MINUS DRUM

Iván Rodríguez Ovín – UO265368 // Carlos Peláez Remis – UO258901
// Adrián Fernández Alonso – UO264268

INFORMÁTICA AUDIOVISUAL Universidad de Oviedo

ÍNDICE

1.- PROPÓSITOS DEL PROYECTO……..2

2.- INSTRUCCIONES DE USO……..2

3.- DISEÑO DEL SISTEMA Y CUESTIONES DE IMPLEMENTACIÓN……..2

4.- DISEÑO DE LOS CONTENIDOS……..3

5.- PROPUESTAS DE MEJORES Y AMPLIACIONES……..5

6.- REFERENCIAS……..5

7.- ANEXOS……..5

8.- ENLACES……..13

1. PROPÓSITOS DEL PROYECTO

Se nos encomendó la tarea de realizar un proyecto para la asignatura de Informática Audiovisual, aplicando técnicas y elementos aprendidos a lo largo del semestre. El objetivo es conseguir una batería virtual, la cual tiene 2 opciones: detección de elementos por ratón y detección de elementos por medio de OpenCV detectando la cara del puño cerrado.

2. INSTRUCCIONES DE USO

Una vez iniciemos la aplicación, podremos elegir entre dos modos para tocar la batería:

- Por defecto, estará activado el modo de detección de manos, el modo más complejo y que mayores dificultades ha planteado para su programación y ejecución correcta. Durante este modo el usuario podrá ver sus acciones a través de otra cámara en la pantalla.
- El segundo será usando el ratón, un modo sencillo de utilizar y bastante eficaz para lograr el objetivo. En este modo se desactivará la cámara.

Para cambiar de modo bastará con pulsar la tecla 'r'.

3. DISEÑO DEL SISTEMA Y CUESTIONES DE IMPLEMENTACIÓN

Será necesario tener instalado en nuestro ordenador las librerías Minim, para uso de sonido; Video, para detectar la cámara y reproducir video; y OpenCV para detección de partes del cuerpo. Además, hemos empleado una cascade haar (fichero xml producido por inteligencia artificial para detectar partes del cuerpo, en este caso, las manos), la cual deberá de estar en la siguiente ruta:
Documentos -> Processing -> libraries -> opencv_processing -> library -> cascade-files (C:/Users/USUARIO/Documents/Processing/libraries/opencv_processing/library/cascade-files/hand.xml)

El código se estructurará en 7 clases:

- Minus_Drum: clase principal del proyecto en el cual se reconocen cada uno de los elementos de la batería, se procesa el vídeo y las imágenes. Es la clase que lleva el peso de la aplicación.
- EstaDentroDeElipse: detecta que las baquetas estén en contacto con los elementos de la batería para saber si tiene que reproducir sonido o no.
- Base: elemento de la batería que ejecuta el sonido correspondiente.
- Snare (caja): elemento de la batería que ejecuta el sonido correspondiente.
- Charles: elemento de la batería que ejecuta el sonido correspondiente.
- Crash (platillo): elemento de la batería que ejecuta el sonido correspondiente.
- Tomhi: elemento de la batería que ejecuta el sonido correspondiente.
- Tommid: elemento de la batería que ejecuta el sonido correspondiente.

- Ride (platillo 2): elemento de la batería que ejecuta el sonido correspondiente.

El sistema detectará las manos (máximo de 2) como si fuesen las baquetas en el caso de la detección de manos y en el caso de detección por ratón solo habrá una baqueta.

El sonido fue recogido de grabaciones de sonido en directo y procesado para su posterior utilización en nuestra aplicación con ayuda del programa Logic Pro X.

4. DISEÑO DE LOS CONTENIDOS

De primeras, ponemos una imagen como capa principal para el vídeo, la cual representa la batería y dependiendo del modo añadimos o no la cámara del usuario.

Imagen base:

Baquetas:

En el modo por defecto se verá de la siguiente manera nuestra aplicación:

Y así se verá en el modo ratón:

En la carpeta “sonidos”, tendremos todos los recursos para reproducir audio y en la carpeta “Cascades” tendremos el fichero con extensión xml que nos ayudan en la detección de las manos, el cual habrá que guardar en la ruta expuesta con anterioridad.

5. PROPUESTAS DE MEJORAS Y AMPLIACIONES

A la hora de implementar el uso del ratón fue realmente sencillo, pero no era el objetivo principal, por lo que tal y como está lo consideramos bien hecho.

El problema nos surgió al intentar implementar la detección de manos para tocar la batería, pues OpenCV, sobre todo usando cascade haar que no tengan que ver con rasgos faciales, no es del todo fiable. De esta manera, nuestra detección no es del todo correcta y requiere de una buena iluminación. En su momento, dudamos de utilizar Kinnect, pero puesto que no terminábamos de decidirnos y no sabíamos cómo encararlo, decidimos usar Processing y OpenCV, de los cuales fuimos adquiriendo experiencia a lo largo del semestre. Una vez mejorada la detección, podríamos ampliar el proyecto con más instrumentos, para dotar de variedad a la aplicación.

A lo largo del proyecto, intentamos capturar vídeo, de tal manera que el usuario pudiese guardar sus “obras” generadas con nuestra batería. Es un concepto que podría ser considerado a la hora de tener que realizar alguna ampliación.

6. REFERENCIAS

Referencias de Processing: https://processing.org/reference/

Librerías de Processing: https://processing.org/reference/libraries/

Repositorio git público del cual obtuvimos el cascade haar (hand.xml): https://github.com/Balaje/OpenCV/tree/master/haarcascades

7. ANEXOS

Código de Minus_Drum:

```
//Clase general, detecta las manos y manda reproducir sonidos

import ddf.minim.*;
import gab.opencv.*;
import processing.video.*;
import java.awt.Rectangle;

Capture video;
OpenCV opencvMano;

static Minim minim;
```

```
PImage bateria;
PImage baqueta;

Base base;
Snare caja;
Charles charles;
Tomhi tomIzquierdo;
Tommid tomDerecho;
Ride platilloIzquierdo;
Crash platilloDerecho;

boolean raton = false;

void setup(){
  size(1280, 720);

  minim = new Minim(this);

  video = new Capture(this,width, height);
  frameRate(5);
  video.start();

  opencvMano = new OpenCV(this,width, height);
  opencvMano.loadCascade("hand.xml");

  bateria = loadImage("/img/bateria.jpeg");
  baqueta = loadImage("/img/baqueta.png");

  platilloIzquierdo = new Ride(new Elipse(new Punto(160,0),242.5,157.5));
  platilloDerecho = new Crash(new Elipse(new Punto(1280,50),340,240));
  tomIzquierdo = new Tomhi(new Elipse(new Punto(465,195),132.5,100));
  tomDerecho = new Tommid(new Elipse(new Punto(805,180),145,107.5));
  charles = new Charles(new Elipse(new Punto(125,300),200,100));
  caja = new Snare(new Elipse(new Punto(220,550),197.5,135));
  base = new Base(new Elipse(new Punto(1130,515),250,150));

  image(bateria,0,0);
}

void draw(){
 image(bateria,0,0);

 if (video.height>0 && video.width>0){
   opencvMano.loadImage(video);

   stroke(0);
   noFill();
   strokeWeight(3);
```

```
  Rectangle[] manos = opencvMano.detect();

  if(!raton){
   frameRate(5);
   video.start();
   image(video,480,510,width/4,height/4);

   for (int i = 0; i < min(2,manos.length); i++) {
    image(baqueta, manos[i].x, manos[i].y);

    Punto p = new
Punto(manos[i].x+manos[i].width/2,manos[i].y+manos[i].height/2);

    if(base.isIn(p))
     base.reproducirSonido();
    else if(caja.isIn(p))
     caja.reproducirSonido();
    else if(charles.isIn(p))
     charles.reproducirSonido();
    else if(platilloIzquierdo.isIn(p))
     platilloIzquierdo.reproducirSonido();
    else if(platilloDerecho.isIn(p))
     platilloDerecho.reproducirSonido();
    else if(tomIzquierdo.isIn(p))
     tomIzquierdo.reproducirSonido();
    else if(tomDerecho.isIn(p))
     tomDerecho.reproducirSonido();
   }
  }else if(raton){
   video.stop();
   frameRate(5);
   image(baqueta, mouseX, mouseY);

   Punto p = new Punto(mouseX, mouseY);
   if(base.isIn(p))
    base.reproducirSonido();
   else if(caja.isIn(p))
    caja.reproducirSonido();
   else if(charles.isIn(p))
    charles.reproducirSonido();
   else if(platilloIzquierdo.isIn(p))
    platilloIzquierdo.reproducirSonido();
   else if(platilloDerecho.isIn(p))
    platilloDerecho.reproducirSonido();
   else if(tomIzquierdo.isIn(p))
    tomIzquierdo.reproducirSonido();
   else if(tomDerecho.isIn(p))
```

```
      tomDerecho.reproducirSonido();
    }

  }
}

void movieEvent(Movie movie) {
  movie.read();
}

void captureEvent(Capture c) {
  c.read();
}

public static Minim getMinim(){
  return minim;
}

void keyPressed(){
  if(key == 'r'){
    if(raton)
      this.raton = false;
    else
      this.raton = true;
  }
}
```

Código de EstaDentroDeElipse:

```
class Punto{
  int x, y;

  Punto (int x, int y){
    this.x = x;
    this.y = y;
  }
}

class Elipse{
  double alto, ancho;
  Punto centro;

  Elipse(Punto centro, double ancho, double alto){
    this.centro = centro;
    this.alto = alto;
    this.ancho = ancho;
  }
}
```

```
static class EstaDentroDeElipse{

  static boolean isIn(Elipse elipse, Punto p){
    //xcentro = k, ycentro = h, ancho/2 = rx, alto/2 = ry;
   boolean dentro = false;

   int h = elipse.centro.x;
   int k = elipse.centro.y;
   int x = p.x;
   int y = p.y;

   double parte1 = Math.pow(x-h,2);
   double parte2 = Math.pow(elipse.ancho,2);

   double parte3 = Math.pow(y-k,2);
   double parte4 = Math.pow(elipse.alto,2);

   double division1 = parte1/parte2;
   double division2 = parte3/parte4;

   double resultado = division1 + division2;

   if(resultado <= 1)
     dentro = true;

   return dentro;
  }

}
```

Código de Base:

```
//Reproduce los sonidos de la base

class Base{

  Elipse area;
  AudioSample audio;

  Base(Elipse area){
   this.area = area;
   audio = Minus_Drum.getMinim().loadSample("/sonidos/BASE.mp3",1024);
  }

  boolean isIn(Punto p){
```

```
    return EstaDentroDeElipse.isIn(this.area, p);
  }

  void reproducirSonido(){
    audio.trigger();
  }
}
```

Código de Charles:

```
//Reproduce los sonidos del Charles

class Charles{

  Elipse area;
  AudioSample audio;

  Charles(Elipse area){
    this.area = area;
    audio = Minus_Drum.getMinim().loadSample("/sonidos/CHARLES.mp3",1024);
  }

   boolean isIn(Punto p){
     return EstaDentroDeElipse.isIn(this.area, p);
  }

   void reproducirSonido(){
    audio.trigger();
  }
}
```

Código de Crash:

```
//Reproduce los sonidos del ride

class Crash{

  Elipse area;
  AudioSample audio;

  Crash(Elipse area){
    this.area = area;
    audio = Minus_Drum.getMinim().loadSample("/sonidos/CRASH.mp3",1024);
  }

   boolean isIn(Punto p){
     return EstaDentroDeElipse.isIn(this.area, p);
  }
```

```
  void reproducirSonido(){
   audio.trigger();
  }
}
```

Código de Ride:

```
//Reproduce los sonidos del ride

class Ride{

 Elipse area;
 AudioSample audio;

 Ride(Elipse area){
  this.area = area;
  audio = Minus_Drum.getMinim().loadSample("/sonidos/RIDE.mp3",1024);
 }

  boolean isIn(Punto p){
   return EstaDentroDeElipse.isIn(this.area, p);
 }

  void reproducirSonido(){
   audio.trigger();
 }
}
```

Código de Snare:

```
//Reproduce los sonidos de la snare

class Snare{

 Elipse area;
 AudioSample audio;

 Snare(Elipse area){
  this.area = area;
  audio = Minus_Drum.getMinim().loadSample("/sonidos/SNARE.mp3",1024);
 }

  boolean isIn(Punto p){
   return EstaDentroDeElipse.isIn(this.area, p);
 }

  void reproducirSonido(){
```

```
   audio.trigger();
  }
}
```

Código de Tomhi:

```
//Reproduce los sonidos del Tom

class Tomhi{

 Elipse area;
 AudioSample audio;

 Tomhi(Elipse area){
   this.area = area;
   audio = Minus_Drum.getMinim().loadSample("/sonidos/TOM-HI.mp3",1024);
 }

  boolean isIn(Punto p){
    return EstaDentroDeElipse.isIn(this.area, p);
 }

  void reproducirSonido(){
   audio.trigger();
 }
}
```

Código de Tommid:

```
//Reproduce los sonidos del Tom

class Tommid{

 Elipse area;
 AudioSample audio;

 Tommid(Elipse area){
   this.area = area;
   audio = Minus_Drum.getMinim().loadSample("/sonidos/TOM-MID.mp3",1024);
 }

  boolean isIn(Punto p){
    return EstaDentroDeElipse.isIn(this.area, p);
 }

  void reproducirSonido(){
   audio.trigger();
 }
}
```

8. ENLACES

- Demo del proyecto: https://www.youtube.com/watch?v=vUSnV2DA6CY&feature=youtu.be
- Making-of del proyecto: https://www.youtube.com/watch?v=9gjL9TsLfYg