PROCESSING: PROCESADOR DE IMAGENES

Este código maneja las funciones gráficas, es decir, transforma diferentes informaciones a imágenes que procesa y genera los comandos que posteriormente envía a la impresora.

- Blanco y negro, dado que nuestro plotter solo puede pintar en 1 color, lo primero que hacemos es transformar la imagen entrante a blanco y negro.

Ejecutamos una función que cicla por cada pixel de nuestra imagen, compara el brillo de cada pixel con un umbral y a partir de ello lo copia en una nueva imagen o blanco o negro.

```
void blackAndWhite(float threshold)
{
  source.loadPixels();
  destination.loadPixels();

  for (int x = 0; x < source.width; x++ ) {
    for (int y = 0; y < source.height; y++ ) {
      int loc = x + y*source.width;
      // Test the brightness against the threshold
      if (brightness(source.pixels[loc]) > threshold) {
        destination.pixels[loc] = color(255); // White
      } else {
        destination.pixels[loc] = color(0);   // Black
      }
    }
  }
  destination.updatePixels();
  // Display the destination
  image(destination, (1920 - destination.width) /2, (1080 - destination.height) /2);// set image in the centre of the screen
}
```

- Morse- Binario-Notas, lee un archivo de texto y dependiendo de si hay 1s y 0s o . Y – el programa detectará de que se trata.

  Si es morse pintará un pixel por “.”, 3 si es “-”, dejará espacio y pintará el siguiente así para cada carácter del archivo de texto.

  Si se trata de binario, por programación de 7 bits, añadirá un 8 como sistema de referencia para el lector que será siempre 1, es decir pintado en negro. Cada línea contendrá por lo tanto un numero binario, y dejará 1 fila sin pintar entre número y número

```
    if (bin)// si es binario(o notas) se pone el sistema de referencia
    {
            int loc =  i*canvas.width + canvas.width - 1;
            canvas.pixels[loc] = color(0); // dot

    }
    for (int j = 0; j < lines[i/2].length(); j++ ) {

      int loc = j + i*canvas.width + off;
      if (lines[i/2].charAt(j) == '.')
      {
        canvas.pixels[loc] = color(0); // dot
        canvas.pixels[loc + 1] = color(255); // dot
        off+= 1;
      } else if (lines[i/2].charAt(j) == '-')
      {
        canvas.pixels[loc] = color(0); // line = 3 dots
        canvas.pixels[loc + 1] = color(0); // line = 3 dots
        canvas.pixels[loc + 2] = color(0); // line = 3 dots
        canvas.pixels[loc + 3] = color(255); // dot

        off+=3;
      } else if (lines[i/2].charAt(j) == '/')
      {
        canvas.pixels[loc] = color(255);
        canvas.pixels[loc + 1] = color(255);
        canvas.pixels[loc + 2] = color(255);
        canvas.pixels[loc + 3] = color(255);
        canvas.pixels[loc + 4] = color(255);
        off+=4;
      }else if (lines[i/2].charAt(j) == ' ')
      {
        canvas.pixels[loc] = color(255);
        canvas.pixels[loc + 1] = color(255);
        canvas.pixels[loc + 2] = color(255);
        off+=2;
      } else if (lines[i/2].charAt(j) == '1')
      {
        canvas.pixels[loc] = color(0);
      }
    }
  }
```

```
66
67    // rellenar los vacios de blanco
68    for (int i = 0; i< canvas.height; i++)
69    {
70      for (int j = 0; j < canvas.width; j++ )
71      {
72        int loc = j + i*canvas.width;
73        if ((canvas.pixels[loc]) != color(0))
74        {
75          canvas.pixels[loc] = color(255);
76        }
77      }
78    }
79    canvas.updatePixels();
80    //canvas.resize(800, 0);
81    image(canvas, 0, 0, displayWidth/2, (displayWidth/4) * canvas.height/canvas.width);
82    canvas.save("Output.jpg");
83  }
```

- Procesado y generación de comandos, esta parte lee las imágenes generadas por lo anterior y las traduce a lenguaje impresora, comandos (ir a Arduino: plotter)

Este código cicla por los píxeles de la imagen fila a fila, compara el color de un pixel con el siguiente si son iguales va acumulando la longitud hasta que llega a un pixel diferente o acaba la línea, enviará un comando que pinte/ viaje la longitud de la línea, esto es para evitar que dibuje punto a punto que daría peores resultados. Una vez pintada una línea vuelve al origen, avanza el papel y pinta de nuevo. En el caso de que una línea esté vacía por completo el programa lo detectará (longitud línea blanca == longitud foto) y únicamente moverá el papel. Se ejecutará este código hasta el último pixel de la imagen

```
void slicer()
{
  int[] xSteps ={}, servoSteps = {};
  int servoState = 0, layerSteps = 0;
  if(pos < totalPixels - destination.width)
  {
    pos+= destination.width;
  } else
  {
    isActive = false;
  }
  //reset linelenght each different line
  lineLenght = 1;
  layerSteps = 1;
  for (int x = 0; x < destination.width; x++)
  {
    // vemos el color para ver si hay que pintar o no
    if (color(destination.pixels[pos + x]) == color(255))// si es el blanco
    {
      servoState = 0;// estado no pintar
    }
     else// si es negro
    {
      servoState = 1;// pintar
    }

    if (x != destination.width - 1) //  salvo la ultima columna compara con el de la dcha
    {
      if (color(destination.pixels[pos + x + 1]) == color(destination.pixels[pos + x]))// mismo color
      {
        lineLenght ++;
        layerSteps ++;

        if (color(destination.pixels[pos + x]) == color(255))// si es el blanco
        {
          finalimg.pixels[pos + x] = color(0, 0, 255);
        } else
        {
          finalimg.pixels[pos + x] = color(0, 0, 0);
        }
      } else // distinto color
      {
        //enviar
        xSteps = append(xSteps, lineLenght);
        servoSteps = append(servoSteps, servoState);
        finalimg.pixels[pos + x] = color(0, 255, 0);
        lineLenght = 1;
        layerSteps++;
      }
    }
     else// ultima columna compara con el anterior
    {
      if (color(destination.pixels[pos + x]) == color(255) && color(destination.pixels[pos + x]) == color(destination.pixels[pos + x -1]))// si es el blanco
      {
        layerSteps ++;
        lineLenght ++;
        //amarillo la parte que ignoro
        ignore(pos - 1, lineLenght -1);
        //avanzo el papel
        //println("LINEA VACIA");
      } else
      {
        xSteps = append(xSteps, lineLenght);
        servoSteps = append(servoSteps, servoState);
        lineLenght ++;
        layerSteps ++;
        finalimg.pixels[pos + x] = color(0, 255, 0);
      }
    }
  }
  printDraw(xSteps, servoSteps);//imprimo los valores
  sendData(-(layerSteps -lineLenght ), -1, 0);//vuelve al origen

  finalimg.updatePixels();
  image(finalimg, (1920 - destination.width) /2, (1080 - destination.height) /2);//coloco en el centro
}
```