

Imagem3D

Jorge Cardoso

21 de Novembro de 2005

Built with L^AT_EX

Conteúdo

1	Resumo	1
2	Requisitos/Configuração	1
3	Introdução	1
4	Descrição do Programa	2
5	Implementação	2
5.1	Obter os Pixeis da Imagem	2
5.2	Desenhar a Imagem	2
5.3	Rodar a Imagem	3
5.4	Listagem do Código	4
6	Adaptações	5

1 Resumo

O programa Image3D constroi uma imagem como um conjunto de linhas 3D. Cada linha corresponde a um pixel da imagem original e o comprimento da linha depende da cor desse pixel. O programa é baseado no exemplo “Zoom” [1] do Processing

2 Requisitos/Configuração

- Processing (BETA) [2]

3 Introdução

As fotografias são normalmente visualizadas como objectos a duas dimensões, uma vez que, de facto, as imagens digitais não são mais do que matrizes de pixels.

No entanto, nada nos impede de atribuir um significado diferente a um pixel da imagem. Podemos por exemplo, representar cada pixel através de uma estrutura 3D como uma esfera, um cubo, uma linha, polígono, etc.

A imagem da Figura 1 mostra uma imagem representada por linhas 3D.

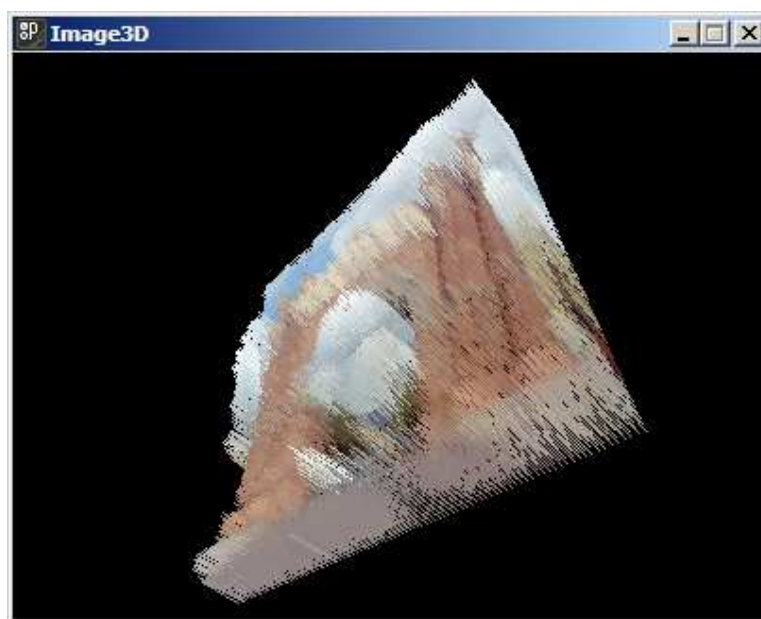


Figura 1: Imagem 3D

4 Descrição do Programa

O programa `Image3D` representa cada pixel de uma imagem através de linhas 3D, em que cada linha tem um comprimento dependente da cor do pixel correspondente. As linhas são desenhadas todas a começar no mesmo plano e perpendiculares a este. O ponto final da linha é calculado de forma a que o comprimento varie consoante a cor do pixel correspondente.

Para dar a noção de uma imagem 3D, a imagem representada por linha é rodada continuamente no centro do ecrã.

5 Implementação

5.1 Obter os Pixeis da Imagem

Para podermos representar a imagem como linhas 3D, temos primeiro de conseguir “ler” cada pixel da imagem. O tipo `PImage` do Processing fornece-nos uma forma de conseguir isso facilmente: através do campo `PImage.pixels[]`. Este campo é um vector de inteiros em que cada elemento representa um pixel da imagem. Os pixeis da imagem são organizados da esquerda para a direita e de cima para baixo neste vector.

Se quisermos saber qual o índice do vector que contém o pixel da coluna x e da linha y da imagem basta-nos utilizar a expressão

$$i = y * largura + x \tag{1}$$

em que `largura` é a largura da imagem em pixeis (a largura de uma imagem pode ser obtida com o método `PImage.width`).

Assim, para percorrer toda a imagem e ler o valor de cada pixel podemos fazer:

```
PImage img = loadImage("imagem.jpg");

for(int i=0; i < img.height; i = i + 1) {
    for(int j=0; j < img.width; j = j + 1) {
        int c = img.pixels[i*img.width+j];
    }
}
```

5.2 Desenhar a Imagem

Uma forma de converter a cor da imagem para um comprimento é somar os valores dos três componentes de cor de um pixel: vermelho, verde e azul.

Cada um dos componentes pode ser obtido através dos métodos: `red(cor)`, `green(cor)` e `blue(cor)`. Uma vez que o valor máximo da soma dos três componentes é $3 \times 255 = 765$ e este valor é demasiado grande para uma linha no ecrã, podemos reduzir o valor dividindo-o por 20, por exemplo. Neste caso ficamos com valores entre 0 e 38.

O nosso código para desenhar a imagem passa então a ser:

```
float rr, gg, bb, tt;
for(int i=0; i<a.height; i+=res) {
    for(int j=0; j<a.width; j+=res) {
        int c = a.pixels[i*a.width+j];
        rr = red(c);
        gg = green(c);
        bb = blue(c);
        tt = rr+gg+bb;

        tt = tt/20;

        stroke(c);

        line(j, i, 0, j, i, tt );
    }
}
```

As linhas são desenhadas de forma perpendicular ao ecrã e começam no plano $z = 0$. A coordenada z final da linha é indicada pelo comprimento. A cor com que a linha é desenhada é igual à cor do pixel original.

Para facilitar mais à frente a rotação da imagem pelo seu centro vamos desenhá-la de forma a que o centro da imagem fique colocado sobre o ponto $(0, 0)$. Para isso temos de subtrair metade da largura e da altura às coordenadas correspondentes:

```
float rr, gg, bb, tt;
for(int i=0; i<a.height; i+=res) {
    for(int j=0; j<a.width; j+=res) {
        int c = a.pixels[i*a.width+j];
        rr = red(c);
        gg = green(c);
        bb = blue(c);
        tt = rr+gg+bb;

        tt = tt/20;

        stroke(c);

        line(j-a.width/2, i-a.height/2, 0, j-a.width/2, i-a.
            height/2, tt );
    }
}
```

5.3 Rodar a Imagem

Para rodar a imagem temos de usar as transformações geométricas 3D. Neste caso, como queremos que a imagem apareça em rotação no centro do ecrã temos de efectuar duas transformações: uma translação para o centro do ecrã seguida de uma rotação.

```

translate(width/2, height/2, 0);
rotateX(rotX);
rotateY(rotY);

```

Uma vez que pretendemos que a imagem rode continuamente temos que incrementar continuamente o valor do ângulo de rotação de cada vez que a imagem é desenhada:

```

/* update the rotation angle */
rotX = (rotX + 0.05) %(TWO_PI);
rotY = (rotY + 0.03) %(TWO_PI);

```

O incremento é diferente para cada eixo de rotação.

5.4 Listagem do Código

```

// Image3D - Based on Zoom by REAS <http://reas.com>
//
// Author: Jorge Cardoso
// http://jorgecardoso.org
//
// 2 July 2005

/* A imagem a desenhar */
PImage a;

/* Resolution of the image */
int res = 2;

/* Current rotation angle */
float rotX = 0;
float rotY = 0;

/* Origin of the image on screen */
int originX;
int originY;

void setup()
{
  size(500, 400, P3D);

  a = loadImage("arc.jpg");

  /* calculate the origin of the image on the screen */
  originX = a.width/2;
  originY = a.height/2;

  framerate(15);
}

void draw()
{
  background(0);

```

```

    /* rotate around the center of the screen, a bit to the
       back */
    translate(width/2, height/2, 0);
    rotateX(rotX);
    rotateY(rotY);

    /* draw the image as a set of 3d lines*/
    float rr, gg, bb, tt;
    for(int i=0; i<a.height; i+=res) {
        for(int j=0; j<a.width; j+=res) {

            int c = a.pixels[i*a.width+j];
            rr = red(c);
            gg = green(c);
            bb = blue(c);
            tt = rr+gg+bb;

            stroke(c);

            line(j-originX, i-originY, tt/20, j-originX, i-
                originY, -10 );
        }
    }

    /* update the rotation angle */
    rotX = (rotX + 0.05) %(TWO_PI);
    rotY = (rotY + 0.03) %(TWO_PI);
}

/* change 'resolution' */
void keyPressed() {
    if(key == '1') {
        res = 1;
    }
    else if (key == '2') {
        res = 2;
    }
    else if (key == '3') {
        res = 3;
    }
    else if (key == '4') {
        res = 4;
    }
    else if (key == '5') {
        res = 5;
    }
}
}

```

6 Adaptações

1. Retirar a rotação automática da imagem e permitir ao utilizador rodar a imagem segundo o eixo dos yy e dos xx.

Referências

- [1] “Zoom”. Exemplo do Processing. <http://processing.org/learning/examples/zoom.html>
- [2] Ben Fry and Casey Reas, “Processing (BETA)”, <http://processing.org/>