

Processing

Captura de Vídeo

Captura de Vídeo

- Duas bibliotecas
 - Video (Processing)
 - JMyron (Externa)
 - Tracking de cor
 - Detecção de movimento

Biblioteca Video

- Captura vídeo de uma *Webcam*
 - Necessitamos de usar a biblioteca video
 - `import processing.video.*;`
 - Classe `Capture`
 - Basicamente capturamos frame a frame e desenhamos a imagem no ecrã

Exemplo_1

```
import processing.video.*;

// O objecto que captura da webcam
Capture camera;

void setup() {
  size(600, 400);

  camera = new Capture(this, 320, 200, 25);
}

void draw() {
  camera.read();

  // o objecto de captura funciona como uma imagem: podemos desenha-lo directamente
  image(camera, 0, 0, mouseX, mouseY);
}
```

Exemplo_2

```
import processing.video.*;

// O objecto que captura da webcam
Capture camera;

void setup()
{
  size(640, 400);
  frameRate(10);

  camera = new Capture(this, 320, 200, 10);
}

void draw()
{
  camera.read();

  // nao queremos linhas nos rectangulos
  noStroke();

  // cada pixel é desenhado como um rectangulo de tamanho aleatorio entre 0 e 10 pixeis.
  for (int i = 0; i < 320; i++) {
    for (int j = 0; j < 200; j++) {
      fill(camera.pixels[j*320+i]);
      rect(i+random(10), j+random(10), random(10), random(10) );
    }
  }
  println(frameRate);
}
```

Biblioteca JMyron

- `import JMyron.*;`
- Classe principal
 - `JMyron`
- Construtor
 - `myron = new JMyron();`
 - `myron.start(320, 240);` //cuidado com a dimensao: usar valores standard
- Actualização da frame
 - `myron.update();`
- Obter a frame
 - `myron.image();` // devolve um array de int

Exemplo_3

```
import JMyron.*;

JMyron myron;//a camera object
PImage img;

void setup(){
  size(640, 400);

  myron = new JMyron();//make a new instance of the object

  myron.start(320, 240); //cuidado com a dimensao: usar valores standard

  myron.findGlobs(0);//disable the intelligence to speed up frame rate
}

void draw(){
  myron.update();//update the camera view

  int [] myronPixels = myron.image(); //get the normal image of the camera

  // criar uma nova PImage para colocar a frame
  img = new PImage(320, 240);

  // copiar os pixels para a PImage
  arraycopy(myronPixels, img.pixels);

  // Desenhar a imagem normalmente
  image(img, 0, 0, mouseX, mouseY);

  println(frameRate);
}

void mousePressed(){
  myron.settings();//click the window to get the settings
}

public void stop(){
  myron.stop();//stop the object
  super.stop();
}
```

30-04-2007

Jorge Cardoso

Tracking de cor

- Podemos definir uma cor a ser detectada pelo JMyron
 - R, G, B e Tolerância
- Em cada frame o JMyron devolve-nos
 - [globCenters](#)
 - [globBoxes](#)
 - [globEdgePoints](#)
 - [globQuads](#)
 - [globPixels](#)

Exemplo_4

- Ver Projecto Processing

Retina e Difference Images

- O JMyron calcula, em cada frame, as chamadas “retina” e “difference”
- Retina é uma imagem que se vai adaptando à imagem real
- Difference é a imagem diferença entre a imagem actual e a retina
- Os globs são calculados a partir da difference image

Exemplo_5

```
import JMyron.*;

JMyron m;//a camera object

PImage retina;
PImage difference;

void setup(){
  size(640, 480);
  m = new JMyron();//make a new instance of the object
  m.start(320,240);//start a capture at 320x240

  m.findGlobs(0);//disable the intelligence to speed up frame rate

  // Definir a velocidade de adaptação da retina
  m.adaptivity(10);
}

void draw(){
  m.update();//update the camera view
  retina = createImage(320, 240, RGB);
  difference = createImage(320, 240, RGB);

  retina.pixels = m.retinaImage();
  difference.pixels = m.differenceImage();

  image(retina, 0, 0);
  image(difference, 320, 0);
}

void mousePressed(){
  m.settings();//click the window to get the settings
}

public void stop(){
  m.stop();//stop the object
  super.stop();
}
```

30-04-2007

Jorge Cardoso

Exemplo_6

- Ver Projecto Processing