade de dividir imagens com apenas dois pixels em quadrantes; neste caso dois dos quadrantes serão vazios.

O Kit Fornecido

Os *bitmaps* serão armazenados em ficheiros com a seguinte estrutura:

- uma linha contendo um inteiro correspondente ao número r de linhas da imagem, seguida de
- uma linha contendo um inteiro correspondente ao número c de colunas da imagem, seguida de
- $r \times c$ linhas, cada uma contendo três inteiros separados por espaços. Estes três números codificam uma cor segundo o padrão RGB. Cada uma destas linhas de texto corresponde então a um pixel; os pixels aparecem no ficheiro a partir do vértice superior esquerdo da imagem, sendo depois varridas as linhas uma a uma da esquerda para a direita.

O ficheiro `visual.NET.zip` contém um visualizador de bitmaps (escrito sobre a plataforma .NET) descritos segundo este formato. Para poder ler e visualizar bitmaps, terá antes de mais que

1. Escrever uma função monádica de leitura de bitmaps para o tipo `BitMap`.
2. Escrever uma função `show` para o tipo `BitMap`, que escreva ficheiros segundo o formato descrito.

Exemplo de Divisão de uma Imagem em Quadrantes

A imagem total corresponde ao nó numerado com 1, a raiz da árvore. A sequência 2,6,10,14 representa um caminho na árvore, desde a raiz até uma folha correspondente a um pixel. No entanto a construção da árvore poderá parar mais cedo, caso se atinja uma seção em que todos os pixels tenham a mesma cor (cores não se encontram representadas na figura).

