INICIACIÓN EN LA CAPTURA DE MOVIMIENTO CON KINECT Y XTION

SISTEMAS ÓPTICOS DE CAPTURA DE MOVIMIENTO BASADOS EN DISPOSITIVOS DE VISIÓN COMPUTARIZADA KINECT Y XTION PARA EL DESARROLLO DE APLICACIONES INTERACTIVAS GENERATIVAS.[1]

Director: Emiliano Causa
Autores: Francisco Alvarez Lojo, Ezequiel Rivero, Ignacio Siri, Agustin Bacigalup, Daniel Loaiza, Hernán González Moreno, Carolina Ojcius. (integrantes del Laboratorio EmmeLab) [2]

RESUMEN

El arte interactivo posee un campo de desarrollo cada vez más accesible gracias a la aparición de múltiples sistemas tecnológicos que simplifican las operaciones necesarias para vincular al hombre - *espectador/ usuario* - y la máquina, dotando fácilmente a estas obras de arte de gran gestualidad y potencial expresivo.

El presente trabajo de investigación propone un acceso inicial para quienes deseen introducirse a la captura de movimiento a través de la utilización de los nuevos dispositivos de visión sintética Asus Xtion y Microsoft Kinect, orientándose al diseño y desarrollo de obras de arte interactivas de tipo generativo.

Dicha investigación, luego de introducir al lector en la problemática de la captura de movimiento, plantea en una segunda instancia una serie de limitaciones que presentaban los desarrollos iniciales en el área de las artes interactivas con captura de movimiento, a causa de particularidades propias de los dispositivos existentes en el mercado en aquellas épocas, y describe posteriormente cómo la aparición de estos nuevos dispositivos compuestos resultan particularmente útiles para la resolución de la mayoría de ellas.

Luego de la comparación de los distintos sistemas existentes para el desarrollo de este tipo de obras, se introduce a la utilización del dispositivo de Microsoft, tanto desde un abordaje tangible (considerando las necesidades y requerimientos propios del hardware) hasta una primera aproximación al desarrollo de software de control, describiendo detalladamente cada una de las secciones necesarias en el algoritmo para su utilización y operación técnica. De modo que el código pueda ser comprendido fácilmente por desarrolladores nóveles sin necesidad de una gran investigación.

Kinect
Xtion
mocap
motion capture
interactivo
detección de movimiento
captura de movimiento

Se exponen finalmente una serie de ejemplos de algoritmos desarrollados por los autores de la investigación para la fácil implementación de estas técnicas y futuros desarrollos.

El presente documento es una descripción de la primera fase del trabajo de investigación llevado a cabo en el Laboratorio EmmeLab, dependiente de la Secretaría de Ciencia y Técnica de la Facultad de Bellas Artes, de la Universidad Nacional de La Plata, durante los años 2014 y 2015.

Introducción

A partir de su historia y sus disímiles acepciones, podríamos definir de modo genérico a la captura de movimiento como el proceso por el cual un dispositivo de recepción de imágenes capta movimientos de un objeto del mundo físico real para luego transformarlos en un sistema de control o representación en un entorno sintético o virtual.

El concepto de captura de movimiento en sus acepciones más puras comenzó en la antigua Grecia, donde Aristóteles describe el *De Motu Animalium* (Sobre el movimiento de los animales), en la cual realizaba una comparación de los cuerpos de los animales con sistemas cinéticos mecánicos.

Mucho tiempo después, Leonardo da Vinci describe algunos de los mecanismos que permiten los movimientos del cuerpo humano, tales como caminar, sentarse, dar un salto, etc.

Varios años después, Giovanni Alfonso Borelli,considerado el padre de la biomecánica estudia las articulaciones del cuerpo humano.

El fotógrafo e investigador Eadweard Muybridge fue el primero en discretar el movimiento humano y animal, a través de múltiples cámaras fotográficas, con las que captó los diferentes instantes contiguos en el tiempo de un movimiento contínuo.

Más tarde, con la aparición de la técnica de rotoscopia, que consiste en reemplazar los fotogramas reales de una filmación por dibujos calcados sobre cada fotograma, se comenzaría a utilizar la captura de movimiento en personas para agilizar la producción de dibujos animados. Ejemplos de esta técnica son los filmes de animación *Blancanieves y los siete enanitos,* de Walt Disney y los múltiples filmes de *Betty Boop,* de Max Fleischer, entre muchos otros.

En los años 70, la aparición de las computadoras y distintos periféricos electrónicos introduciría la captura de movimientos para la generación de gráficos por computadora.

Es así que a partir del uso de tecnologías complejas se desarrollan tres grandes tipos de captura de movimiento, la mecánica, la electromagnética y la óptica.

Captura de movimiento mecánica

La captura de movimiento mecánica consiste en el registro del movimiento de un sujeto a partir del uso de un conjunto de objetos, tales como guantes, brazos articulados semejantes a un exoesqueleto con piezas rígidas y sensores de medición de magnitudes dispuestos en dichas articulaciones, registrando la amplitud de las rotaciones.

Los sistemas de captura de movimiento mecánicos pueden funcionar en tiempo real, según su complejidad pueden tener desde bajos costos hasta otros muy elevados, tienen la ventaja de no verse afectados por interferencias con luces o campos magnéticos, y tienen un volumen de captura de movimiento ilimitada.

Sin embargo el equipo requiere de un cierto tiempo de instalación y montaje, debe ser calibrado con regularidad y dado que no se tienen las posiciones absolutas del usuario se las debe calcular relativamente a través de los valores que arrojan las rotaciones.

Captura de movimiento electromagnética

La captura de movimiento electromagnética consiste en el registro de movimientos mediante la utilización de sensores electromagnéticos adosados al cuerpo, los cuales miden la relación espacial con un transmisor cercano.

De acuerdo a la intensidad relativa recibida por la central, estos sistemas pueden calcular la posición espacial y la orientación de cada uno de los sensores.

Los sistemas electromagnéticos de captura de movimiento obtienen posiciones absolutas dado que las rotaciones son medidas completamente, los resultados obtenidos de los rastreadores magnéticos se pueden visualizar en tiempo real utilizándolos con sistemas computarizados que tengan la capacidad de renderizar polígonos en tiempo real.

Poseen cierta limitación espacial, ya que presentan distorsión magnética proporcionalmente a las distancias entre emisores y receptores, además pueden presentar interferencia magnética y eléctrica con objetos metálicos que estén presentes en el ambiente o que emitan campos magnéticos. Al igual que los sistemas de captura de movimiento mecánicos, los usuarios tienen que cargar con cables que los conectan a un sistema computarizado, lo cual limita la libertad de movimiento.

Captura de movimiento óptica

La captura óptica de movimiento involucra desde captar la presencia de un usuario por simple contraste de luces y sombras hasta la implementación de complejos trajes compuestos por puntos reflejantes o emisores de luz visible o infrarroja. Estos son seguidos por una o varias cámaras simultáneamente, las cuales comparten la información de registro entre sí para triangular la posición tridimensionalmente de cada punto luminoso.

Los sistemas ópticos de captura de movimiento permiten libertad de movimiento, posibilitando capturar grandes volúmenes, obteniéndose datos altamente detallados.

Sus costos son variados, dependiendo de la complejidad del dispositivo, pudiendo ir desde simples cámaras web hasta complejos dispositivos de triangulación y trajes luminosos especiales.

Cuando se utilizan puntos reflectantes, estos pueden interferir con la luz, los puntos pueden ser obstruidos con el propio cuerpo, lo que provoca la pérdida ciertos datos, aunque el sistema de software se encargue de recuperarlos estimativamente.

Las rotaciones del cuerpo no son absolutas y deben ser calculadas al igual que en el sistema mecánico.

En los sistemas más complejos y precisos, las visualizaciones en tiempo real de los resultados insumen un alto consumo de recursos ya que los datos deben ser interpretados y esto lleva un proceso y tiempo considerables.

Con la utilización de cámaras convencionales de video y el análisis de imagen a cargo diversos lenguajes de programación, la información extraída de esas manchas puede ser analizada e interpretada por librerías especialmente desarrolladas, tales como *Blob detection*, y de esa manera determinar la silueta del usuario.

Estos sistemas poseen problemas de reconocimiento cuando por ejemplo dos o más usuarios se superponen en el eje de visión. En este caso la cámara sigue detectando una sola mancha de mayor tamaño y tal vez mas compleja y una vez entregada esta información ya no se podrá reconocer entre usuarios individuales.

Problemas similares ocurren cuando existe circulación de público u otros tipos de movimiento detrás del área de captura. En esos casos la cámara devolverá datos pertenecientes a una gran mancha contrastada, producto de la fusión de todos los elementos captados.

El hecho de que exista una gran brecha entre la captura de las imágenes y el sistema de análisis e interpretación, también complejiza los resultados cuando se perciben cambios de luz en el ambiente de captura, ya que como la visión computarizada funciona por diferencia de contrastes, la cámara devolverá variaciones y esos cambios serán interpretados como movimiento por el sistema.

Otros sistemas de captura de movimiento

En los últimos años se han desarrollado dispositivos más avanzados capaces de generar datos precisos mediante el seguimiento de características en la superficie, combinando diferentes tipos de sensores y análisis para reducir las limitaciones anteriormente presentadas, permitiendo aumentar el número de usuarios registrados y mejorar la capacidad de seguimiento espacial.

Kinect

En los últimos años ha salido al mercado un dispositivo compuesto que integra múltiples sensores, comercialmente llamado Kinect.

El mismo fué desarrollado por la firma PrimeSense para Microsoft, destinado para la consola de juegos Xbox 360.

Kinect, al igual que las cámaras de captura de movimiento, permite a los usuarios interactuar con el sistema sin necesidad de tener contacto físico con un mando de juegos tradicional o interfaz tangible, pero resolviendo en el mismo dispositivo el análisis de las imágenes obtenidas, reconociendo el posicionamiento del cuerpo en las tres dimensiones, asociándolo a un esqueleto digital cargado en su procesador, entregando los datos de reconocimiento de movimiento y posicionamiento ya procesados, iniciando una nueva categoría de interfaces que ha recibido el nombre de *Interfaz natural*. Así el sistema detecta e interpreta posiciones, movimientos, gestos, comandos de voz, objetos e imágenes.

El dispositivo consta de una barra horizontal de aproximadamente 25 cm sujeta a una base de apoyo con un eje articulado.

Figura 1

Sensor Microsoft Kinect

Interiormente Kinect se compone principalmente de:

- Una cámara de video RGB (De resolución 640x480 a 30fps)
- Un proyector de rayos infrarrojos
- Una cámara de visión infrarroja
- 4 Micrófonos
- Un servomotor

Figura 1

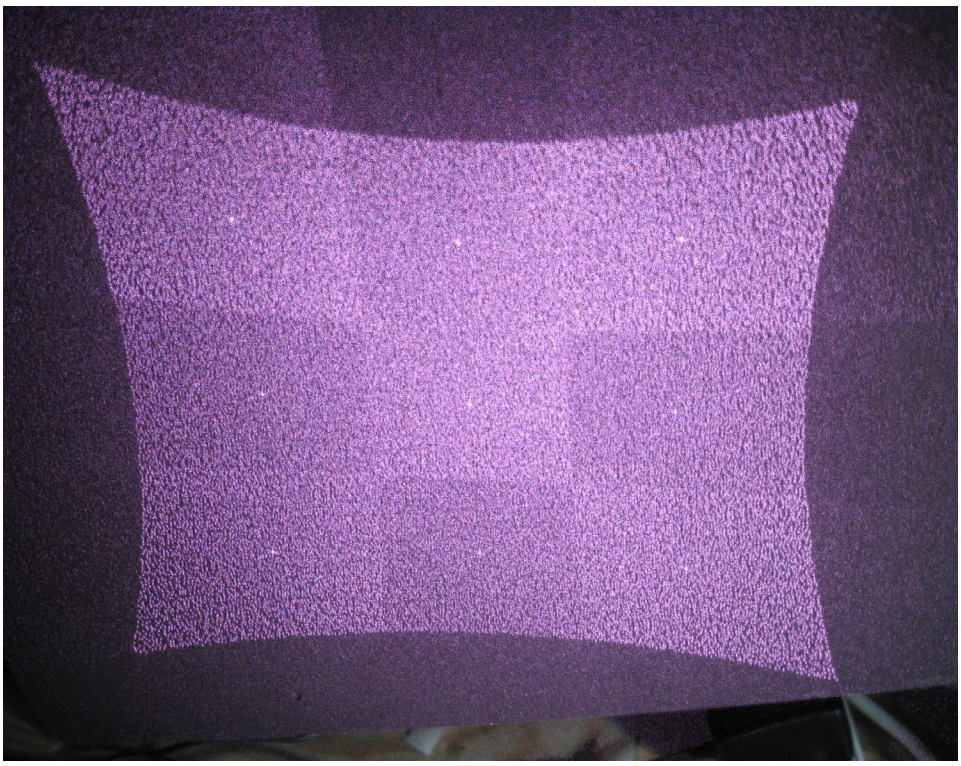
Despiece del dispositivo Kinect. [3]

FUNCIONAMIENTO DE KINECT

El dispositivo tiene un rango de visión de 0,8 a 4 metros. La cámara RGB del dispositivo detecta una primera imagen en color de la escena. Para detectar la distancia a la que se encuentra cada píxel de la imagen detectada se emite una matriz de puntos con el proyector infrarrojo que es captada con la cámara infrarroja y posteriormente calcula la deformación de dicha matriz o nube de puntos.

Figura 3

Proyección infrarroja de la matriz de puntos. [4]



Por comparación con su esqueleto digital, un modelo cinemático preprogramado, y a imágenes capturadas y analizadas en sesiones anteriores, comienza el proceso de reconocimiento de las extremidades del cuerpo del usuario interactor a una frecuencia de 30 veces por segundo.

Una vez detectadas las primeras partes del cuerpo, analiza el posicionamiento y comportamiento de las extremidades reconocidas con claridad, y determina las más dudosas por probabilidad y proximidad con las que no encierran dudas.

Ya reconstruido el cuerpo humano, y habiendo determinado cual es la combinación de extremidades más probable, dibuja su esqueleto digital tridimensional conectándolo a la silueta del usuario.

Tanto los datos de profundidad obtenidos, como los de posicionamiento de las articulaciones de distintas partes del cuerpo, llamados *Joints*, serán los que el dispositivo entregará al sistema para permitir el control remoto.

Figura 4

Detección de cuerpo humano. [5]

Xtion

El dispositivo Xtion fué desarrollado por la misma firma que Kinect, PrimeSense, esta vez para la marca Asus. Las prestaciones son las mismas que Kinect aunque existen distintas versiones, unas más orientadas al desarrollo y otras hacia la aplicación técnica.

Las versiones que lanzó ASUS son Xtion, Xtion PRO y Xtion PRO LIVE. Son similares al sensor de Microsoft y se pueden utilizar de la misma manera. Pero hay ciertas diferencias que se pueden tomar en cuenta antes de tomar la decisión sobre qué dispositivo utilizar.

El Xtion PRO junto con Kinect fueron los primeros sensores para desarrollo

en el ámbito de las nuevas interfaces naturales.

El Xtion PRO LIVE utiliza el mismo sistema de sensores infrarrojos, tecnología de detección de profundidad, de detección de imágenes en color y flujo de audio para capturar una imagen en tiempo real de los usuarios, el movimiento y la voz, haciendo el seguimiento de usuario más preciso.

La versión de desarrollo Xtion PRO LIVE viene con un conjunto de herramientas de desarrollo para que sea más fácil para los desarrolladores crear sus propias aplicaciones basadas en gestos, sin la necesidad de escribir algoritmos de programación complejos. Además, la serie Xtion PRO también es compatible con el motor de juegos de Unity3D para que el desarrollo de juegos o aplicaciones sea más fácil.

Figura 5

Sensor Asus Xtion

Figura 6

Sensor Asus Xtion Pro.

Figura 7

Sensor Asus Xtion Pro Live.

Estos dispositivos, tanto Kinect como Xtion, resuelven la mayoría de las limitaciones mencionadas en los sistemas ópticos de captura de movimiento que utilizan cámaras de video convencionales.

El reconocimiento de profundidad trae la posibilidad de limitar por software el área de captura de movimiento, de modo que no existan interferencias por movimientos o fluctuaciones periféricas.

El control mediado por la asignación de un esqueleto cinemático permite reconocer áreas específicas del cuerpo, su posicionamiento, sus movimientos y sentidos de giro, etc.

Al asociar mediante varios análisis de experiencias previas y probabilidades las extremidades de las siluetas a los esqueletos tridimensionales permite la identificación de múltiples usuarios que puedan estar interactuando.

La acción simultánea de todos los sensores, análisis de datos obtenidos y recursos de estos dispositivos hacen de los datos conseguidos de la captura de movimiento lograda de los más precisos, lo que ha posibilitado su implementación en otros entornos no lúdicos, tales como la medicina, control de sistemas robóticos industriales, etc.

Uso e implementación

En este documento nos vamos a enfocar en la implementación de Kinect sobre el sistema operativo Windows, sin embargo, cabe destacar que también es posible usarlo en Linux y Mac.

Conexiones

El sensor Kinect viene provisto de un único conector, diseñado inicialmente sólo para ser conectado a la consola Xbox.

Este dispositivo de juegos provee la alimentación eléctrica y la circulación de datos a través del mismo puerto.

Figura 8

Sensor Kinect y su conector.

Por lo que para utilizar el sensor Kinect con una computadora convencional necesitaremos utilizar un adaptador que se insertará al conector único y separará la circulación de datos de la alimentación eléctrica.

Figura 9

Conector único del Sensor Kinect.

Este adaptador está provisto de una ficha hembra, a la que se insertará el conector único de Kinect, y dos salidas macho, una USB para datos y una fuente transformadora eléctrica.

La ficha USB se conectará directamente al puerto USB de la computadora, y la fuente transformadora directamente a la red de energía eléctrica.

Figura 10

Adaptador de alimentación.

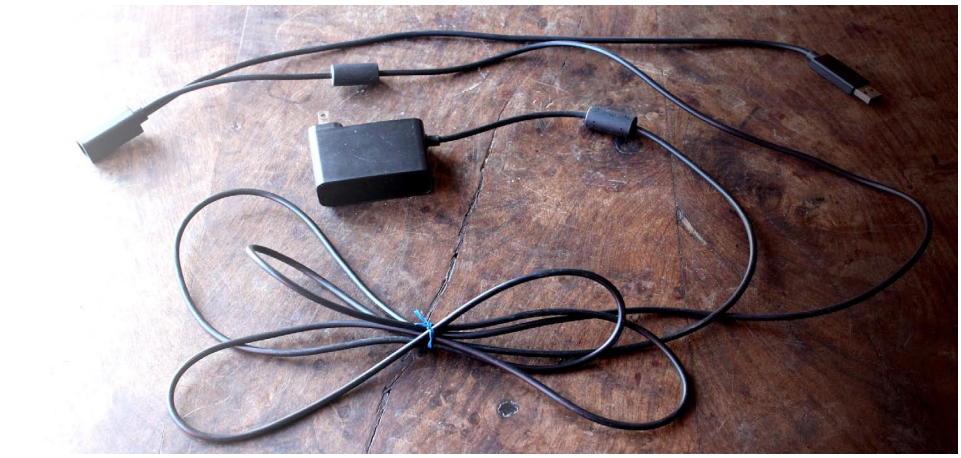


Figura 11

Ficha hembra del adaptador de alimentación junto al conector de Kinect.

De esta manera, el sensor Kinect, diseñado para ser utilizado y alimentado desde un único puerto, no existente en computadoras convencionales, podrá ser alimentado desde un tomacorriente de red eléctrica convencional, comunicándose con la computadora mediante un puerto USB.

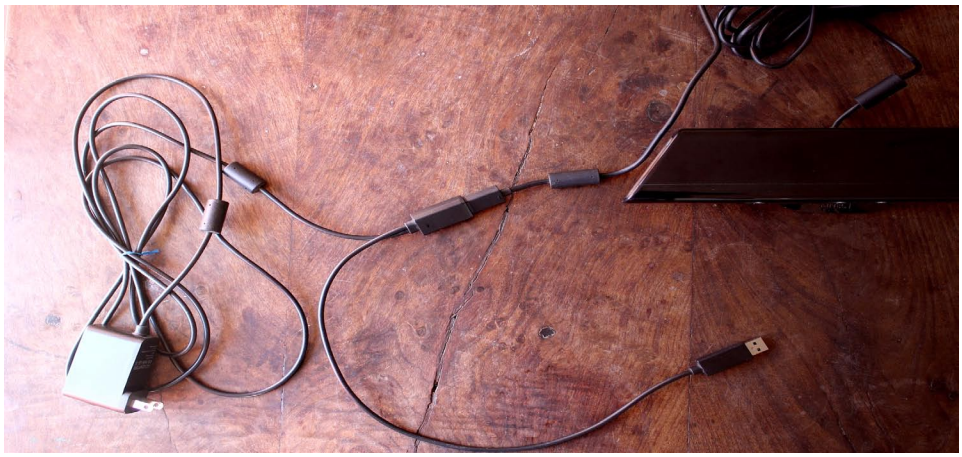


Figura 12

Conexión del adaptador con fichas individuales de alimentación y datos.

Una vez resuelta la conexión del sensor Kinect, alimentado mediante la red eléctrica de 220 volts y conectado a la computadora mediante un puerto USB convencional, ya se encuentra en condiciones de ser operado a través de alguno de los lenguajes de programación disponibles.

Figura 13

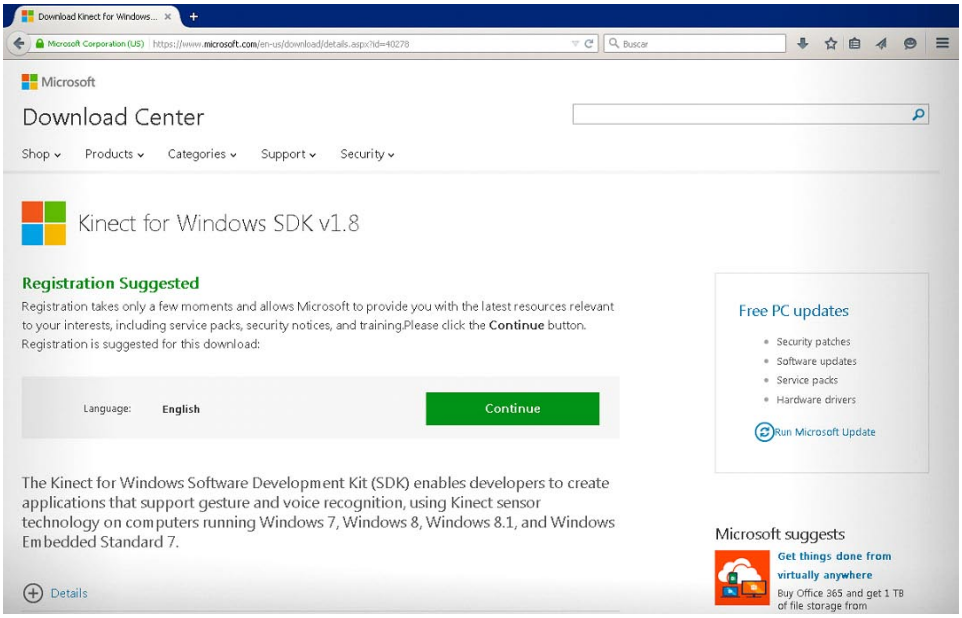
Sensor Kinect conectado vía USB a una computadora y listo para ser conectado a un tomacorriente.

Programación

Para comenzar a desarrollar aplicaciones que utilizan el sensor Kinect en nuestra computadora, inicialmente debemos instalar el SDK y las herramientas de desarrollo.

Figura 14

Sitio de descarga del SDK de Kinect para Windows



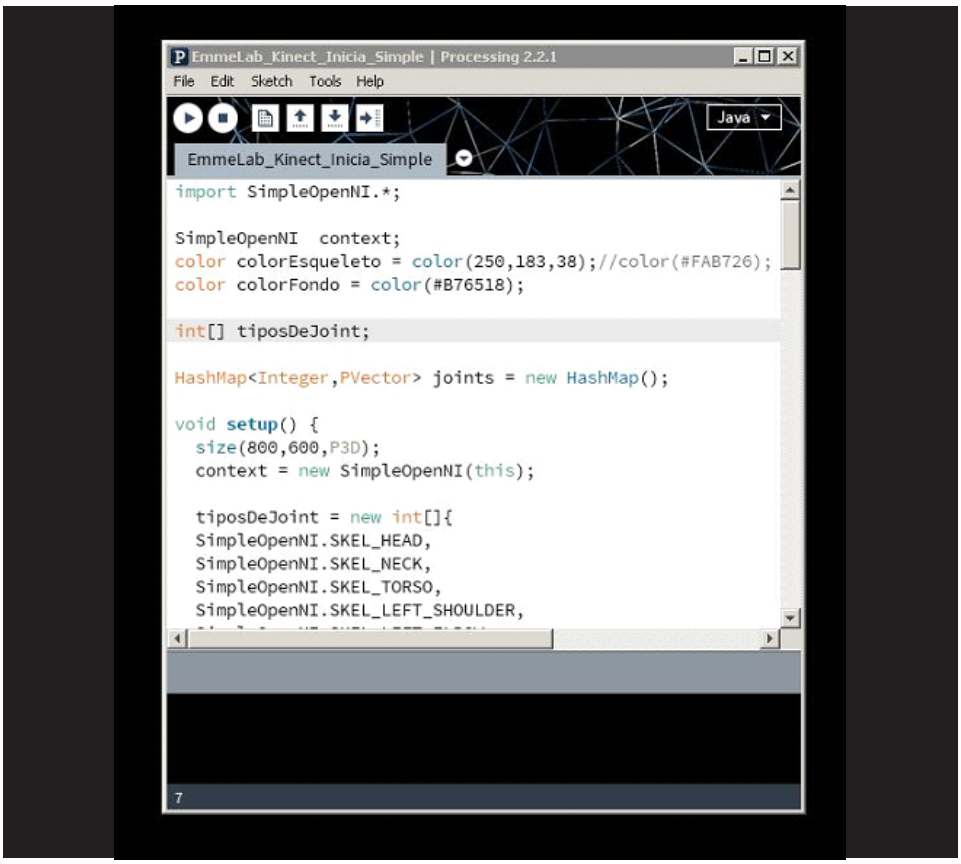
Se pueden utilizar varios lenguajes de programación o frameworks, entre ellos tenemos: Visual Basic .Net; C++; C#; Actionscript; Java; JavaScript; Lisp; Processing; Python; openFrameworks; etc.

Para este trabajo vamos a utilizar Processing[6]. Para ello también existen tres bibliotecas disntintas: Open Kinect for Processing de Daniel Shiffman; Kinect4WinSDK de Bryan Chung; y Simple-OpenNI de max Rheiner.

En este caso vamos a utilizar la librería de Rheiner.

Figura 15

IDE de Processing.



Por medio de la biblioteca OpenNI accedemos a las distintas funcionalidades de Kinect. No obstante en Processing contamos con la biblioteca Simple-OpenNI que trabaja como un recubrimiento (*wrapper*) de OpenNI.

Para instalarla solamente hace falta acceder a la IDE de Processing y de ahí dirigirse a *Sketch -> Import Library -> Add Library*.

Aparecerá una ventana que contendrá un campo para filtrar nuestra búsqueda. Una vez encontrada la biblioteca sólo hará falta darle click al botón instalar.

Figura 16

Instalando la librería SimpleOpenNI.

A continuación publicamos un código muy simple capaz de iniciar el seguimiento de los joints con Kinect.

kinect_inicia_joints_simple.pde

```
//importamos la biblioteca Simple-OpenNI

import SimpleOpenNI.*;


//creamos un instancia de SimpleOpenNI
```

```
SimpleOpenNI context;


//los joints están numerados, guardamos su numeración en un Array

int[] tiposDeJoint;

//Creamos un diccionario (HashMap) capaz de recorrer los joints más
fácilmente

//accedemos a ellos por medio de su numeración

HashMap<Integer,PVector> joints = new HashMap();


void setup() {

  size(800,600,P3D);


  //inicializamos la instancia

  context = new SimpleOpenNI(this);


  //Preguntamos si SimpleOpenNI inicio con éxito

  if(context.isInit() == false)

  {

      println("No se pudo iniciar SimpleOpenNI, quizas la camara no esta
conectada!");

     exit();

     return;

  }


  tiposDeJoint = new int[]{

    SimpleOpenNI.SKEL_HEAD,

    SimpleOpenNI.SKEL_NECK,

    SimpleOpenNI.SKEL_TORSO,

    SimpleOpenNI.SKEL_LEFT_SHOULDER,

    SimpleOpenNI.SKEL_LEFT_ELBOW,

    SimpleOpenNI.SKEL_LEFT_HAND,

    SimpleOpenNI.SKEL_RIGHT_SHOULDER,

    SimpleOpenNI.SKEL_RIGHT_ELBOW,

    SimpleOpenNI.SKEL_RIGHT_HAND,

    SimpleOpenNI.SKEL_LEFT_HIP,

    SimpleOpenNI.SKEL_LEFT_KNEE,
```

```
    SimpleOpenNI.SKEL_LEFT_FOOT,
    SimpleOpenNI.SKEL_RIGHT_HIP,
    SimpleOpenNI.SKEL_RIGHT_KNEE,
    SimpleOpenNI.SKEL_RIGHT_FOOT,
  };

  //función encargada de iniciar el HashMap
  inicializarJoints();

  //activamos el canal de captura de profundidad
  context.enableDepth();

  //activamos el canal de captura RGB
  //context.enableRGB();

  //activamos el canal de seguimiento del esqueleto de los usuarios
  context.enableUser();

  color colorEsqueleto = color(250,183,38);
  fill(colorEsqueleto);
  noStroke();
}

void inicializarJoints() {
  for (int tipoDeJoint : tiposDeJoint) {
    joints.put(tipoDeJoint, new PVector());
  }
}

void draw() {
 //ignora los valores de profundidad en el dibujado
  hint(DISABLE_DEPTH_TEST);
  background(0);

  //actualizamos la captura de la Kinect de frame a frame
```

```
  context.update();

//imprime la imagen que devuelve Kinect
  image(context.userImage(),0,0);

//guardamos la posición de nuestro punto de vista
  pushMatrix();

// trasladamos y rotamos nuestro punto de vista
  translate(width/2, height/2, 0);
  rotateX(PI);
  translate(0,0,-1000);

  //tomamos la lista de todos los usuarios
  int[] userList = context.getUsers();

  //recorremos cada usuario
  for(int i=0;i<userList.length;i++)
  {
    //preguntamos si se está haciendo el seguimiento del esqueleto
    if(context.isTrackingSkeleton(userList[i]))
    {

      //recorremos cada joint
      for (int tipoDeJoint : tiposDeJoint) {

        //preguntamos sobre el PVector del joint actual
       PVector pos = joints.get(tipoDeJoint);

        //sobreescribo el PVector pos con la posición del joint actual del
usuario actual
       context.getJointPositionSkeleton(userList[i], tipoDeJoint , pos );


       //guardo el punto de vista  actual
```

```
        pushMatrix();

       //me traslado al nuevo valor del PVector pos y dibujo una esfera
        translate( pos.x , pos.y , pos.z );
        sphere(80);

        //restablezco el punto de vista
        popMatrix();

      }

    }
  }
//restablecemos el punto de vista guardado
  popMatrix();
}

// ---------------------------------------------------------------

// Estos son los eventos que se ejecutan cuando Kinect detecta que ha
aparecido un nuevo usuario, cuando lo ha perdido y mientras sigue viendo

void onNewUser(SimpleOpenNI curContext, int userId)
{
  println("onNewUser - userId: " + userId);
  println("\tstart tracking skeleton");

  curContext.startTrackingSkeleton(userId);
}

void onLostUser(SimpleOpenNI curContext, int userId)
{
  println("onLostUser - userId: " + userId);
}
void onVisibleUser(SimpleOpenNI curContext, int userId)
```

```
{

  //println("onVisibleUser - userId: " + userId);

}
```

Finalmente incluímos un simple ejemplo de algoritmo que permite la fácil implementación del sensor Kinect para el desarrollo de aplicaciones interactivas generativas.

algoritmo_generativo.pde

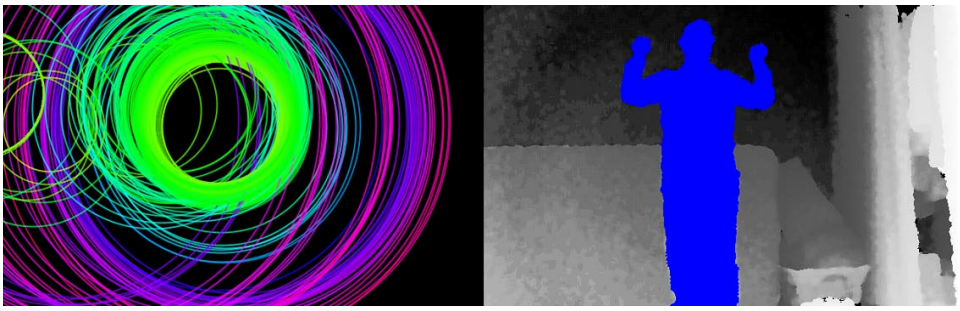
```
import SimpleOpenNI.*;

SimpleOpenNI kinect;

void setup(){
  size(1280, 400);
  background(0);
  colorMode(HSB, 255);

  // iniciamos la Kinect

  kinect = new SimpleOpenNI(this);

  if (kinect.isInit() == false) {
    println("La Kinect o esta conectada");
    exit();
    return;
  }

  // habilitamos la información de profundidad

  kinect.enableDepth();

  // habilitamos la información de usuario y joints

  kinect.enableUser();

}

void draw(){

  // actualizamos la información de la Kinect

  kinect.update();

  // previsualizamos imagen de la Kinect con detección de usuario

  image(kinect.userImage(), 640, 0);

  // guardamos la lista de usuarios detectados por la Kinect

  int[] userList = kinect.getUsers();

  // recorremos cada usuario detectado

  for (int i=0; i < userList.length; i++){

    // preguntamos si la Kinect esta siguiendo al usuario i
```

```
    if (kinect.isTrackingSkeleton(userList[i])) {
      PVector manoIzq = new PVector();
      PVector manoDer = new PVector();
      PVector torso = new PVector();

      // guardamos la posición de la mano izquierda, del usuario i

      kinect.getJointPositionSkeleton(userList[i], SimpleOpenNI.SKEL_LEFT_
HAND, manoIzq);

      // guardamos la posición de la mano derecha, del usuario i

      kinect.getJointPositionSkeleton(userList[i], SimpleOpenNI.SKEL_RIGHT_
HAND, manoDer);

      // guardamos la posición del torso, del usuario i

      kinect.getJointPositionSkeleton(userList[i], SimpleOpenNI.SKEL_TORSO,
torso);

      // en este caso, convertimos los vectores 3D en proyecciones al
plano 2D

      PVector manoIzq2D = new PVector();
      PVector manoDer2D = new PVector();
      PVector torso2D = new PVector();
      kinect.convertRealWorldToProjective(manoIzq, manoIzq2D);
      kinect.convertRealWorldToProjective(manoDer, manoDer2D);
      kinect.convertRealWorldToProjective(torso, torso2D);

        // distancia entre las manos

      float distancia = manoIzq2D.dist(manoDer2D);

     // intensidad de color proporcional a la distancia de dichas manos
           // considerando como máxima distancia el ancho de la pantalla

      float col = map(distancia, 0, width, 0, 255);

       // círculos con centro en el torso
           // donde el diámetro es la distancia entre las manos

      stroke(col, 255, 255, 180);
      strokeWeight(2);
      noFill();
      ellipse(torso2D.x, torso2D.y, distancia, distancia);
    }
  }

}

  // evento que ocurre cuando la Kinect interpreta la presencia de un
nuevo usuario en escena

void onNewUser(SimpleOpenNI kinect, int userId)
{
  println("onNewUser - userId: " + userId);

    // le avisamos a la Kinect que inicie el seguimiento de este nuevo
usuario

  kinect.startTrackingSkeleton(userId);
}
```

```
  // evento que ocurre cuando la Kinect interpreta que un usuario se
retira de escena

void onLostUser(SimpleOpenNI kinect, int userId)
{
  println("onLostUser - userId: " + userId);
}
```

Figura 17

Visualización del ejemplo generativo.



FUENTES WEB

- http://malenyabrego.wordpress.com/2014/01/11/preguntas-frecuentes-sobre-el-desarrollo-de-aplicaciones-para-kinect/#comment-736
- http://blog.jorgeivanmeza.com/2012/01/construccion-de-la-libreria-simple-openni-para-processing-bajo-ubuntu-de-32-bits/
- http://vibe.asus.com/
- http://www.asus.com/ar/Multimedia/Xtion_PRO/
- http://www.asus.com/ar/Multimedia/Xtion_PRO_LIVE/
- http://www.youbot-store.com/youbot-developers/software/drivers/asus-xtion-pro-live-driver
- http://arbelaezgroup.com/
- https://processing.org/
- http://www.microsoft.com/en-us/kinectforwindows/default.aspx
- https://code.google.com/p/simple-openni/
- http://www.davidrokeby.com/vns.html
- http://elartedigital.wordpress.com/artistas/myron-krueger/
- http://en.wikipedia.org/wiki/Videoplace
- http://wiki.ipisoft.com/Depth_Sensors_Comparison

- http://en.wikipedia.org/wiki/PrimeSense
- http://en.wikipedia.org/wiki/Kinect
- http://www.newegg.com/Product/Product.aspx?Item=N82E16826785030
- http://www.asus.com/ar/Multimedia/Xtion_PRO/

BIBLIOGRAFÍA

Guiado gestual de un robot humanoide mediante un sensor Kinect. Sammy Pfeiffer, Barcelona. 2011.

Multimedia at Work, Wenjun Zeng, University of Missouri, 2012.

REFERENCIAS

[1] El presente documento es una descripción de la primera fase del trabajo de investigación llevado a cabo en el Laboratorio EmmeLab, durante los años 2014 y 2015.

[2] El EmmeLab es un Laboratorio de investigación y experimentación en nuevas interfaces para el arte, dependiente de la Secretaría de Ciencia y Técnica de la Facultad de Bellas Artes, UNLP.

[3] Fuente: tecnoblog.net

[3] Fuente: tecnoblog.net

[4] Fuente: futurepicture.org

[5] Fuente: neurogami.com

[6] Mas información sobre Processing en processing.org.