Proyecto Final

Manolo Iñiguez 00212562

20 diciembre 2022

Representación de texto e imágenes por medio de partículas

El proyecto consiste en una representación visual de texto e imágenes mostrado con partículas de color o blanco y negro, en general, el código funciona obteniendo el contenido almacenado de un ArrayList donde se encuentran los textos de tipo String o las imágenes de tipo PImage.

Creo un gráfico de tipo PGraphics donde posiciono el texto o imagen para posteriormente cargar los pixeles.

Texto: El código se encarga de encontrar los pixeles que se cuentan con valor diferente a nulo, lo cual significa que tiene un valor, por ende, existe algo en ese píxel en aquel gráfico.

Imagen: El código se encarga de convertir la imagen a blanco y negro para poder encontrar los pixeles que tienen el valor en RGB de 0 o 255 (negro o blanco)

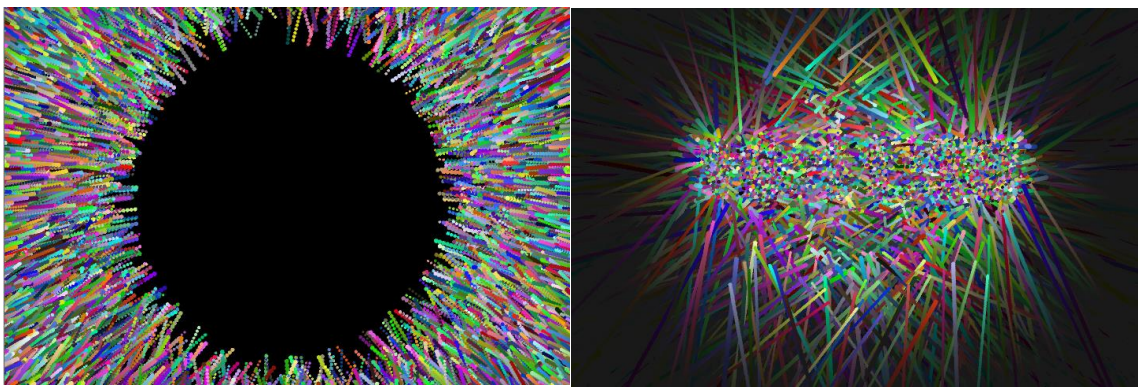
Una vez encontrado un píxel con la información solicitada, creo una partícula con la clase Particulas y le asigno un objetivo que vendría a ser la posición del pixel con la información solicitada, ya que estas partículas funcionan por medio de vectores es más sencillo trabajar con la posición, desplazamiento, velocidad y aceleración de cada partícula.

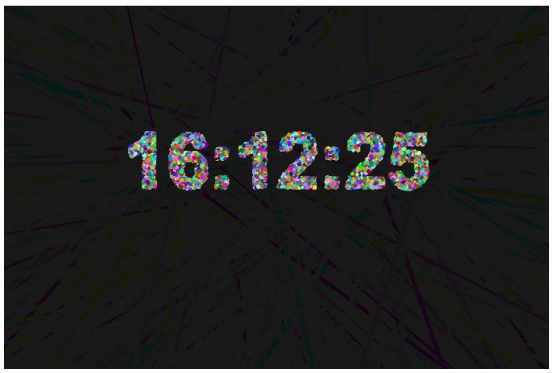
Finalmente agrego fuerza, dirección y aceleración hacia el objetivo y las partículas viajan hacia cada píxel necesario para cubrir el texto o imagen.

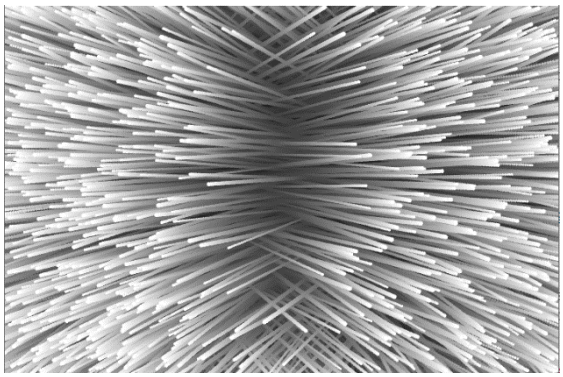
Para cambiar de texto o imagen se debe hacer click en cualquier lugar de la pantalla, esto cambia la posición del arreglo donde esta almacenado todos lo que queremos mostrar.

Ejemplos:

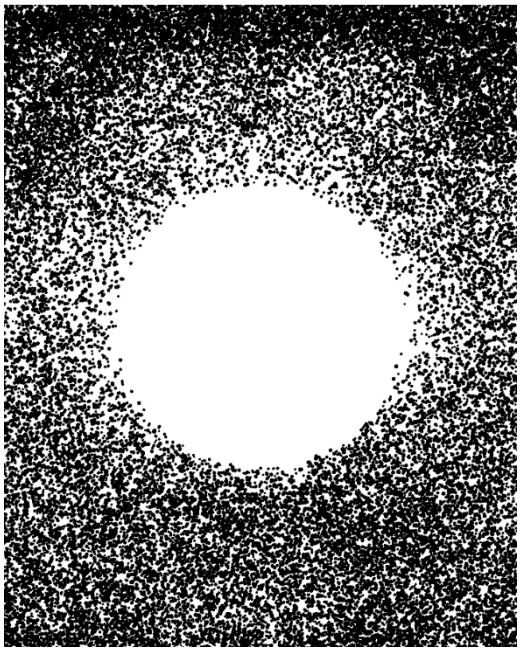
Texto →

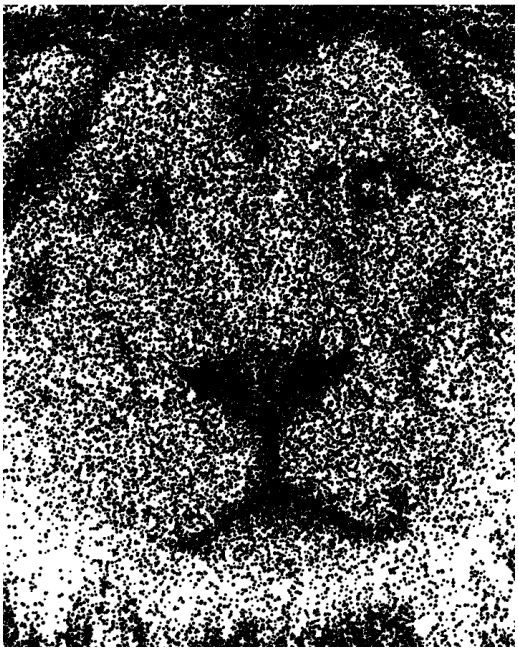


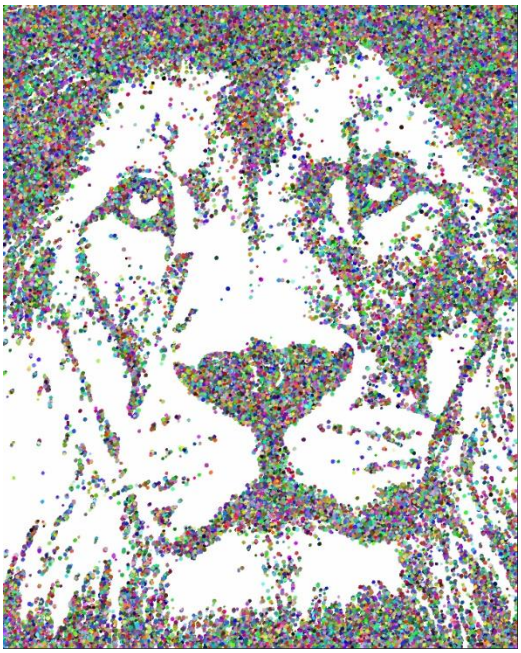




Imágenes →







Código →

Clase Partículas.

Declaración de variables.

```
class Particulas {
  PVector posicion = new PVector(0, 0);
  PVector vel = new PVector(0, 0);
  PVector acc = new PVector(0, 0);
  PVector obj = new PVector(0, 0);

  float distObjetivo = 50;
  float speed = 4.0;
  float fuerza = 0.1;
  float size = 4;
  boolean died = false;
  color colores = color(0);
```

Función para mostrar la partícula.

```
void display() {
  // Dibujo la particula
  noStroke();
  fill(colores);
  ellipse(posicion.x, posicion.y, size, size);
}
```

Función para revisar si la partículas está cerca del objetivo para reducir la velocidad a la que se desplaza, para agregar fuerza y dirección hacia el objetivo y mover la partícula.

```
void move() {
  //Revisa si la particula esta cerca del objetivo para reducir v
  float aproximidad = 1;
  float distancia = dist(posicion.x, posicion.y, obj.x, obj.y);
  if (distancia < distObjetivo) {
    aproximidad = distancia/distObjetivo;
  }

  //Agregamos fuerza hacia el objeto
  PVector toObjetivo = new PVector(obj.x, obj.y);
  toObjetivo.sub(posicion);
  toObjetivo.normalize(); //Toma cualquier vector con cualquier t
  toObjetivo.mult(speed*aproximidad);
  //toObjetivo.setMag(speed*aproximidad);

  //Agregamos direccion hacia el objeto
  PVector direccion = new PVector(toObjetivo.x, toObjetivo.y);
  direccion.sub(vel);
  direccion.normalize();
  direccion.mult(fuerza);
  //direccion.setMag(fuerza);
  acc.add(direccion);

  //Movemos la particula
  vel.add(acc);
  posicion.add(vel);
  acc.mult(0);
}
```

Función para eliminar de pantalla las partículas que no son necesarias.

```
void animacionSalida() {
  if (died == false) {
    //Cambiamos el objetivo fuera de la pantalla para que cambie de direcci
    PVector randomPos = generateRandomPos(width/2, height/2, (width+height)
    obj.x = randomPos.x;
    obj.y = randomPos.y;

    //Cambio de color a particulas muertas
    colores = color(0);
    //colores = color(random(0,255),random(0,255),random(0,255));

    died = true;
  }
}
```

Código Texto. (Solo partes más relevantes)

Declaración de variables.

```
ArrayList<Particulas> particulas = new ArrayList<Particulas>();
int pixelSteps = 1; // Piexeles entre particulas
ArrayList<String> arreglo = new ArrayList<String>(); //Guardarem
String tipoLetra = "Arial Bold";
int arregloIndex = 0;
```

Creación del grafico donde cargo el string.

```
//Creo el grafico
PGraphics grafico = createGraphics(width, height);
grafico.beginDraw();
grafico.fill(0);
grafico.textSize(100);
grafico.textAlign(CENTER);
PFont fuente = createFont(tipoLetra, 100);
grafico.textFont(fuente);
grafico.text(texto, width/2, height/2);
grafico.endDraw();
grafico.loadPixels();
```

Código Imagen. (Solo partes más relevantes)

Declaración de variables.

```
ArrayList<Particulas> particulas = new ArrayList<Particulas>();
int pixelSteps = 1; // Piexeles entre particulas
ArrayList<PImage> arreglo = new ArrayList<PImage>(); //Guardarem
int arregloIndex = 0;
PImage leon, luna, atrapaSueno, rosa, blanco, girasol;
```

Cargo las imágenes y creación del grafico donde cargo la imagen.

```
luna = loadImage("luna.jpg");
luna.filter(THRESHOLD);
leon = loadImage("leon.jpg");
leon.filter(THRESHOLD,0.2);
rosa = loadImage("rosa.jpg");
rosa.filter(THRESHOLD);
atrapaSueno = loadImage("atrapaSueno.jpg");
atrapaSueno.filter(THRESHOLD);
blanco = loadImage("blanco.jpg");
blanco.filter(THRESHOLD);


PGraphics grafico = createGraphics(width,height);
grafico.beginDraw();
grafico.imageMode(CENTER);
grafico.image(imagen, width/2, height/2);
grafico.endDraw();
grafico.loadPixels();
```

Código general.

Obtengo las partículas, muestro y muevo.

Elimino la partícula del arreglo si sale de la pantalla.

```
for (int i = 0; i < particulas.size()-1; i++) {
  //Objtengo las particulas, muestro y muevo
  Particulas particle = particulas.get(i);
  particle.move();
  particle.display();

  //Eliminio la particula del arreglo si sale de pantalla
  if (particle.died == true) {
    if (particle.posicion.x < 0 || particle.posicion.x > width ||
    particle.posicion.y < 0 || particle.posicion.y > height) {
      particulas.remove(particle);
    }
  }
}
```

Genero una posición aleatoria para asignar direcciones aleatorias a las partículas.

```
//Devuelve una posición aleatoria dentro un panel de parámetros
PVector generateRandomPos(int x, int y, float mag) {

  PVector randomDir = new PVector(random(0, width), random(0, height));
  PVector posicion = new PVector(x, y);

  posicion.sub(randomDir);
  posicion.normalize();
  posicion.mult(mag);
  posicion.add(x, y);

  return posicion;
}
```

Detecta si el píxel es de color negro y asigna sus coordenadas como valor.

```
//Solo continua si el pixel es negro
if (grafico.pixels[IndexCor] == color(0)) {
  //Convierte indice a sus coordenadas
  int x = IndexCor % width;
  int y = IndexCor / width;
```

Usa partículas que ya están el pantalla para cargar el siguiente modelo, o, crea una nueva si es necesario.

```
Particulas auxParticula;

if (IndexParticulas < conteo) {
  //Usa una particula que esta en pantalla
  auxParticula = particulas.get(IndexParticulas);
  auxParticula.died = false;
  IndexParticulas += 1;
} else {
  //Crea una nueva
  auxParticula = new Particulas();

  PVector randomPos = generateRandomPos(width/2, height/2, (width+height)/2);
  auxParticula.posicion.x = randomPos.x;
  auxParticula.posicion.y = randomPos.y;

  auxParticula.speed = random(2.0, 5.0);
  auxParticula.fuerza = auxParticula.speed*0.025;
  auxParticula.size = random(3, 6);

  particulas.add(auxParticula);
}
```

Le doy color negro u otro a cada partícula que se crea.

Y le asigno el nuevo objetivo para que se dirija la partícula.

```
  //Agregamos colores aleatorios a
  auxParticula.colores = color(0);
  //auxParticula.colores = color(ra


  //Nuevo objetivo para direccionar
  auxParticula.obj.x = x;
  auxParticula.obj.y = y;
}
```

Elimino las partículas que sobran al cambiar de modelo.

```
//Elimina las particulas que sobren al cambiar de g
if (IndexParticulas < conteo) {
  for (int i = IndexParticulas; i < conteo; i++) {
    Particulas particle = particulas.get(i);
    particle.animacionSalida();
  }
}
```

<u>Adicional:</u>

Adicional a esto, hice una versión donde se detecta el color exacto de cada pixel de una imagen a color y se asigna una partícula de ese mismo color exacto, esto es obteniendo el valor RGB de cada pixel y seteando la partícula a esa coordenada con ese mismo color.

Desafortunadamente esto es muy pesado para mostrar y el programa demora bastante, sin embargo está comprobado que funciona por los valores que devuelve de cada pixel con su información RGB.

```
for (int h = 0; h < height; h++) {
  for(int k = 0; k < width; k++){
    int loc = k + h*width;
    float r = red(grafico.pixels[loc]);
    float g = green(grafico.pixels[loc]);
    float b = blue(grafico.pixels[loc]);

    //Solo continua si el pixel es del color *****
    if (grafico.pixels[IndexCor] == color(r,g,b)){
      //Convierte indice a sus coordenadas
      int x = IndexCor % width;
      int y = IndexCor / width;

auxParticula.colores = color(r,g,b);

//Nuevo objetivo para direccionar
auxParticula.obj.x = x;
auxParticula.obj.y = y;
```