| | | | |
|---|---|---|---|
| Institución Universitaria | INFORME FINAL DE TRABAJO DE GRADO | Código | FDE 089 |
| | | Versión | 03 |
| | | Fecha | 2015-01-27 |

# CONTRUCCION DE UNA APLICACIÓN GESTUAL E INTERACTIVA DE SONIDO MEDIANTE EL USO DE KINECT


SEBASTIAN ALARCÓN OCHOA

DANIELA CARDONA HERRERA

**PROGRAMA ACADEMICO**

INGENIERIA DE SISTEMAS

**DIRECTOR DEL TRABAJO DE GRADO**

MAURICIO ARIAS CORREA

**INSTITUTO TECNOLÓGICO METROPOLITANO**


**2018**

# RESUMEN

El objetivo del presente trabajo es desarrollar un software de mezcla musical básico, que permita su interacción a través del lenguaje corporal por reconocimiento de movimiento.
Para este desarrollo se empleó el paradigma de programación orientada a objetos, se usó processing versión 3.3.6 y las librerías libres OpenKinect, Sound, sDrop, ControlP5 y Video. Durante el desarrollo se modificó contenido y funcionalidad de la librería OpenKinect, de tal manera que la imagen de profundidad creada por dicha librería sirviera para reconocer el movimiento humano.

Finalmente, el proyecto llevado a cabo permitió concluir que la interacción natural entre el usuario y la máquina, puede ser empleada en aplicaciones comerciales o de carácter investigativo. Se encontró que Processing permite manipular la información recibida del Kinect de forma fácil y efectiva para emplear en aplicaciones comerciales.

# RECONOCIMIENTOS

La realización de este proyecto de investigación fue posible gracias al laboratorio de visión artificial y fótonica ITM, por el préstamo de las instalaciones y el equipo Kinect 1414; necesario para el desarrollo de la aplicación. También se debe al profesor investigador Mauricio Arias Correa, por sus importantes asesorías, orientaciones y las respectivas correcciones realizadas durante el desarrollo del proyecto.

# ACRÓNIMOS

**PX** pixels.

**FPS** frames per second (cuadros por Segundo).

**MIDI** Musical Intrument Digital Interface (Interfaz de instrumentos musicales digital).

**OSC** Open Sound Control (control de sonido abierto).

**RGB** Red, Green blue (Rojo, verde, azul).

# TABLA DE CONTENIDO

# 1. INTRODUCCIÓN

La presente tesis de investigación, tiene por objeto facilitar la interacción entre los usuarios de los mezcladores de audio digital con una interfaz, aprovechando las ventajas de un sistema de adquisición RGBD demostrando la posibilidad de la interacción natural entre el usuario y el equipo de cómputo a través de movimientos registrados por el Kinect. Los datos se obtuvieron del estudio de las diferentes librerías de Processing para el procesamiento de imágenes de profundidad, controladores del aplicativo y lectura y reproducción de pistas de sonido. KHAXER se realizó para demostrar que es posible facilitar el manejo de un aplicativo, a través de movimientos que permiten eliminar la necesidad de interacción entre un software y el usuario, por medio del teclado y ratón. Durante la investigación se empleó inicialmente la librería SimpleOpenNI, la cual generó dificultades en el desarrollo del aplicativo, pues no permitía la correcta interpretación de información por el Kinect y estaba desactualizada para las nuevas versiones de Processing. Tras una investigación, se descubrió la librería OpenKinect que está al día con las tecnologías necesarias para la implementación del software.

# 2. MARCO TEÓRICO

## 2.1. Definición de la interfaz natural de usuario (Interacción Natural)

Es una interfaz de usuario invisible para el mismo, que permanece de esta manera a medida que el usuario aprende a realizar interacciones cada vez más complejas, se emplea el término natural en el sentido que la mayoría de interfaces usan dispositivos de control que deben ser aprendidos a manejar (Teclado, ratón, lápiz táctil, entre otros).

Una de las principales características de la interfaz natural de usuario, es la rápida curva de aprendizaje de su uso, gracias a la interacción generada a través de movimientos gestuales, ya sean de carácter general o específicos.

## 2.2. Que es Processing

Es un lenguaje de programación e IDE de código abierto basado en Java, que permite producción de proyectos principalmente multimedia e interactivos de diseño.

## 2.3. Que es el Kinect

Es un dispositivo que inicialmente se pensó como un controlador de video juegos, para jugar sin la necesidad de mando y en su lugar, hacerlo a través de movimientos y gestos, los cuales son reconocidos gracias a los componentes que lo integran, como son el proyector de infrarrojos, la cámara RGB y el sensor de infrarrojos que le permite capturar, reconocer y posicionar el esqueleto humano en un plano.

## 2.4. Librerías empleadas

Se usaron las siguientes librerías:

**OpenKinect:** Librería libre para Processing para usar el sensor del Kinect. Usa drivers fuentes de los proyectos libfreenect y libfreenect2

**Sound:** Librería libre para Processing, que provee una manera sencilla de trabajar con audio. Puede reproducir, analizar y sinterizar sonidos.

**sDrop:** Librería libre para Processing, que permite arrastrar y soltar objetos como imágenes, archivos, texto, entre otros, al programa en Processing.

**ControlP5:** Librería libre para Processing, para construir interfaces de usuario que incluye que ofrece controladores de interacción como slider, botones, campos de texto, entre otros.

**Video:** Librería libre para Processing, que reproduce archivos de video y captura la información de video de una cámara.

### 2.5. Imagen de profundidad

Imagen que contiene información en relación a la distancia Z de las superficies de los objetos, en una escena desde el punto de vista del dispositivo que la toma.

### 2.6. RGB

Composición del color en términos de la intensidad de los colores primarios.

### 2.7. Proyector infrarrojo y sensor infrarrojo

Es un sensor de medición de distancia que emplea un sistema de emisión/recepción de radiación lumínica en el espectro infrarrojo. El sensor de profundidad funciona a partir del sensor infrarrojo del Kinect al reconocer este tipo de luz. El sensor de infrarrojos proyecta una matriz de rayos de luz infrarroja sobre el espacio en que se ubica el Kinect, al rebotar esos rayos en los objetos son capturados por el sensor infrarrojo.

### 2.8. Repositorio

Espacio centralizado donde se almacena información digital, siendo generalmente archivos informáticos como trabajos científicos, conjuntos de datos o software, entre otros.

### 2.9. OSC

Protocolo para comunicar instrumentos de música con computadores y otros dispositivos multimedia para compartir música en tiempo real sobre una red, se desarrolló para reemplazar a Midi siendo superior en sus características.

### 2.10. Midi

Protocolo, interfaz digital y conector que permite la conexión entre varios instrumentos musicales electrónicos, ordenadores y otros dispositivos para comunicarse entre sí.

El análisis de imágenes de profundidad se ve plasmado en distintos artículos de investigación, que buscan interpretar los datos de las cámaras Kinect para distintos propósitos, tales como: mover un brazo robótico, reconocimiento de lenguaje de señas, reconocimiento de gestos, rehabilitación física, seguimiento de individuos, entre otros.
En foros de Processing y repositorios de librerías como OpenKinect para Processing, se puede observar cómo se emplea el reconocimiento de esqueleto y gestos para identificación de una persona en habitaciones, desplazar objetos en un programa, entre muchas otras funciones básicas. También permite la identificación de dedos, lados del cuerpo e incluso superposición de imágenes sobre el esqueleto para realizar animaciones con movimientos realistas.

# 3. METODOLOGÍA

**Fase 1:** Investigación de entorno de desarrollo integrado y librerías que permitan manejar la información del Kinect para el seguimiento de movimiento.

En esta fase se decidió trabajar con Processing, en los sistemas operativos Windows y Mac OS de dispositivos personales de los estudiantes. Se realizaron investigaciones sobre cómo se podría emplear el Kinect para detectar movimientos y posición de un usuario en un espacio, encontrando la librería SimpleOpenNI que permitía el crear un esqueleto a partir del cuerpo del usuario.

**Entorno de programación Processing**

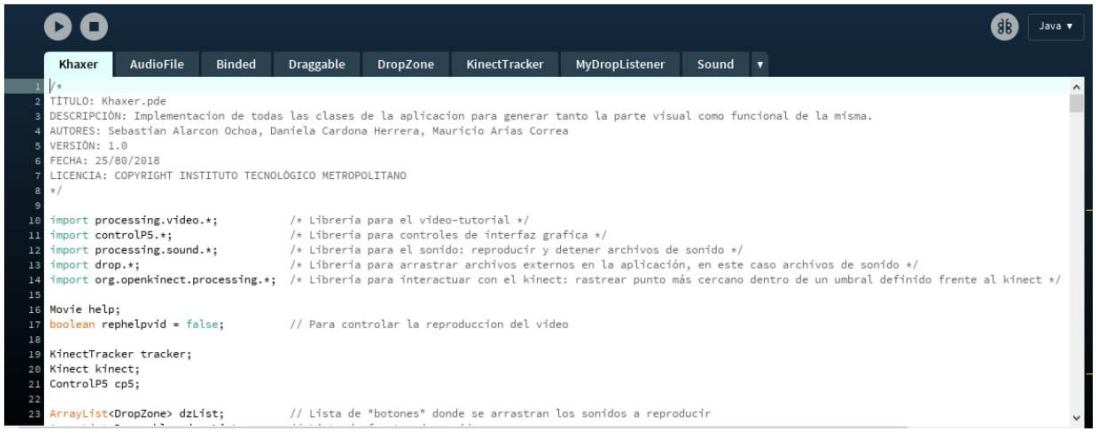


**Imagen 1.** Interfaz Processing

**Fase 2:** Diseño de interfaz básica usando librerías Sound y sDrop.

Se desarrolló una interfaz básica y se probaron las librerías de sonido y la sDrop, para arrastrar los archivos de audio a casillas dispuestas para la reproducción de sonido. Para esto se definieron las posiciones en el canvas de los componentes dispuestos para almacenar, reproducir y pausar las pistas de sonido, con el propósito de probar la funcionalidad

actual. Se descargaron pistas de audio de uso libre para probar la reproducción de sonido y el funcionamiento correcto del aplicativo.

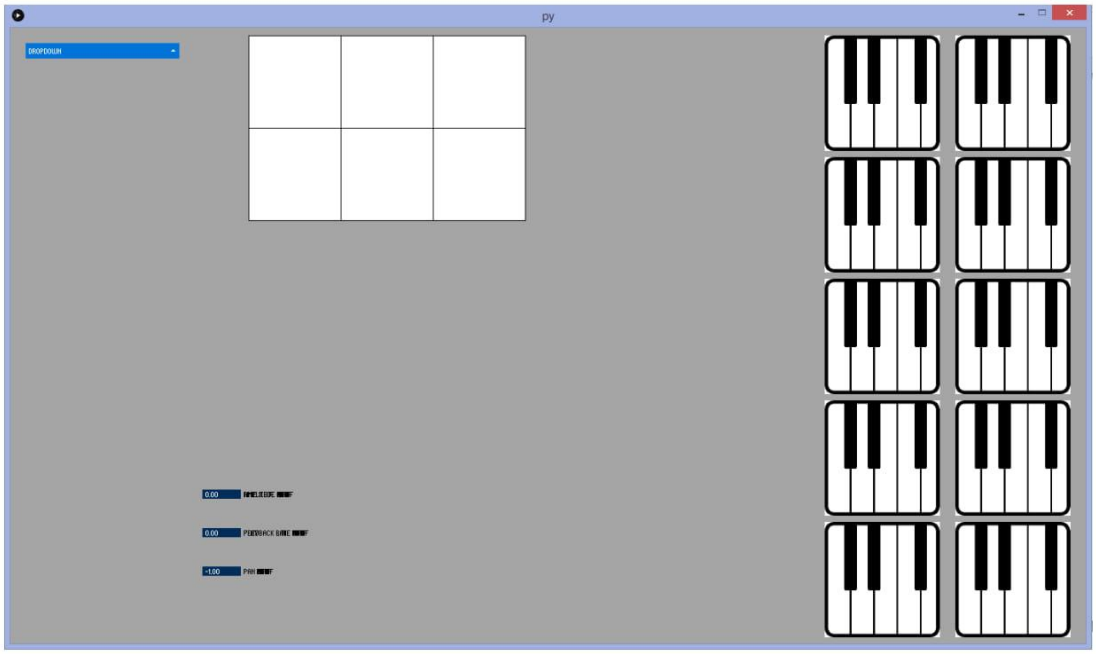


**Imagen 2.** Interfaz original

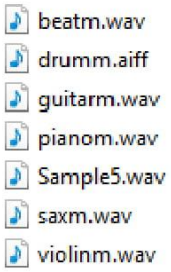


**Imagen 3.** Archivos de audio de uso libre

**Fase 3:** Implementación del Kinect en la aplicación.

Una vez funcional la reproducción de las pistas de audio se procedió a implementar el propósito del aplicativo. Originalmente se usó SimpleOpenNI pero debido a problemas de rendimiento y a que constantemente se detenía la detección del esqueleto se generó la búsqueda de librearías que permitieran programar con el Kinect y no generar problemas de perdida de usuario. Tras investigar en foros y

repositorios concernientes a Processing se encontró la librería OpenKinect. Se decidió implementar la imagen de profundidad ofrecida por dicha librería para reconocer el movimiento y posición del usuario dentro del aplicativo.

**Fase 4:** Implementación de OpenKinect.

Se procedió a reemplazar el código para trabajar con SimpleOpenNI con la nueva librería OpenKinect, se empleó la detección de profundidad de la librería para dibujar en una imagen de profundidad si el usuario es detectado, una vez logrado este propósito se encontró una dificultad: la librería solo permite desplazarse en dimensiones de la cámara Kinect, es decir en un rango de pixeles de 640x480px.

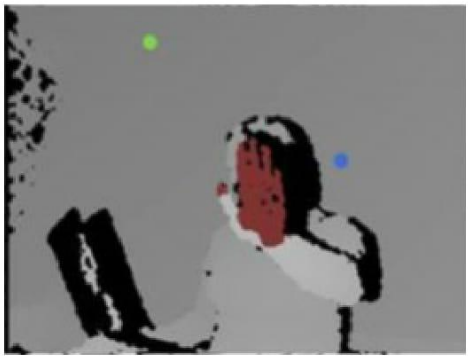

**Imagen 4.** Implementación de la librería OpenKinect

**Fase 5:** Mapeo del rango de imagen de profundidad y reconocimiento de componentes dentro de la aplicación.

Se trabajó sobre el problema encontrado en la fase 4. Se implementó la opción de mapeo de Processing que permite ampliar el rango de detección al transformar un rango de px estándar a un nuevo rango más amplio o corto. Una vez solucionado el problema de limitación de espacio, se procede a obtener la posición en tiempo real del Kinect, para que cuando el usuario se encuentre encima de un componente de reproducción o botón del aplicativo, este lo reconozca como si se hubiera dado Clic.

**Fase 6:** Diseño de la interfaz final.

En esta fase se procede a crear el diseño final de la interfaz, añadiendo botones faltantes para su funcionalidad esperada y creando, además, una experiencia de usuario más amena, a través de un fondo acorde al tema del aplicativo, usando colores suaves a la vista para que el programa sea agradable para cualquier persona que lo utilice.

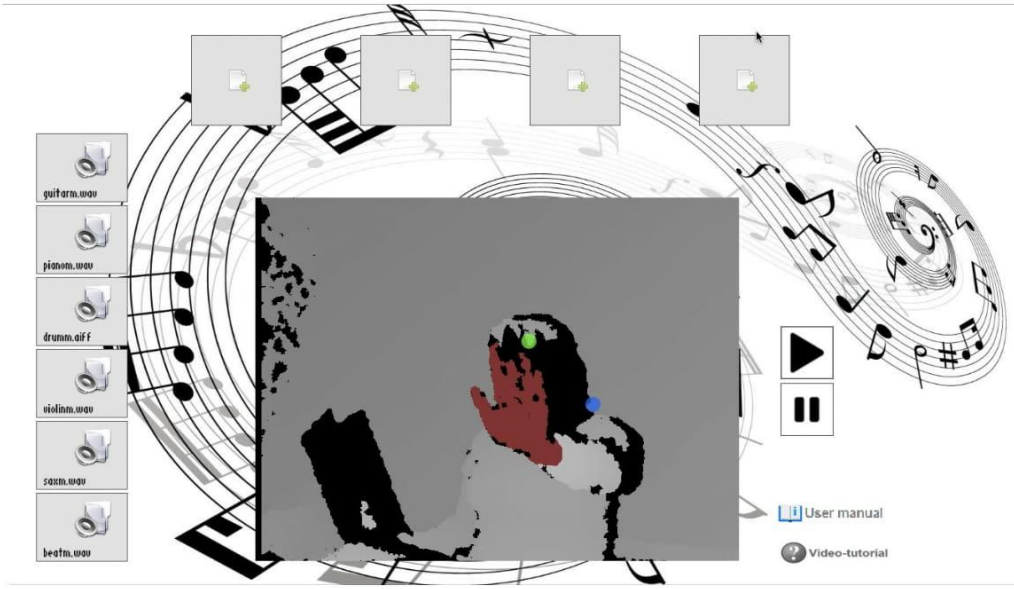


**Imagen 5.** Diseño final de la interfaz de usuario

**Fase 7:** Generación de ejecutables.

Se crean los ejecutables para Windows y MacOs, los cuales se prueban en distintos equipos y se encuentra una correcta ejecución del programa sin necesidad de instalar librerías externas.

# 4. RESULTADOS Y DISCUSIÓN

**DISCUSIÓN**

El objetivo del proyecto se fijó en desarrollar un aplicativo que permita manipular el programa, usando el movimiento de las manos como mediador en función del ratón, para así poder seleccionar y trabajar con el programa sin necesidad de utilizar los dispositivos de control de uso común, permitiendo así un mayor dinamismo en el manejo del software. Al emplear herramientas de software libre se minimiza los costos y el usar el Kinect permite identificar la posición del usuario en el aplicativo en tiempo real.

**RESULTADOS**

Se encontraron dos programas que siguen la misma línea de desarrollo: Kinectar y Synapse, en comparación con estos programas KHAXER ofrece una interactividad completa con el ratón, es decir, Kinectar permite manipular los audios con efectos y programarlos con la finalidad de reconocer movimientos del usuario para generar una melodía durante el proceso. La desventaja de este es que todo debe ser programado a mano, indicando que acción realizará cada movimiento y esto puede ser molesto para las personas que utilicen el software. KHAXER ofrece un sistema de control sencillo e intuitivo, donde el movimiento y la selección están programados desde el desarrollo y el usuario solo debe aprender a desplazar su mano dentro del programa, sin embargo, Kinectar tiene la ventaja de implementar ecualizadores y efectos de sonido no presentes en la versión actual de KHAXER. Synapse

es un software para Windows y Mac que permite usar el Kinect para controlar Aleton Live, Quartz Composer, Max/MSP/Jitter o cualquier otra aplicación que reciba eventos OSC, permitiendo así la versatilidad de control. Este comparte la limitación del software anterior, siendo solo un mediador, es decir, no posee funcionalidad individual como mezclador o reproductor, sino como una interfaz que puede ser programada para manipular las funcionalidades de un software ajeno a través del movimiento de las manos y del cuerpo.

Las limitaciones encontradas durante el desarrollo del programa, fueron la carencia de un sistema para guardar las pistas de audio creadas y la inclusión de efectos para la pista de sonido, sin embargo, al final de la primera versión se tiene un software de fácil uso, el cual puede ser manipulado por cualquier persona; desde niños hasta adultos mayores, sin mayor dificultad y permitir un primer paso al mundo musical para las personas que no tengan los estudios respectivos. Una vez se desarrollen las versiones posteriores, se tendrá una aplicación que permitiría su uso hasta a nivel profesional.

# 5. CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO

**CONCLUSIONES**

Desarrollar un mezclador de sonido que se maneje usando el Kinect para generar interactividad gestual en la interfaz del programa, generando un manejo dinámico e intuitivo de la aplicación, demostrando una manera de usar los programas sin necesidad de los dispositivos de control común como el teclado y ratón del computador es posible gracia a Processing y las librerías libres empleadas en el desarrollo del aplicativo.

- Processing es un entorno de desarrollo integrado que permite una curva de aprendizaje rápida, creación de aplicaciones con interfaces sencillas e interpretación de datos de distintas fuentes, permitiendo también crear conexión con otros programas para así poder recibir incluso señales Midi.

- El Kinect permite generar un medio de manejo de aplicativos preciso, dado que su taza de fps es de 30, permitiendo así un seguimiento en tiempo real.
- Las librerías libres para interpretación de datos del Kinect tienen mucho potencial, pues capturan en tiempo real la información y permiten un procesamiento interno y rápido sin afectar el rendimiento de la aplicación.

**RECOMENDACIONES**

Al ser la primera versión de la aplicación, no se programó el guardar las pistas de sonido de los usuarios. La imagen de profundidad central se debería cambiar por una imagen RGB para seguir la línea de diseño amigable del aplicativo. También se debe definir un umbral de profundidad correcto para que el usuario se ponga a la distancia idónea de la cámara.

**TRABAJO FUTURO**

Al terminar con la versión 1 de KHAXER se encuentra la oportunidad de generar mayores cambios a futuro dentro del aplicativo, para así poder entrar en el mercado como una aplicación competitiva. Se deberá trabajar la parte de guardado de las mezclas realizadas por los usuarios y agregar distintos efectos para no solo reproducir sonidos y generar una melodía a través de la reproducción y pausa de los mismos, sino, generar cambios internos en la pista musical por medio de efectos de audio.

# REFERENCIAS


St. Jean, J. 2012. Kinect Hacks. O'REILLY. https://www.safaribooksonline.com/library/view/kinect-hacks/9781449332181/ch04.html

Borenstein, G. 2012. Making Things See. O'REILLY. http://www.hmangas.com/Electronica/Datasheets/Arduino/LIBROS%20Y%20MANUALES/[Making.Things.See(2012.01)].Greg.Borenstein.pdf

Raheja, J. Chaudhary, A. Singal, K. 2011. Tracking of Fingertips and Centers of Palm Using KINECT. IEEE XPLORE. https://ieeexplore.ieee.org/abstract/document/6076365

Chang, Y, J. Chen, S, F. Huang, J, D. 2011. A Kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities. ScienceDirect. Volume 32. https://www.sciencedirect.com/science/article/pii/S0891422211002587

Patsadu, O. Nukoolkit, C. Watanapa, B. 2012. Human Gesture Recognition Using Kinect Camera. People.Cs. 28, 32. http://people.cs.pitt.edu/~chang/231/y14/gesture2.pdf

Ren, Z. Yuan, J. Meng, J. Zhang, Z. 2013. Robust Part-Based Hand Gesture Recognition Using Kinect Sensor. IEEE XPLORE. https://ieeexplore.ieee.org/abstract/document/6470686

Marin, G. Dominio, F. Zanuttigh, P. 2014. Hand gesture recognition with leap motion and kinect devices. IEEE XPLORE. https://ieeexplore.ieee.org/abstract/document/7025313

Zafrulla, Z. Brashear, H. Starner, T. Hamilton, H. Presti, P. 2011. American sign language recognition with the kinect. ACM DIGITAL LIBRARY. https://dl.acm.org/citation.cfm?id=2070532

K. K. Biswas. Basu, S, K. 2011. Gesture recognition using Microsoft Kinect. IEEE XPLORE. https://ieeexplore.ieee.org/abstract/document/6144864

Papadopoulos, G, Th. Axenopoulos, A.. Daras, P.2014. Real-Time Skeleton-Tracking-Based Human Action Recognition Using Kinect Data. Springer.Volumne 8325. 473-474 https://link.springer.com/chapter/10.1007/978-3-319-04114-8_40

Molina, A. 2011. Finger detection class. Open Processing https://www.openprocessing.org/sketch/29977#

Shiffman, D.  2017. Open Kinect for Processing. GitHub. https://github.com/shiffman/OpenKinect-for-Processing

Villicaña González, C. Orvañanos Guerrero, M,T.  Rodríguez Figueroa, E. 2018. BRAZO ROBOTICO CONTROLADO POR MEDIO DE VISION COMPUTACIONAL UTILIZANDO UN KINECT. TECNOLOGICO NACIONAL DE MEXICO. VOLUMEN 39, 128. http://www.itcelaya.edu.mx/ojs/index.php/pistas/article/view/1153

# APÉNDICE

**Apéndice A: producto de laboratorio**

**Código aplicación:**

**Khaxer.pde**

```
/*
TÌTULO: Khaxer.pde
DESCRIPCIÒN: Implementacion de todas las clases de la aplicacion para generar tanto la parte visual como funcional de la misma.
AUTORES: Sebastian Alarcon Ochoa, Daniela Cardona Herrera, Mauricio Arias Correa
VERSIÒN: 1.0
FECHA: 25/80/2018
LICENCIA: COPYRIGHT INSTITUTO TECNOLÒGICO METROPOLITANO
*/

import processing.video.*;          /* Libreria para el video-tutorial */
import controlP5.*;                 /* Libreria para controles de interfaz grafica */
import processing.sound.*;          /* Libreria para el sonido: reproducir y detener archivos de sonido */
import drop.*;                      /* Libreria para arrastrar archivos externos en la aplicación, en este caso archivos de sonido */
import org.openkinect.processing.*;  /* Libreria para interactuar con el kinect: rastrear punto más cercano dentro de un umbral definido frente al kinect */

Movie help;
```

```
boolean rephelpvid = false;          // Para controlar la reproduccion del video

KinectTracker tracker;
Kinect kinect;
ControlP5 cp5;

ArrayList<DropZone> dzList;              // Lista de "botones" donde se arrastran los sonidos a reproducir
ArrayList<Draggable> dragList;       // Lista de fuentes de sonidos
ArrayList<Binded> binds;             /* Lista de sonidos que se encuentran en la zona de reproducción, es decir, configurados para reproducir,
                                        el nombre Bind porque es la unión entre los sonidos fuentes (DragList) y los sonidos en la zona de reproducción (DropZoneList) */
int xInitial = 250;                  /* Posiciones iniciales de los botones DropZone */
int yInitial = 40;
int dzWidth = DropZone.dzWidth;      /* Tamaño de los botones DropZone */
int dzHeight = DropZone.dzHeight;

int dragWidth = 100;                 /* Tamaño de los botones DragZone */
int dragHeight = 50;

final int dragSrc = 1;                   /* Variables para saber si se esa arrastrando un elemento (pista de sonido) */
final int dragNone = 0;
int dragging = dragNone;
```

```
int dragSource = -1;                    // Para saber cual elemento (pista de sonido) se esta arrastrando

int totalDz = 4;                    /* Numero de botones DropZone */

int n = -1;                         /* PENDIENTE Para que es la n?? */
int dropZone = -1;

float xMapped;                      /* Coordenadas de rastreo del kinect */
float yMapped;

PImage img_splash;

PVector trackingVector;                 /* Vector con el que validaremos las posiciones en todo momento del rastreo del kinect */
boolean btnAll = false;                 // Para manejar funcionalidad botones Play all y Stop all
boolean isPlaying = false;              /* Para manejar funcionalidad botones Play all y Stop all: si esta sonando alguna pista se debe detener,
                                        ya sea que presionen parar todo o reproducir todo */
int xBtnPlay = 1030, xBtnStop = 1030, yBtnPlay = 425, yBtnStop = 500, sizeBtnPlay = 70, sizeBtnStop = 70;  // configuracion botones Play all y Stop all

PImage logo;                        /* Imagen del icono */
PImage bckg;                        /* Imagen del background */
PImage helpimg;                         /* Iconos de reproducir/resetear video tutorial */
PImage stopvideo;
```

```
PImage showmanual;                    /* Imagen para mostrar el manual de
ayuda*/
String manualpath;                  /* Ruta para llamar el manual*/

SDrop drop;                         /* Libreria para arrastrar archivos externos a
la aplicación */

void setup() {

  size(1440,790);                   /* Tamaño de la ventana */
  noStroke();

  logo = loadImage("images/icon.png");      /* Configuración del icono y el
background */
  logo.resize(60, 25);
  bckg = loadImage("images/background.jpg");
  bckg.resize(1440, 790);
  surface.setIcon(logo);
  helpimg = loadImage("images/help.png");
  helpimg.resize(170, 35);
  stopvideo = loadImage("images/stop_vid.png");
  stopvideo.resize(35, 40);
   img_splash = loadImage("images/splash_f.png");   /* Configuración del
splash mostrado al inicio de la ejecución de la aplicación */
  img_splash.resize(490, 600);
  showmanual = loadImage("images/help_manual.png");
  showmanual.resize(170, 35);
        manualpath = dataPath("manual") + File.separator +
"khaxer_manual.pdf"; /* definiendo ruta de manual*/
```

```
help = new Movie(this, "videos/demo.mov");

cp5 = new ControlP5(this);

dragList = new ArrayList();
dzList = new ArrayList();
binds = new ArrayList();
drop = new SDrop(this);

cp5.addButton("play video")
   .setPosition(1030, 700)
   .setSize(150,30)
   .setVisible(false)
   .setImage(helpimg)
   ;

cp5.addButton("stop video")
   .setPosition(1440, 700)
   .setSize(30,30)
   .setImage(stopvideo)
   ;

cp5.addButton("show manual")
  .setPosition(1035, 650)
  .setSize(180,30)
  .setVisible(false)
  .setImage(showmanual)
  ;
```

```
    dragList.add(new Draggable("images/sound_source.png", 55, 180,
dragWidth, dragHeight, 0, new AudioFile(this, "guitarm.wav", cp5), 0,
this));
    dragList.add(new Draggable("images/sound_source.png", 55, 275,
dragWidth, dragHeight, 0, new AudioFile(this, "pianom.wav", cp5), 1,
this));
    dragList.add(new Draggable("images/sound_source.png", 55, 370,
dragWidth, dragHeight, 0, new AudioFile(this, "drumm.aiff", cp5), 2,
this));
    dragList.add(new Draggable("images/sound_source.png", 55, 465,
dragWidth, dragHeight, 0, new AudioFile(this, "violinm.wav", cp5), 3,
this));
    dragList.add(new Draggable("images/sound_source.png", 55, 560,
dragWidth, dragHeight, 0, new AudioFile(this, "saxm.wav", cp5), 4, this));
    dragList.add(new Draggable("images/sound_source.png", 55, 655,
dragWidth, dragHeight, 0, new AudioFile(this, "beatm.wav", cp5), 5,
this));

  kinect = new Kinect(this);
  tracker = new KinectTracker();
}


void draw() {

  if (millis() < 10000) {  /* Mostrar splash por 10 segundos */
    background(255);
    image(img_splash, 475, 85);
    return;
  }
```

```
 if (dzList.size() < totalDz){  /* Pintar elementos DropZone */
   dzList.add(new DropZone(xInitial, yInitial));
   xInitial += dzWidth + 104;
 }

  background(bckg);  /* Mostrar imagen de background luego del splash
*/

       image(loadImage("images/play_all_2.png"),   xBtnPlay,   yBtnPlay,
sizeBtnPlay, sizeBtnPlay);  // Pintar botones play y stop all
       image(loadImage("images/stop_all_2.png"),   xBtnStop,   yBtnStop,
sizeBtnStop, sizeBtnStop);

 Draggable drag;

 if (n > -1){ /* PENDIENTE: Para que es la n?? */
   drag = dragList.get(n);
 }

  kinectTrack();  /* Con la libreria open kinect pintamos las dos elipses
que rastrean el movimiento */

    xMapped  =  map(trackingVector.x,  0,  640,  0,  displayWidth);  /*
trackingVector es una copia de la segunda elipse que contiene las
posiciones rastreadas por el kinect */
 yMapped = map(trackingVector.y, 0, 480, 0, displayHeight);

 // Funcionalidad botones play y stop
```

```
  if (xMapped > xBtnPlay && xMapped < xBtnPlay + sizeBtnPlay &&
yMapped > yBtnPlay && yMapped < yBtnPlay + sizeBtnPlay){ /* Si
estamos sobre los botones play All o stop All */
    btnAll = true;
    isPlaying = true;
  } else if (xMapped > xBtnStop && xMapped < xBtnStop + sizeBtnStop
&& yMapped > yBtnStop && yMapped < yBtnStop + sizeBtnStop){
    btnAll = true;
    isPlaying = false;
  }

  dropZoneHandle();
  dragHandle();

 if (dragging == dragSrc){  /* Si se esta arrastrando un elemento, pintar
en forma de la imagen del Drag en miniatura */
     image(loadImage("images/sound_source.png"),  xMapped,  yMapped,
50, 50);
 }

 bindHandle();  // Recorrer los binds para lo de sonar.....
 cp5.getController("play video").setVisible(true);
 cp5.getController("show manual").setVisible(true);

 //Para reproducir o parar el video
 if (rephelpvid==true){
     image(help, 170, 170, 850, 580);
     help.play();
  }  else if (rephelpvid == false){
     help.stop();
```

```
    cp5.getController("stop video").setPosition(1500,1000);
  }
}

void kinectTrack() {

  tracker.track();    // Run the tracking analysis
  tracker.display();  // Show the depth image

  // Let's draw the raw location
  PVector trackingVectorOriginal = tracker.getPos();
      float   xMapped1   =   map(trackingVectorOriginal.x,   0,   640,   0,
displayWidth);
      float   yMapped1   =   map(trackingVectorOriginal.y,   0,   480,   0,
displayHeight);
  fill(50, 100, 250, 200);
  noStroke();
  ellipse(xMapped1, yMapped1, 20, 20);

   // Let's draw the "lerped" location
  PVector v2 = tracker.getLerpedPos();
  float xMapped3 = map(v2.x, 0, 640, 0, displayWidth);
  float yMapped3 = map(v2.y, 0, 480, 0, displayHeight);
  fill(100, 250, 50, 200);
  noStroke();
  ellipse(xMapped3, yMapped3, 20, 20);

  // Display some info
  /*int t = tracker.getThreshold();
  fill(0);
```

```
 textSize(17);
 text("Umbral: " + t + "    " +  "Cuadros por segundo: " + int(frameRate)
+ "    " +
    "ARRIBA para aumentar umbral, ABAJO para disminuir umbral", 335,
235);*/
 trackingVector = trackingVectorOriginal;
}

void dropZoneHandle(){

 for (int i = 0; i < dzList.size(); i++){

   DropZone dz = dzList.get(i);

     if (xMapped > dz.x && xMapped < dz.x + DropZone.dzWidth &&
yMapped > dz.y && yMapped < dz.y + DropZone.dzHeight){ /* Si me
encuentro dentro de un DropZone */

     dragging = dragNone;       /* Se deja de arrastrar el elemento ya que
llegó al DropZone */
     dropZone = i;

     if (dragSource != -1){     /* Si se esta arrastrando un elemento (pista
de sonido) */
       Draggable dragCheck = dragList.get(dragSource);
       dragSource = -1;

         /* Si existe el sonido en otro DropZone, cambiar las coordenadas
para que no quede duplicado, es decir, no se puede tener una misma
pista de sonido en dos DropZone a la vez */
```

```
      dragExist(dragCheck, dz);

      Binded bind = new Binded();
      bind.dz = dz;
      bind.drag = dragCheck;

      binds.add(bind);
     }
     break;
    }
  }
}

void dragHandle(){

  for (int i = 0; i < dragList.size(); i++){

   Draggable drag = dragList.get(i);

       if (drag.isDropDownItem == 0){     /* PENDIENTE: para que
isDropDownItem */
     drag.m.draw();                /* Dibujar el Drag, con un evento de drop
incluido para que detecte cuando se arrastra un archivo externo */
   }

     if (xMapped > drag.x && xMapped < drag.x + drag.dWidth &&
yMapped > drag.y && yMapped < drag.y + drag.dHeight){ /* Si nos
encontramos sobre los botones fuentes de los sonidos */

     dragging = dragSrc;         // si se esta arrastrando un elemento
```

```
            if (drag.isDropDownItem == 1){ /* PENDIENTE: para que
isDropDownItem */
       dragSource = n;
     } else {
       dragSource = i;
     }
   }
  }
}

void bindHandle() {

  for (int i = 0; i < binds.size(); i++){

    Binded bind = binds.get(i);

   if (xMapped > bind.dz.x && xMapped < bind.dz.x + DropZone.dzWidth
&& yMapped > bind.dz.y && yMapped < bind.dz.y +
DropZone.dzHeight){

      btnAll = false;                /* Si se esta dentro del DropZone, no se
encuentra dentro de los botones play all and stop all */

       if (bind.drag.sound.sonar){  /* Toggle de reproducción del sonido
cuando esta en el DropZone */
      bind.drag.sound.sonar = false;
      bind.drag.sound.isPlaying = !bind.drag.sound.isPlaying;

      if (bind.drag.sound.isPlaying){
        bind.drag.sound.play();
```

```
      } else {
        bind.drag.sound.stop();
      }
     }
    } else {
     bind.drag.sound.sonar = true;
    }

     if (isPlaying != bind.drag.sound.isPlaying && btnAll){ // Si se esta sobre los botones reproducir todo o parar todo
      bind.drag.sound.isPlaying = isPlaying;
      btnAll = false;

      if (bind.drag.sound.isPlaying){
        bind.drag.sound.play();
      } else {
        bind.drag.sound.stop();
      }
     }

      Button dzButton = ((Button) cp5.get("" + bind.dz.x));  /* Obtener boton Control P5 del DropZone */
     fill(255);                                  /* Escribir nombre de la pista de sonido debajo del DropZone */
     noStroke();
     rect(bind.dz.x, bind.dz.y + 130, 120, 20);
     fill(0);
      text(((AudioFile)bind.drag.sound).trackName, bind.dz.x + 5, bind.dz.y + 140);
```

```
    if (bind.drag.sound.isPlaying) {                    /* Si la pista de sonido dentro del DropZone esta sonando o no, colocar las respectivas imagenes de play y stop */
      dzButton.setImage(loadImage("images/stop_dz_f_3.png"));
      dzButton.setPosition(bind.dz.x, bind.dz.y);
    } else {
      dzButton.setImage(loadImage("images/play_dz_f_3.png"));
      dzButton.setPosition(bind.dz.x, bind.dz.y);
    }
  }
}

//Para leer el video
void movieEvent(Movie m) {
  m.read();
}

void dragExist(Draggable drag, DropZone dz){
  /* Si la pista de sonido ya se encuentra en otro DropZone... eliminarla... para posteriormente reemplazarla en la linea 221 con la creación de una nuevo Binded */
  Button dzPreviousBtn;

  for (int i = 0; i < binds.size(); i++) {

    Binded bind = binds.get(i);

    String trackNameSrc = ((AudioFile) bind.drag.sound).trackName;
    String trackNameCheck = ((AudioFile) drag.sound).trackName;
```

```
    if (trackNameSrc == trackNameCheck || bind.dz.x == dz.x){  /* Se
valida la existencia de la pista de sonido en el DropZone, por el nombre
y la posicion */

     if (trackNameSrc == trackNameCheck) {                    /* Al anterior
DropZone donde estaba la pista de sonido, se pone la imagen de vacío ó
disponible */

      dzPreviousBtn = ((Button) cp5.get("" + bind.dz.x));
      dzPreviousBtn.setImage(loadImage("images/plus_green_f.png"));
     }

     bind.drag.sound.stop();                              /* Se elimina la anterior
posición de la pista de sonido, stop si estaba reproduciendo */
     bind.drag.sound.isPlaying = false;
     bind.drag = null;
     bind.dz = null;
     bind = null;
     binds.remove(i);
   }
  }
}

void dropEvent(DropEvent theEvent) {}                     /* Necesario para
el funcionamiento del MyDropListener */

public void controlEvent(ControlEvent theEvent) {

  String controllerName = theEvent.getController().getName();
```

```
 switch(controllerName){
   case "play video":
      rephelpvid=true;
      cp5.getController("stop video").setPosition(1200, 700);
      break;
   case "stop video":
      rephelpvid=false;
      break;
   case "show manual":
      launch(manualpath);
      break;
   default:
      //System.out.println("Unknown controller");
  }
}

/*void keyPressed() {                  // Si se desea tener la posibilidad del usuario modificar el threshold (umbral) del kinect, con las flechas arriba y abajo del teclado
  if (key == CODED){

    int t = tracker.getThreshold();

    switch(keyCode){
      case UP:
        t+=5;
        tracker.setThreshold(t);
      break;
      case DOWN:
        t-=5;
```

```
      tracker.setThreshold(t);
    break;
    default:
   }
  }
}*/
```

**AudioFile.pde**

```
/*
TÌTULO: Audiofile.pde
DESCRIPCIÒN: Clase donde se define la informacion de los archivos de audio, y de los metodos de reproducir y parar los sonidos.
AUTORES: Sebastian Alarcon Ochoa, Daniela Cardona Herrera, Mauricio Arias Correa
VERSIÒN: 1.0
FECHA: 25/80/2018
LICENCIA: COPYRIGHT INSTITUTO TECNOLÒGICO METROPOLITANO
*/

public class AudioFile extends Sound{

 String trackName;
 SoundFile sf;
 float lastDuration;
 PApplet parent;
 ControlP5 cp5;

 AudioFile(){}

 AudioFile(PApplet parent, String trackName, ControlP5 cp5){
```

```
  this.sf = new SoundFile(parent, trackName);
  this.trackName = trackName;
  this.parent = parent;
  this.cp5 = cp5;
 }

 void play(){
   this.sf.loop();
 }

 void stop(){
  this.sf.stop();
  this.lastDuration = (millis() - this.lastDuration) / 1000;
 }
}
```

**Binded.pde**

```
/*
TÌTULO: Binded.pde
DESCRIPCIÒN: Clase donde se definen los binds.
AUTORES: Sebastian Alarcon Ochoa, Daniela Cardona Herrera, Mauricio
Arias Correa
VERSIÒN: 1.0
FECHA: 25/80/2018
LICENCIA: COPYRIGHT INSTITUTO TECNOLÒGICO METROPOLITANO
*/

class Binded {
 Draggable drag;
 DropZone dz;
```

```
 Binded(){

 }

}
```

**Draggable.pde**

```
/*
TÌTULO: Draggable.pde
DESCRIPCIÒN: Clase en que se define las variables y metodos para el
drag de los archivos de sonido.
AUTORES: Sebastian Alarcon Ochoa, Daniela Cardona Herrera, Mauricio
Arias Correa
VERSIÒN: 1.0
FECHA: 25/80/2018
LICENCIA: COPYRIGHT INSTITUTO TECNOLÒGICO METROPOLITANO
*/

public class Draggable {
 String imgSrc;
 float x;
 float y;
 float dWidth, dHeight;
 PImage img;
 Sound sound;
 int isDropDownItem;
 MyDropListener m;
 int pos;
 PApplet py;
```

```
  Draggable(String imgSrc, float x, float y, float dWidth, float dHeight, int
isDropDownItem, Sound sound, int pos, PApplet py){
    this.imgSrc = imgSrc;
    this.x = x;
    this.y = y;
    this.dWidth = dWidth;
    this.dHeight = dHeight;
    this.img = loadImage(this.imgSrc);
    this.isDropDownItem = isDropDownItem;
    this.pos = pos;
    this.sound = sound;
    this.py = py;

    this.m = new MyDropListener(this.imgSrc, this.x, this.y, this.dWidth,
this.dHeight, this.pos, this.py, this.sound);
    drop.addDropListener(this.m);
  }

}
```

**DropZone.pde**

```
/*
TÌTULO: DropZone.pde
DESCRIPCIÒN: Clase en la que se define las variables y metodos
necesarios para la zona donde se arrastraran los sonidos de la
aplicacion.
AUTORES: Sebastian Alarcon Ochoa, Daniela Cardona Herrera, Mauricio
Arias Correa
VERSIÒN: 1.0
FECHA: 25/80/2018
LICENCIA: COPYRIGHT INSTITUTO TECNOLÒGICO METROPOLITANO
```

```
*/

public class DropZone{

  final static int dzWidth = 120;
  final static int dzHeight = 120;
  float x, y;
  float cRed = 255, cGreen = 255, cBlue = 255;

  DropZone(float x, float y){
    this.x = x;
    this.y = y;

    cp5.addButton("" + x)
     .setPosition(x, y)
     .setSize(dzWidth,dzHeight)
     .setColorBackground(color(255, 255, 255))
     .setImage(loadImage("images/plus_green_f.png"))
     //.setVisible(false)
     ;

     //rect(x - 50, y - 50, dzWidth, dzHeight);
  }
}
```

**KinectTracker.pde**

```
// Daniel Shiffman
// Tracking the average location beyond a given depth threshold
// Thanks to Dan O'Sullivan
```

```
// https://github.com/shiffman/OpenKinect-for-Processing
// http://shiffman.net/p5/kinect/

class KinectTracker {

  // Depth threshold
  int threshold = 500;

  // Raw location
  PVector loc;

  // Interpolated location
  PVector lerpedLoc;

  // Depth data
  int[] depth;

  // What we'll show the user
  PImage display;

  KinectTracker() {
    // This is an awkard use of a global variable here
    // But doing it this way for simplicity
    kinect.initDepth();
    kinect.enableMirror(true);
    // Make a blank image
    display = createImage(kinect.width, kinect.height, RGB);
    // Set up the vectors
    loc = new PVector(0, 0);
    lerpedLoc = new PVector(0, 0);
```

```
}

void track() {
  // Get the raw depth as array of integers
  depth = kinect.getRawDepth();

  // Being overly cautious here
  if (depth == null) return;

  float sumX = 0;
  float sumY = 0;
  float count = 0;

  for (int x = 0; x < kinect.width; x++) {
    for (int y = 0; y < kinect.height; y++) {

      int offset =  x + y*kinect.width;
      // Grabbing the raw depth
      int rawDepth = depth[offset];

      // Testing against threshold
      if (rawDepth < threshold) {
        sumX += x;
        sumY += y;
        count++;
      }
    }
  }
  // As long as we found something
  if (count != 0) {
```

```
    loc = new PVector(sumX/count, sumY/count);
   }

   // Interpolating the location, doing it arbitrarily for now
   lerpedLoc.x = PApplet.lerp(lerpedLoc.x, loc.x, 0.3f);
   lerpedLoc.y = PApplet.lerp(lerpedLoc.y, loc.y, 0.3f);
  }

  PVector getLerpedPos() {
   return lerpedLoc;
  }

  PVector getPos() {
   return loc;
  }

  void display() {
   PImage img = kinect.getDepthImage();

   // Being overly cautious here
   if (depth == null || img == null) return;

   // Going to rewrite the depth image to show which pixels are in
threshold
   // A lot of this is redundant, but this is just for demonstration purposes
   display.loadPixels();
   for (int x = 0; x < kinect.width; x++) {
    for (int y = 0; y < kinect.height; y++) {
     //println(x, y);
     int offset = x + y * kinect.width;
```

```
      // Raw depth
      int rawDepth = depth[offset];
      int pix = x + y * display.width;
      if (rawDepth < threshold) {
        // A red color instead
        display.pixels[pix] = color(150, 50, 50);
      } else {
        display.pixels[pix] = img.pixels[offset];
      }
    }
  }
  display.updatePixels();

  // Draw the image
  image(display, 335, 255);
}

int getThreshold() {
  return threshold;
}

void setThreshold(int t) {
  threshold =  t;
}
}
```

**MyDropListener.pde**

```
/*
TÌTULO: MyDropListener.pde
DESCRIPCIÒN: Clase donde se define la parte visual de los dropzone y la implementacion de archivos de audio externos.
```

AUTORES: Sebastian Alarcon Ochoa, Daniela Cardona Herrera, Mauricio Arias Correa
VERSIÒN: 1.0
FECHA: 25/80/2018
LICENCIA: COPYRIGHT INSTITUTO TECNOLÒGICO METROPOLITANO
*/

import java.io.File; /* Librerias java para copiar los archivos externos que se arrastran a la app, a una ubicación especifica */
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.io.IOException;
import java.nio.file.StandardCopyOption;

// A custom DropListener class.
class MyDropListener extends DropListener {

  int myColor;
  String imgSrc;
  float x;
  float y;
  float dWidth, dHeight;
  MyDropListener m;
  AudioFile sound;
  int pos;
  PApplet py;

  MyDropListener(String imgSrc, float x, float y, float dWidth, float dHeight, int pos, PApplet py, Sound sound) {

```
  myColor = color(255);

  setTargetRect(x - 10, y - 10, dWidth + 20, dHeight + 40);  // set a
target rect for drop event.

  this.x = x;
  this.y = y;
  this.dWidth = dWidth;
  this.dHeight = dHeight;
  this.imgSrc = imgSrc;
  this.pos = pos;
  this.py = py;
  this.sound = (AudioFile) sound;
 }

 void draw() {

  /* Dibuja cada elemento Drag (Fuente de los sonidos) */
  fill(230);
  stroke(100);
  rect(this.x - 10, this.y - 10, this.dWidth + 20, this.dHeight + 40);

  image(loadImage(this.imgSrc), this.x + 40, this.y, 50, 50);
  textSize(15);
  fill(0);
  text(this.sound.trackName, this.x, this.y + this.dHeight + 20);
 }

 // if a dragged object enters the target area, dropEnter is called.
 void dropEnter() {
```

```
  myColor = color(255,0,0);
 }

 // if a dragged object leaves the target area, dropLeave is called.
 void dropLeave() {
  myColor = color(255);
 }

 void dropEvent(DropEvent theEvent) {

  try{
    Path oldFile = Paths.get(theEvent.filePath());                    /* Ruta
original del archivo */
    Path newFile = Paths.get(dataPath("") + "/" +
oldFile.getFileName()); /* Nueva ruta donde se va a ubicar el archivo,
que será en la carpeta data */

    Files.copy(oldFile, newFile, StandardCopyOption.REPLACE_EXISTING);
/* Si no existe el archivo lo crea, si existe lo reemplaza, para no crear
muchas archivos iguales */

    /* Agregar el nuevo archivo a la lista de elementos Drag o fuentes de
sonido */
    dragList.add(new Draggable("images/sound_source.png", this.x,
this.y, this.dWidth, this.dHeight, 0, new AudioFile(this.py,
newFile.getFileName().toString(), cp5), this.pos, this.py));

  } catch (Exception e) {
    println("it didnt work");
    e.printStackTrace();
```

```
    }
  }
}
```

**Sound.pde**

```
 /*
 TÌTULO: Sound.pde
 DESCRIPCIÒN: Clase donde se define las variables y metodos para
validar si se debe reproducir o parar los archivos de sonido.
 AUTORES: Sebastian Alarcon Ochoa, Daniela Cardona Herrera, Mauricio
Arias Correa
 VERSIÒN: 1.0
 FECHA: 25/80/2018
 LICENCIA: COPYRIGHT INSTITUTO TECNOLÒGICO METROPOLITANO
 */
 public abstract class Sound{

    boolean sonar = true;
    boolean isPlaying = false;

    public abstract void play();
    public abstract void stop();
}
```

| Institución Universitaria | INFORME FINAL DE TRABAJO DE GRADO | Código | FDE 089 |
|---|---|---|---|
| | | Versión | 03 |
| | | Fecha | 2015-01-27 |

FIRMA ESTUDIANTES _____

_____

_____

FIRMA ASESOR _____

FECHA ENTREGA: 31/08/2018

FIRMA COMITÉ TRABAJO DE GRADO DE LA FACULTAD _____

RECHAZADO___ ACEPTADO____ ACEPTADO CON MODIFICACIONES_____

ACTA NO._____

FECHA ENTREGA: _____

FIRMA CONSEJO DE FACULTAD_____

ACTA NO._____

FECHA ENTREGA: _____