Artículo Científico

https://doi.org/10.52428/20758944.v10i32.732

CONTROLAR EL PUNTERO DE WINDOWS A TRAVÉS DE LA INTERFACE DE USUARIO NATURAL UTILIZANDO CAMARA KINECT

MOUSE POINTER CONTROL THROUGH NATURAL USER INTERFACE USING KINECT CAMERA

Gastón Silva Sánchez (1)

RESUMEN

Una tendencia actual y creciente es la de controlar un computador por medio de nuevas tecnologías como: pantallas táctiles, control de voz, control gestual o de movimientos.

El presente trabajo describe el desarrollo de una aplicación que permita controlar el puntero del mouse de Windows, a través de movimientos y gestos de la mano del usuario.

El sistema emplea el Sensor Kinect, desarrollada por Microsoft, y su implementación está basada sobre el Kit de Desarrollo de Software de Kinect y el Entorno de desarrollo Visual Studio 2010.

La aplicación ofrece al usuario un modo alternativo de manejar el cursor del mouse y hacer click. También permite configurar la sensibilidad y el retardo entre clicks para que su manejo sea simple y amigable.

Palabras clave: Sensor Kinect. Reconocimiento de comandos. Interfaz de usuario natural. Control.

ABSTRACT

A current trend in computer controlling is using new technologies such as touchscreens, voice control, gesture or movement control.

This paper describes the development of an application that can control the Windows mouse pointer through movements and gestures of the user's hand.

The system uses the Kinect Sensor, developed by Microsoft, and its implementation is based on the Software Development Kit for Kinect, and Visual Studio 2010.

The application gives users an alternative way to handle the mouse cursor and click. It also allows setting the responsiveness and the delay between clicks so that its use is simple and friendly.

Keywords: Kinect Sensor. Recognition command. Natural user interface. Control.

INTRODUCCIÓN

1.1. Antecedentes

El advenimiento del dispositivo Kinect de Microsoft, introdujo una revolución no sólo en el campo de los videojuegos, sino en una amplia área de aplicaciones de captura y detección de movimiento, detección de espacios en tercera dimensión, procesamiento de imágenes, procesamiento de voz y comandos hablados.
Es la capacidad para detectar movimientos un área interesante que puede ser explotada para el desarrollo de aplicaciones de control de la computadora por parte de los usuarios, realizadas simplemente con gestos y

1. Ingeniero de Sistemas Informáticos.
Docente tiempo completo Univalle Cochabamba.
gsilvas@univalle.edu

Páginas 40 a 47
Fecha de recepción: 15/08/14
Fecha de aprobación: 18/12/13

movimientos, sin necesidad de tocar o manejar algún dispositivo manual.

En la actualidad el manejo y control de la interface de Windows está restringido básicamente a dos opciones: el teclado y el ratón.

1.2. Naturaleza y Alcance

El presente trabajo pretende que a través del Kit de Desarrollo de Software de Kinect (SDK por sus siglas en inglés) se implemente una aplicación que permita interactuar con la interface de Windows por medio del control básico del puntero del mouse.

La misma trabajará con la detección del esqueleto, con la integración del control del Mouse del sistema de Windows y detección de Click Izquierdo.

No se considera la detección de doble Click, ni la detección de gestos o posturas.

1.3. Justificación

El presente proyecto pretende abrir las puertas al desarrollo de aplicaciones que utilicen la Interface de Usuario Natural, y motivar a los estudiantes de la materia de Computación Gráfica para que puedan entender y emplear esta tecnología que va más allá del control lúdico o de objetos en pantalla, debido a su aplicabilidad en otras áreas de la ciencia y la tecnología.

2. OBJETIVOS

2.1. Objetivo General:

Desarrollar una aplicación informática utilizando el dispositivo Kinect para controlar acciones de Windows a través de interface de usuario natural.

2.2. Objetivos Específicos:

- Ampliar el control de la cámara y la adquisición de imágenes.

- Desplegar la captura del esqueleto y obtener coordenadas de la mano derecha del usuario.

- Implementar el sistema de control de mouse y click izquierdo.

3. MARCO TEÓRICO

3.1. La cámara Kinect

Kinect es un dispositivo de detección de movimiento que fue originalmente desarrollado para la consola de juegos Xbox 360. Uno de los factores que lo hacen distintivos entre otros de este género es que no es un dispositivo controlado a mano, sino que detecta la posición del cuerpo, el movimiento y la voz (1).

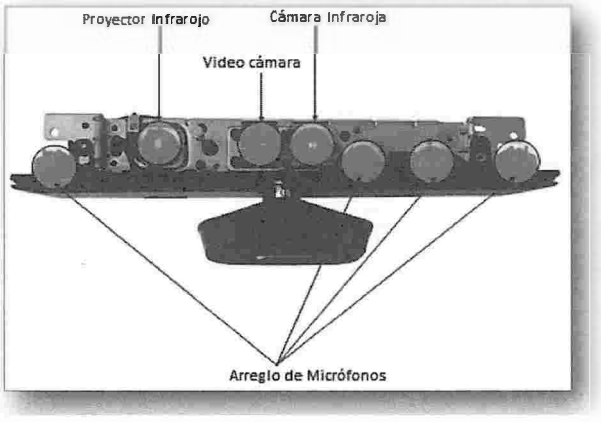
Kinect ofrece una interfaz de usuario natural (NUI) para la interacción mediante el movimiento del cuerpo y gestos, así como los comandos de voz.

3.2. Partes del Sensor

3.2.1. Sensor de profundidad:

El sensor de profundidad es un proyector de infrarrojos combinado con un sensor CMOS monocromo que permite a Kinect ver la habitación en 3D en cualquier condición de luz ambiental. El rango de detección de la profundidad del sensor es ajustable gracias al software de Kinect y capaz de calibrar automáticamente el sensor (1).

FIGURAN° 1
COMPONENTES PRINCIPALES DEL SENSOR KINECT



Fuente: (2).

3.2.2. Cámara Color RGB

Permite la captura de una escena de video a colores con características y resoluciones de 640x480 a 30 fps y una máxima de 1024 x 768 a una tasa de 15 fps (3).

3.2.3. Motor de Inclinación

La base y el cuerpo del sensor están conectados a un motor pequeño. Se utiliza para cambiar el ángulo de inclinación de la cámara para obtener la posición correcta del esqueleto humano dentro de la habitación (2).

3.2.4. Arreglo de micrófonos

Contiene cuatro micrófonos dispuestos a lo largo de la parte inferior de la barra como se muestra en la figura N° 1, dos en los extremos izquierdo y derecho, y dos más en el lado derecho de la unidad, utiliza estos micrófonos para ayudar a determinar de dónde proviene una voz en particular (1).

3.2.5. Especificaciones del sensor

TABLA 1: ESPECIFICACIONES TÉCNICAS DEL SENSOR KINECT

Especificaciones del sensor Kinect	
Ángulo de visión	43° de campo de visión vertical 57° de campo de visión horizontal
Ángulo de movimiento del motor	+28° y -28°
Velocidad de *frames*	30 *frames* por segundo (FPS)
Resolución (profundidad)	QVGA (320 x 240) 16bits
Resolución (Cámara RGB)	VGA (640 x 480) 32 bits
Formato de Audio	16-kHz, 16-bit PCM
Características de entrada de Audio	Cuatro micrófonos en array con un convertidor analógico a digital de 24 bits(ADC)
Distancia de operación	Mínima: 1.5 metros; Máxima 4 metros

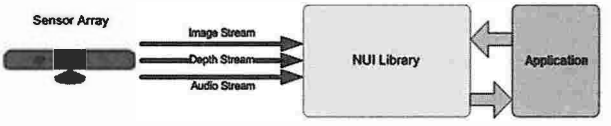
Fuente: (4).

3.3. Microsoft Kinect SDK

El SDK proporciona una biblioteca de software sofisticado y herramientas para ayudar a los desarrolladores a utilizar la rica forma de ingreso natural basado en Kinect, que detecta y reacciona a los acontecimientos del mundo real.

La figura N° 2 esquematiza la interacción de la aplicación con el sensor por medio de la Librería de interface Natural de Usuario (NUI) contenida en la API de Kinect (1).

FIGURA N° 2
DIAGRAMA DE LA INTERFACE KINECT

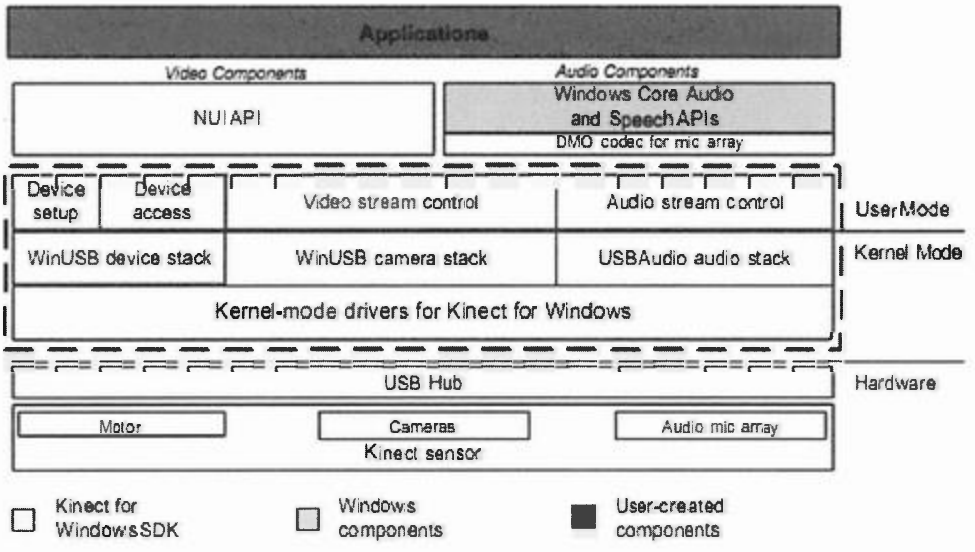


Fuente: (3).

3.4. Arquitectura del SDK

El SDK de Kinect presenta una arquitectura estructurada en capas, como se detalla en la figura N° 3. La capa de control en modo Kernel y modo User, además de la API de NUI, que permite al desarrollador utilizar los componentes de Seguimiento de esqueleto, Sensor de profundidad cámara RGB y micrófonos (2).

FIGURA N° 3
ARQUITECTURA DE SDK KINECT



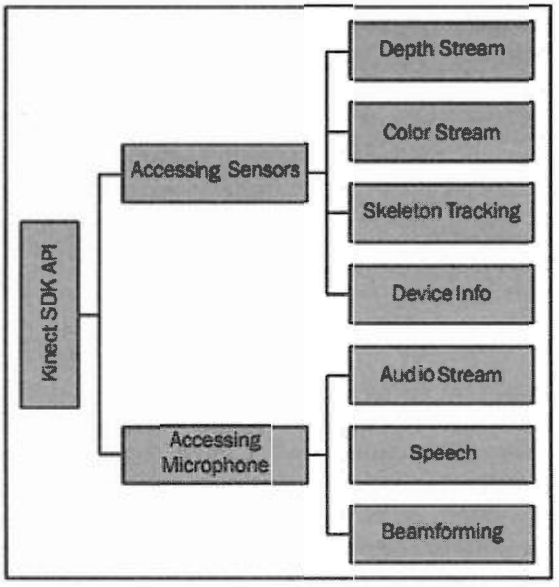
Fuente: (5).

3.5. Kinect SDK API

La figura N° 4 muestra los componentes del API de Kinect, en los que se resaltan los empleados en la aplicación, como ser el Sensor de Profundidad (DepthStream) y el Seguimiento del Esqueleto (SkeletonTracking).

3.5.1. Depth Stream: El sensor Kinect devuelve los datos de profundidad de 16-bit. Cada uno de los pixeles de datos representa la distancia entre el objeto y el sensor. Las resoluciones soportadas son 640 x 480, 320 x 240, and 80 x 60 pixeles (1).

FIGURAN° 4
DIAGRAMA DEL API DE KINECT



Fuente: (3).

3.5.2. Skeleton Tracking: Permite detectar y seguir los movimientos del esqueleto humano y su partes componentes. Siendo capaz de seguir la pista hasta de seis esqueletos y devolver las posiciones de sus articulaciones (1).

La implementación de la aplicación se la realizó en el Entorno de Desarrollo Integrado (IDE) Visual Studio 2010, empleando el lenguaje de CSharp, empleando el Framework 4.0 de Microsoft.NET.

4. INGENIERÍA DEL PROYECTO

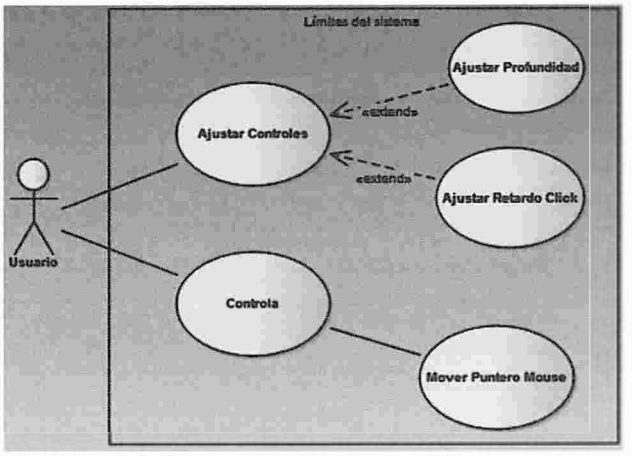
4.1. Diseño del Sistema

El diseño del sistema es presentado por medio de Diagramas de Casos de Uso, Diagrama de Secuencias, y el Diagrama de Clases, empleando la notación UML (Unified Modeling Language).

4.2. Casos de Uso

El diagrama de casos de uso (ver figura N° 5) representa la forma como un Usuario opera con el sistema.

FIGURAN° 5
DIAGRAMA DE CASOS DE USO

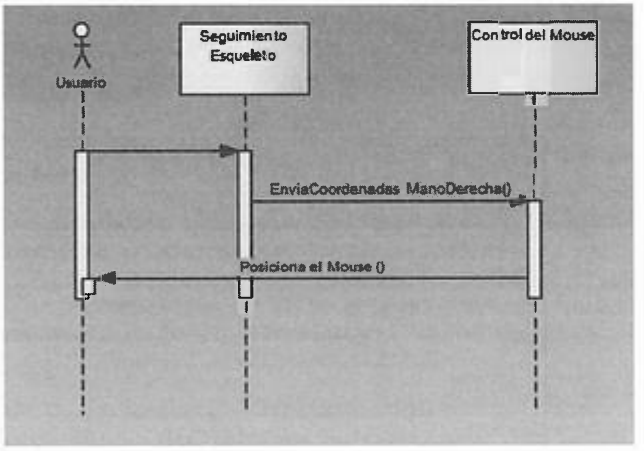


Fuente: Elaboración propia, 2014.

4.3. Diagrama de Secuencias

La figura N° 6 muestra el comportamiento del sistema.

FIGURAN° 6
DIAGRAMA DE SECUENCIAS

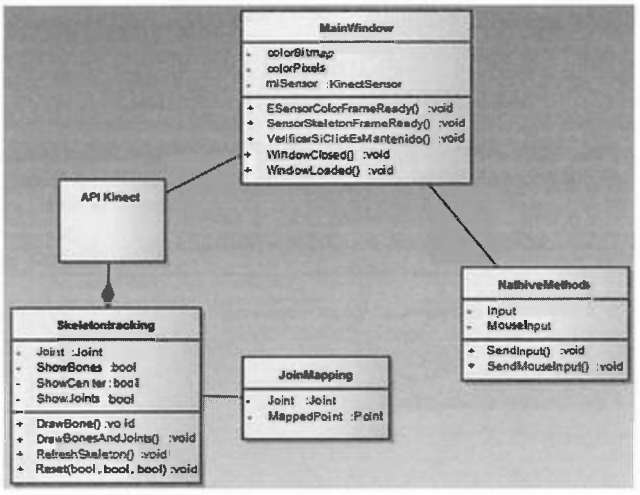


Fuente: Elaboración propia, 2014.

4.4. Diagrama de Clases

Representa el modelo lógico del sistema, y las relaciones entre los objetos utilizados. En la figura N° 7 se observan las clases definidas para la aplicación.

FIGURAN° 7
DIAGRAMA DE CLASES



Fuente: Elaboración propia, 2014.

4.5. Implementación

A continuación se presenta una breve explicación de las porciones relevantes de la implementación en CSharp.

CÓDIGO N° 1
ACTIVACIÓN DE LA CAPTURA

```
//habilitar sensor de Esqueleto
this.miSensor.SkeletonStream.Enable();
//habilitar sensor Color RGB
this.miSensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);
//Habilitar el DepthStream
this.miSensor.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);
//Aplicar el modelo de eventos para la captura de cada frame
this.miSensor.AllFramesReady += SensorAllFramesReady;
this.miSensor.Start();
```

Fuente: Elaboración propia, 2014.

4.6. Captura de imagen

El enfoque empleado es el modelo de eventos (ver Código N° 1), en el cual el sensor Kinect envía el frame o cuadro capturado a la aplicación siempre que un nuevo fotograma es capturado por el sensor (1).

4.7. Seguimiento del esqueleto

La porción de Código N° 2 describe la sección de captura del esqueleto y de las coordenadas de la mano derecha del usuario. La función de escalamiento de las coordenadas de la posición de la mano respecto del tamaño de la pantalla de despliegue y la llamada al método nativo de posicionamiento del cursor del mouse (6).

CÓDIGO N° 2
RASTREO DEL ESQUELETO

```
oreach (Skeleton esqueletos in allSkeletons)
{
// Para el primer esqueleto encontrado
 if (esqueletos.TrackingState == SkeletonTrackingState.Tracked)
 {
   // Tracking de la mano DERECHA
       if (esqueletos.Joints[JointType.HandRight].TrackingState ==
       JointTrackingState.Tracked)
     {
       var muniecaDerecha = esqueletos.Joints[JointType.WristRight];
       //HAce un escalamiento de las coordenadas del Tracking al tamaño de la pantalla
       var manoDerechaEscalada =
       muniecaDerecha.ScaleTo((int)SystemParameters.PrimaryScreenWidth,
       (int)SystemParameters.PrimaryScreenHeight, EsqueletoMaxX, EsqueletoMaxY);
       //Obtiene las coordenadas de la mano DERECHA y las Asigna al Cursor del MOUSE
       cursorX = (int)manoDerechaEscalada.Position.X;
       cursorY = (int)manoDerechaEscalada.Position.Y;

       //Verificar si se hizo Click IZQ
       _oldZ = cursorZ;
       cursorZ = muniecaDerecha.Position.Z;
       _UmbralProfundidad = _oldZ - cursorZ;
```

```
        //VERIF UMBRAL
        ClickIZQ = VerificarSiClickEsMantenido2222(cursorX, cursorY);

        //Método Nativo de Control del Cursor del mouse de Windows
        //Envio del Coordenaras cursorX, cursorY, ClickIZQ
        NativeMethods.SendMouseInput(cursorX, cursorY,
        (int)SystemParameters.PrimaryScreenWidth,
        (int)SystemParameters.PrimaryScreenHeight, ClickIZQ);
      }
    }
}
```

Fuente: Elaboración propia, 2014.

El atributo UmbralProfundidad permite establecer la profundidad de la carrera o extensión que el usuario debe presionar para lograr un click.

El método nativo utilizado finalmente recibe las coordenadas obtenidas: cursorX, cursorY, de la mano derecha y permite el control del puntero del mouse. También recibe como argumento una bandera Booleana de estado: ClickIZQ que indica si se detectó un click por parte del usuario.

4.8. Detección del gesto de Click del mouse

Se empleó un algoritmo que permite detectar el grado de profundidad del gesto de click basado en el umbral de profundidad enmarcado en un límite máximo de tiempo que evita detecciones falsas de click o dobles click seguidos no deseados (ver Código N°3).

El marco de tiempo es ajustable por el usuario y permite definir la sensibilidad y el retardo entre el primer click detectado y el segundo, evitando la saturación de doble clicks.

CÓDIGO N°3
DETECCIÓN DEL GESTO DE CLICK

```
private bool VerificarSiClickEsMantenido2222(int x, int y)
        {
            if (((Math.Abs(x - _oldX)) > 3) || (Math.Abs(y - _oldY) > 3))
                Moviendo = true;
                else Moviendo = false;
              if ((_UmbralProfundidad > Profundidad.Value) && (!clic_largo) && (!
             Moviendo)
                && ((int)_contadorTiempoDeMantenerClick.ElapsedMilliseconds <=
             (RetardoClick.Value* 450)))
            {
                if (!_contadorTiempoDeMantenerClick.IsRunning)
                    _contadorTiempoDeMantenerClick.Start();
                ContadorClicks++;
                if (ContadorClicks < 2)
                {
                    sp2.Play();
                    _oldX = x;
                    _oldY = y;
                    return true;
                }
            }
             if ((int)_contadorTiempoDeMantenerClick.ElapsedMilliseconds >
             (RetardoClick.Value * 650))
                {
                    _contadorTiempoDeMantenerClick.Reset();
                    _contadorTiempoDeMantenerClick.Stop();
                    ContadorClicks = 0;
                    clic_largo = false;
                }
            //   }
            //rescatar Coordenadas anteriores del mouse
            _oldX = x;
            _oldY = y;
            return false;
        }
```

Fuente: Elaboración propia, 2014

4.9. Interface con control mouse

Se empleó el control directo del cursor del mouse por medio de llamada a métodos nativos de Windows (4) (véase Código N° 4).

CÓDIGO N°4
LLAMADA A CÓDIGO NATIVO DE CONTROL DE MOUSE

```
//Método Nativo de Control del Cursor del mouse de Windows
//Envio del Coordenaras cursorX, cursorY, ClickIZQ
NativeMethods.SendMouseInput(cursorX, cursorY,
                    (int)SystemParameters.PrimaryScreenWidth,
                    (int)SystemParameters.PrimaryScreenHeight,
                    ClickIZQ);
```

Fuente: Elaboración propia, 2014.

4.10. Ajustes de sintonía fina.

A fin de mejorar la experiencia de usuario y facilitar el uso de la interface natural con Kinect, se adicionaron las siguientes mejoras:

4.10.1. Filtrado de movimiento.

Se aplicó filtrado de movimiento en los ejes X y Y, que activan la variable booleana Moviendo, que indica que el puntero del mouse se mueve y evita detección de clicks falsos cuando el usuario desplaza la mano con más rapidez (ver Código **N°** 5).

CÓDIGO N°5
FILTRADO DE MOVIMIENTO

```
if (((Math.Abs(x - _oldX)) > 3) || (Math.Abs(y - _oldY) > 3))
                Moviendo= true;
                else Moviendo= false;
```

Fuente: Elaboración propia, 2014.

4.10.2. Suavizado

Para lograr un mejor control con los gestos de movimiento de la mano, se aplicaron las transformaciones de suavizado, que permiten corregir el movimiento errático del mouse (ver Código N° 5).

CÓDIGO N°6
TRANSFORMACIÓN DE SUAVIZADO

```
// Añadido para mejorar el control
      var parameters = new TransformSmoothParameters();
      parameters.Smoothing = 0.5f;
      parameters.Correction = 0.3f;
      parameters.Prediction = 0.4f;
      parameters.JitterRadius = 1.0f;
      parameters.MaxDeviationRadius = 0.5f;
      // fin añadido

      this.miSensor.SkeletonStream.Enable(parameters);
```

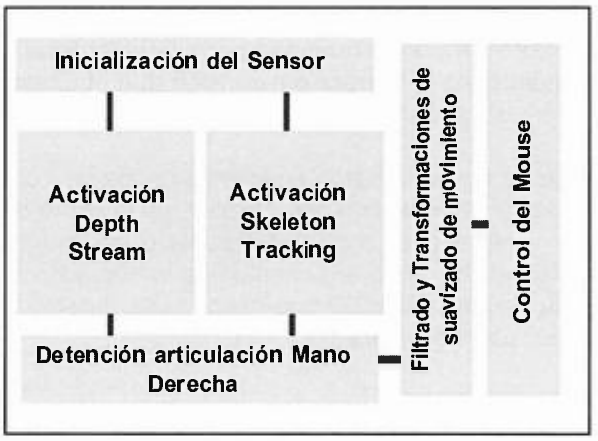
Fuente: Elaboración propia, 2014.

Los valores empleados son los propuestos por defecto en la documentación de la API de Kinect, los mismos que permitieron un movimiento más fluido del cursor del mouse.

4.11. Estructura Final de la aplicación

La Figura N° 8 muestra la estructura final de la aplicación, con la adición del filtrado de movimiento y el suavizado.

FIGURAN° 8
ESTRUCTURA FINAL DE LA APLICACIÓN



Fuente: Elaboración propia, 2014.

5. CONCLUSIONES

La aplicación permite replicar los movimientos de la mano derecha del usuario y transformar en movimientos del cursor del mouse.

Si bien la herramienta SkeletonTracking del API de Kinect, es una herramienta que facilita la identificación y rastreo de movimiento de las articulaciones del usuario identificado por el sensor, las tareas de reconocimiento gestual (hacer Click) presentan un reto interesante por el hecho de los problemas inherentes de cómo interpretar si el usuario hizo un click o no, debido a la distancia del usuario vs la cámara, la profundidad del gesto de la mano, y el tiempo que mantiene presionado.

La aplicación opera dentro de los límites de distancia del usuario y la cámara, ajustando automáticamente la sensibilidad de la profundidad del gesto de click, dentro del rango mínimo de 1.5 metros, hasta un máximo de 3.6 metros.

De igual manera, el escalado de las coordenadas de captura de la mano derecha, permite autoajustar la corrección de distancia en el movimiento del cursor del mouse para los casos que el usuario se encuentre

cerca o alejado de la cámara.

La detección del gesto de click (izquierdo) permite la configuración de dos parámetros: la distancia de profundidad y el tiempo de retardo entre clicks consecutivos.

La configuración de la distancia de profundidad, además permite al usuario calibrar la sensibilidad de detección del click de acuerdo a su comodidad.

La configuración del tiempo de retardo elimina la detección de clicks falsos, y permite inclusive activar la sensibilidad del doble click, mejorando la percepción de usabilidad del usuario.

La adición de las transformaciones de suavizado permiten corregir el movimiento errático del mouse, mejorando la fluidez de su movimiento y su visibilidad en la pantalla, evitando los saltos molestos y facilitando al usuario en su posicionamiento sobre algún área u objeto sobre la pantalla.

6. RECOMENDACIONES

Se recomienda complementar el trabajo con la adición de mayores funcionalidades, que permitan por ejemplo agarrar y arrastrar objetos en la pantalla (drag and drop).

La identificación gestual de posturas del usuario podrían emplearse no únicamente para el control del puntero del mouse, sino que además podrían implementarse a manera de comandos que permitan realizar acciones compuestas como abrir y cerrar alguna aplicación, permitir el desplazamiento o scroll de pantallas o páginas, saltar entre aplicaciones entre otras.

7. REFERENCIAS BIBLIOGRÁFICAS

(1) JANA, ABHIJIT. Kinect for Windows SDK Programming Guide. s.l. : Packt Publishing, 2012.

(2) MILES, ROB. Learn Microsoft Kinect API. s.l. : O'Reilly Media, Inc., 2012.

(3) IRALDE, IÑAKI; PINA ALFREDO. Desarrollo de aplicaciones con Microsoft Kinect. s.l. : UpNa, 2012.

(4) MSDN Microsoft Visual Studio. [En línea] [Citado el: 07 de 01 de 2014.] http://social.msdn.microsoft.com/Forums/vstudio.

(5) Kinect Programming Walkthroughs . [En línea] [Citado el: 13 de 12 de 2012.] http://research.microsoft.com/kinectsdk/.

(6) Coding4Fun Kinect Toolkit. [En línea] [Citado el: 3 de 02 de 2014.] http://c4fkinect.codeplex.com.

Fuentes de financiamiento: Esta investigación fue financiada con fondos de los autores.

Declaración de conflicto de intereses: Los autores declaran que no tiene ningún conflicto de interés.