Juego interactivo con Leap Motion

Cristina Subirana Garcia

Resumen— Leap Motion está diseñado para el reconocimiento 3D gestual de nuestras manos. Por lo tanto, la versatilidad de esta tecnología es muy grande, ya que hay muchas aplicaciones de campos muy diferentes en los cuales el uso de gestos puede facilitar la interacción persona-máquina. La motivación principal de este proyecto fue poder ver todas las posibilidades del dispositivo para manipular entornos 3D. Se trabajó con objetos 3D en la Asignatura "Visualització Gràfica Interactiva", creando un aplicativo Visual C++ que hace uso de la librería gráfica OpenGL. Los dispositivos de entrada utilizados en este aplicativo eran ratón y teclado, mediante los cuales se podía manipular el objeto 3D. Por ello, la idea de este proyecto es aprender el funcionamiento del dispositivo Leap Motion para crear y probar un lenguaje gestual que permita interaccionar con objetos 3D. De esta manera, se añaden nuevas funcionalidades al aplicativo y aquellas que se hacían mediante ratón y teclado se podrán hacer con el uso de gestos mediante Leap Motion.

Palabras clave— Aplicación Visual Studio con Leap Motion, Juego en OpenGL con Leap Motion, Reconocimiento gestual 3D, Transformaciones 3D con Leap Motion

Abstract— Leap Motion is designed to recognize 3D gestures of our hands. There are so many fields where Leap Motion could make a difference, as the use of gestures could change the way people interact with machines. Learning all the possibilities Leap Motion can bring to 3D object manipulation was the main motivation of this project. We had worked with 3D objects at "Visualització Gràfica Interactiva " lectures. We did a Visual C++ application, using OpenGL graphics library. This application had mouse and keyboard as input devices. 3D objects were manipulated by these devices. This project is based on this idea, learning how Leap Motion works in order to create and test a gesture language that let us interact with 3D objects. Thus, new functionalities are added to this application, and others (which were previously activated by mouse or keyboard) will use Leap Motion as an input device.

Index Terms—3D gesture recognition, 3D transformations with Leap Motion, OpenGL Game on Leap Motion, Visual Studio app on Leap Motion

1 Introducción

La forma en que nos comunicamos con los dispositivos ha evolucionado mucho en los últimos años. Se ha pasado de depender de un par de botones de ratón o un conjunto de teclas de un gamepad, a interaccionar mediante gestos. Estos cambios se han introducido en las aplicaciones de escritorio pero donde realmente se ha hecho un cambio de paradigma es en la industria del videojuego. Dispositivos táctiles como el iPhone y las tabletas en general han pasado a ser una de las plataformas escogidas por los desarrolladores haciendo uso de gestos táctiles. Pero Microsoft lanzó al mercado un dispositivo para interaccionar con su consola mediante el movimiento. No se trataba ya de tocar un dispositivo sino de uno que capturase la posición corporal y el movimiento. Se trata de la Kinect. Esta tecnología no se restringe al uso doméstico de la Xbox 360, sino que es una herramienta de investigación, especialmente en el campo de la visión por computador.

Y esto nos lleva a Leap Motion [1]. Un dispositivo de sólo 90 € que se conecta mediante USB a un ordenador Mac o Windows. Este dispositivo se diferencia de la Kinect no sólo por su gran precisión (0.01 mm) sino en que Leap Motion está diseñado para el reconocimiento 3D gestual de nuestras manos [2]. Por lo tanto, la versatilidad de esta tecnología es muy grande, ya que hay muchas aplicaciones de campos muy diferentes en los cuales el uso de gestos puede facilitar la interacción persona-máquina. Pensemos por ejemplo, en el campo de la medicina donde evitando contacto con el dispositivo se ganaría mucho control de higiene. Otro también podría ser la arquitectura, donde el uso de la gestualidad de las manos facilitaría la manipulación de los objetos 3D.

La tecnología de Leap Motion consiste en dos cámaras y tres LEDs de infrarrojos. La intensidad de la luz de los LEDs se ve limitada por la energía que ofrece la conexión USB. El controlador USB del dispositivo lee la información que recibe del sensor y la envía via USB al tracking-software de Leap Motion. El servicio de Leap Motion se encarga de procesar la información de la imagen. Una vez ha compensado la luz ambiental y los objetos de fondo (por ejemplo una cabeza), se analizan las imágenes para reconstruir una representación 3D de lo que captura el dispositivo. La siguiente capa se encarga de extraer la información que coincide con elementos como los dedos.

La motivación principal de este proyecto fue poder ver todas las posibilidades del dispositivo para manipular

- *E-mail de contacto: cristina.subirana@e-campus.uab.cat*
- *Mención realizada: Computación.*
- *Trabajo tutorizado por: Enric Martí i Gòdia (Ciències de la Computació)*
- *Curso 2014/15*

entornos 3D. Se trabajó con objetos 3D en la Asignatura "Visualització Gràfica Interactiva", creando un aplicativo Visual C++ que hace uso de la librería gráfica OpenGL. Los dispositivos de entrada utilizados en este aplicativo eran ratón y teclado, mediante los cuales se podía manipular el objeto 3D. Por ello, la idea de este proyecto es aprender el funcionamiento del dispositivo Leap Motion para crear y probar un lenguaje gestual que permita interaccionar con objetos 3D. De esta manera, se añaden nuevas funcionalidades al aplicativo y aquellas que se hacían mediante ratón y teclado se podrán hacer con el uso de gestos mediante Leap Motion.

Leap Motion tiene una amplia comunidad de desarrolladores, siendo ésta muy activa. Se celebran muchos eventos y hackathons como las 3D Jams y los VR Challenges [3]. Los lenguajes más populares en los que se programan las aplicaciones para Leap Motion son JavaScript y Unity3D.

Las apps de Leap Motion abarcan campos tan variados como la medicina, la música, arte y realidad virtual. No se ha encontrado código abierto en C++ para manipular objetos 3D en Leap Motion mediante OpenGL, ya que la mayoría de la comunidad de desarrollo trabaja con la herramienta de Unity3D. Por lo tanto, en la realización del proyecto, no se ha podido reutilizar código ni comparar con otros proyectos.

Hay publicados artículos [4] [5] [6] [7] que tratan el dispositivo, algunos de ellos haciendo uso de aplicaciones de la propia tienda de LeapMotion.

El objetivo principal de este proyecto es crear y probar un lenguaje gestual que permita interaccionar con los objetos 3D del aplicativo antes mencionado, mediante el dispositivo Leap Motion.

Y para ello se debe pasar por todas estas etapas:

- Experimentación con el dispositivo y su API.
- Definición de un lenguaje gestual para la interacción 3D mediante LeapMotion.
- Implementación y testeo de diferentes interacciones 3D, como transformaciones de punto de vista y transformaciones geométricas.
- Diseño de un juego con la interacción del dispositivo.
- Implementación y testeo del juego.

A continuación, se tratará en detalle el funcionamiento del dispositivo Leap Motion, se plantearán todas las funcionalidades de la aplicación creada y el funcionamiento de cada una de ellas. Se expondrán los resultados obtenidos y los casos de prueba que se han considerado, para finalizar con unas conclusiones.

2 LEAP MOTION

La API de Leap Motion define una clase [9] que representa cada uno de los objetos primarios capturados. Mediante el objeto *Frame* se accede a toda la información de todo aquello que ha sido capturado. Un nuevo *Frame* es creado en cada intervalo de tiempo. Este objeto contiene una lista de las manos, dedos, herramientas, y gestos capturados en ese instante. El objeto *Hand* describe la posición y orientación de la mano, contiene el movimiento entre frames y también una lista de los dedos asociados a esta mano. La clase *Gesture* y sus subclases representan alguno de los cuatro gestos reconocidos por el software Leap Motion. La clase *Vector* es una de las clases de utilidades que se usan en esta aplicación. Esta clase describe puntos y direcciones. También trae sus funciones matemáticas para poder manipular estos vectores.

El dispositivo de Leap Motion reconoce cuatro gestos diferentes:

- Circle: el gesto de dibujar un círculo con el dedo índice. Se trata de un gesto continuo, ya que el dispositivo actualiza el progreso antes de que se inicie el gesto hasta que se finaliza.(véase Fig.1)

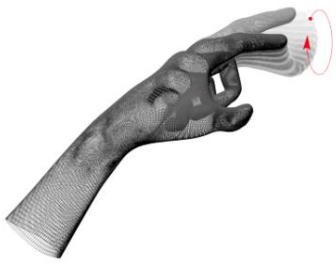

Fig.1. Representación del gesto Circle

- Swipe: el gesto de mover un dedo de forma lineal. También se trata de un gesto continuo.(véase Fig.2)

Fig.2. Representación del gesto Swipe

- KeyTap: el gesto equivalente a teclear un teclado, es decir un gesto rápido del dedo en sentido descendiente. Se trata de un gesto discreto.(véase Fig.3)

- ScreenTap: el gesto equivalente a tocar la pantalla, es decir movimiento rápido del dedo en dirección perpendicular al anteriormente mencionado KeyTap. También es discreto.(véase Fig.4)

Fig.3. Representación del gesto KeyTap

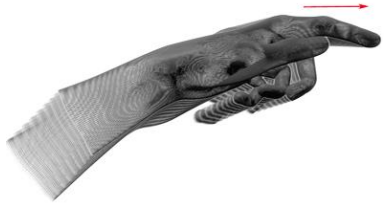

Fig.4. Representación del gesto ScreenTap

El servicio Leap Motion (véase Fig.5) recibe los datos del controlador a través del bus USB. Se procesa esta información y se envía a las aplicaciones Leap-enabled que se estén ejecutando. De forma predeterminada, este servicio solamente envía los datos a las aplicaciones que estén en primer plano. Sin embargo, las aplicaciones pueden configurarse para que reciban los datos en segundo plano (pudiéndose cancelar por medio del usuario).

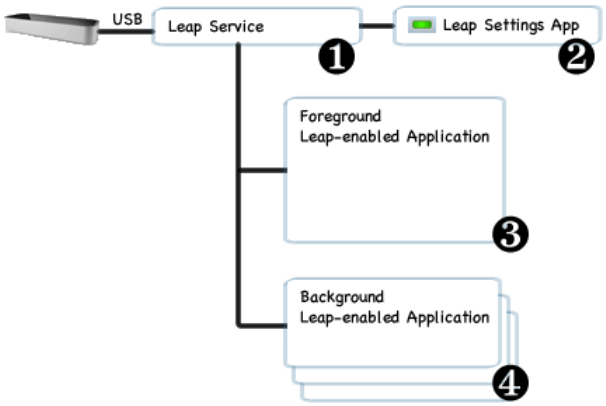

Fig.5: Arquitectura Sistema Leap Motion [1]

La aplicación de Leap Motion se ejecuta de forma separada del servicio y permite al usuario configurar la instalación del dispositivo. La aplicación de Leap Motion es una applet en el panel de control en Windows y una aplicación en la barra de en Mac OS X.

La aplicación en primer plano, recibe toda la información de motion tracking desde el servicio. Para que la aplicación pueda hacer uso del Servicio del dispositivo debe conectarse con la API. Esta API está guardada en una librería dinámica. Esta librería se conecta al servicio de Leap Motion y entrega la información capturada a la aplicación. Se puede enlazar directamente a la librería en las aplicaciones desarrolladas en lenguaje C++.

Cuando una aplicación que usa Leap Motion pasa a segundo plano, el Servicio deja de enviar datos a esta aplicación: Se debe pedir permisos para seguir obteniendo estos datos en segundo plano ya que al ponerse en segundo plano los ajustes de configuración que gobiernan son los de la aplicación que esté en primer plano.

3 DESARROLLO

Las funcionalidades desarrolladas en la aplicación han sido las siguientes:

- Conexión y Desconexión con el controlador del dispositivo Leap Motion.
- Transformación del punto de vista.
- Escalado, Rotación y Traslación.
- Juego estilo Simon.

Todas ellas se gestionan en la aplicación como eventos relacionados con su recurso de menú. Se hace uso de la información que contiene el controlador para interaccionar con la escena 3D.

Para cada funcionalidad se ha definido el lenguaje gestual asociado para que el usuario interaccione en el entorno 3D.

Se ha decidido crear un gesto propio para el escalado (véase Fig.7a y Fig.7b), ya que el que reconoce el dispositivo se debe realizar con las dos manos. De esta forma, el uso del gesto propio nos permite usar una sola mano para hacer el escalado del objeto.

Se ha decidido usar el gesto de KeyTap para el juego tipo Simon por tres razones: 1) Es uno de los gestos más precisos junto con el Circle. 2) Es el gesto más parecido al que se haría en el juego físico Simón. 3) El hecho de ser un gesto discreto lo hace más sencillo de implementar y debugar.

Seguidamente, se detalla el funcionamiento específico de cada funcionalidad.

3.1 Conexión y desconexión a Leap Motion

Para crear la conexión con el dispositivo se crea un objeto controller mediante su constructor. La desconexión se hará mediante el destructor.
La función con la que se obtiene la información capturada por el dispositivo en cada frame es frame().

Hay dos maneras de obtener estos frames: mediante polling Y mediante callbacks. En el primer caso se trata de llamar a la función frame() del objeto controller(), cuando la aplicación esté preparada para capturar un frame del dispositivo. Es la forma más sencilla y más habitual de obtener los frames. La segunda forma consiste en crear dos objetos, un controller y un listener. El objeto listener obtendrá los frames con el frame-rate configurado por el dispositivo. El objeto Controller llamará la función On-

Frame() del Listener cada vez que haya un frame preparado para ser transmitido, y dentro de la función OnFrame() se llamará a la función frame() del objeto Controller. Esta implementación es mucho más compleja ya que los callbacks son multithread.

Se ha decidido, por lo tanto, hacer servir la técnica de polling para la interacción 3D y el juego, ya que de esta forma desde la aplicación se controla en qué momento se quiere capturar el frame que se recibe del controller. Así se ha podido suavizar el efecto de parpadeo que se origina al refrescar la imagen.

3.2 Transformación del punto de vista

Sabiendo que las coordenadas del punto de vista se corresponden con el sistema de coordenadas de cámara, y que actualmente se mueven por mouse, se quiere transformar este sistema mediante la interacción con el dispositivo Leap Motion.

Para el usuario las trasformaciones de punto de vista se harán moviendo la mano izquierda. Cuanto más se aleje la mano, más se alejará el punto de vista. La implementación se basa en consultar el dispositivo con una cierta frecuencia (la cual se podrá ir configurando dependiendo del clock de la cpu y de los fps que nos interesen) y adquirir la información de la palma de la mano izquierda que nos será facilitada por el controlador del dispositivo.

Se obtendrán las coordenadas esféricas (véase Fig.6) (R, alfa, beta) del punto de vista mediante la fórmula (1), (2) y (3) respectivamente, siendo x, y, z las coordenadas del centro de la palma de la mano derecha.

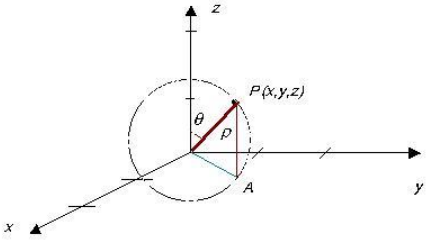

Fig.6. Coordenadas esféricas y cartesianas

$$R = \sqrt{x^2 + y^2 + z^2} \qquad (1)$$

$$alfa = \frac{\arcsin\left(\frac{z}{R}\right)\cdot 180}{\pi} \qquad (2)$$

$$beta = \frac{\arctan\left(\frac{y}{x}\right)\cdot 180}{\pi} \qquad (3)$$

3.3 Traslación, Escalado y Rotación

Las transformaciones geométricas que se aplicaran al objeto que se visualice en la escena serán:

- Traslación (tx, ty, tz) por lo tanto desplazamiento para x, y, z.
- Escalado (Sx, Sy, Sz) véase factor de escala para x, y,z.
- Rotación respecto eje x, eje y, eje z.

Para el usuario, las transformaciones geométricas que actualmente se hacen por mouse y por teclado, se harán moviendo la mano derecha. La implementación se basa en consultar el dispositivo en una cierta frecuencia (la cual se podrá configurar) y adquirir la información de la mano derecha que será facilitada por el controlador del dispositivo.

Las variables de estado de transformaciones existentes en el entorno: trasl, rota, escal, serán las que condicionaran qué transformación geométrica se va a aplicar (traslación, rotación y escalado respectivamente).

El escalado del objeto será proporcional al radio de la esfera imaginaria que cabría dentro de la mano del usuario, tal y como puede verse en la fórmula (4). De esta forma cuando el usuario cierre la mano derecha el radio de esta esfera imaginaria se reducirá (véase Fig.7a y Fig.7b).

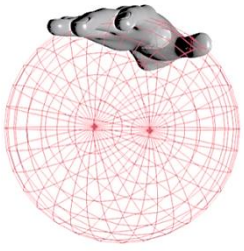

Fig.7a. Gesto amplitud del escalado

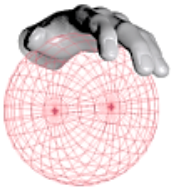

Fig.7b. Gesto reducción del escalado

$$VScal.x = \frac{hand.sphereRadius()}{70} \qquad (4)$$

La traslación hará uso del vector de traslación entre el frame actual i el previo. Véase fórmula (5), siendo VTras.x la coordenada x del vector de traslación de nuestra aplicación, antFrame el frame previo.

$$VTras.x = hand.translation(antFrame).x \qquad (5)$$

La rotación hará uso del vector de rotación entre el frame actual i el previo. Véase fórmula (6), siendo VRota.x la coordenada x del vector de rotación de nuestra aplicación, antFrame el frame previo y Leap::Vector::xAxis() el eje X.

$$VRota.x = \frac{hand.rotationAngle(antFrame, xAxis()) \cdot 360}{\pi} \quad (6)$$

3.4 Juego de esferas

El desarrollo del juego de esferas está formado por:

- Una escena 3D (véase Fig.8).
- Sus botones y el tratamiento de sus eventos, para así escoger las diferentes funcionalidades del juego.
- Una escena simplificada para así poder seleccionar la esfera deseada mediante la técnica de Picking.
- La clase Simón encargada de gestionar el algoritmo del juego, guardando los estados de éste.
- Toda la gestión del reconocimiento de gestos para seleccionar la esfera.

El diseño de la escena 3D, el objeto sobre el cual se interacciona, está formado por cuatro esferas de diferentes colores separadas de forma equidistante entre sí. Éstas pueden tener su color respectivo a la mitad de intensidad y al máximo de intensidad para que se vean iluminadas.

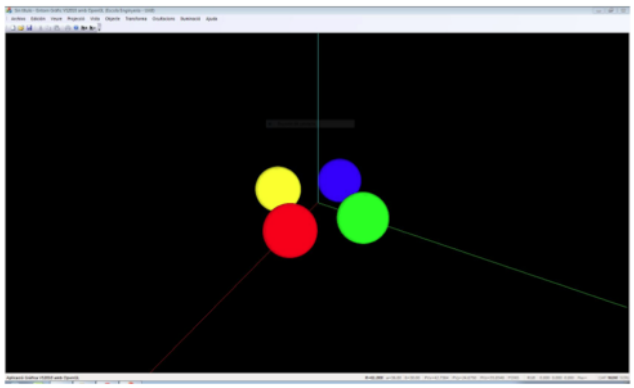

Fig.8. Escena 3D del Juego

En cuanto a los recursos de menú se ha creado uno para cada funcionalidad: "inicia" para la funcionalidad de iniciar el juego, "visualitza" para la funcionalidad de visualizar la secuencia, y "juga" para entrar en el modo de juego. Una vez creados los recursos de menú se creó su código de tratamiento de evento y sus variables de estado (de tipo booleano).

La función de interrupción usará estas variables de estado de estas tres formas: 1) Si se está en modo "inicia" se llamará al *Generador Secuencia* 2) Si se está en modo "visualitza", llamará al *Visualizador Secuencia*. 3) Si se está en modo "juga", y ya se ha hecho conexión con el dispositivo Leap Motion, se capturará la información de la mano derecha y se llamará al *Verificador Secuencia* (véase Fig.9).

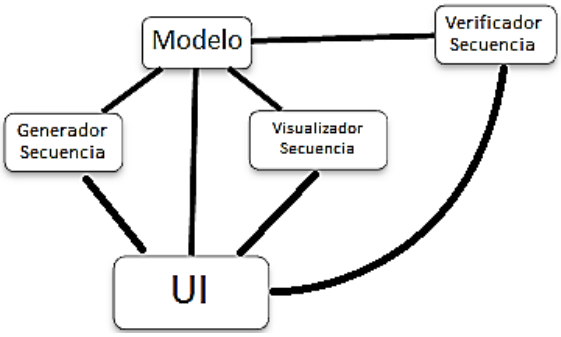

Fig.9. Diagrama de módulos del Juego

El *Generador Secuencia* genera una secuencia de valores de uno a cuatro (siendo 1 el equivalente a la esfera roja, 2 para la esfera verde, 3 para la esfera azul y 4 para la esfera amarilla) mediante una función de generación de números aleatorios. Esta función seguirá la formula (7), siendo r el número aleatorio generado, usando la función srand() la cual crea la semilla para la generación de número aleatorio

$$r = 1 + rand()\%4 \quad (7)$$

El *Visualizador Secuencia* recorre toda la secuencia. En cada iteración se marca como iluminada la esfera que indica esa posición de secuencia. Hay una fase intermedia entre cada iteración en la que se visualizan todas las esferas sin iluminar.

El *Verificador Secuencia* usa la posición del cursor en pantalla en el instante en que se captura el gesto para obtener la esfera seleccionada. Esta esfera se obtiene mediante la técnica de picking. La secuencia nos dice que esfera es la correcta para esta iteración. Por lo tanto, se compara la esfera seleccionada con la correcta. Si coinciden, se incrementa el contador de aciertos y el índice para recorrer la secuencia de juego. En caso que se haya seleccionado la esfera incorrecta, se incrementará el contador de errores, y se iniciará el índice para recorrer la secuencia. También se modificará las variables de estado, estando activo el modo "juga" y "visualitza". De esta forma se volverá a mostrar toda la animación desde el inicio, para recordar al jugador la secuencia a memorizar.

La forma de interactuar con la escena del juego mediante el dispositivo Leap Motion se podría separar en dos partes: primero posicionando el puntero alrededor de la escena acorde al movimiento de nuestra mano y otra seleccionando un pixel haciendo el gesto KeyTap.

La forma de posicionar el puntero en una posición dentro de pantalla es haciendo uso de la función de Windows setCursorPos (int x, int y) [10] [11] [12], siendo *x,y* la coordenada x,y del centro de la palma de la mano que nos da el controlador para ese frame. Se deben hacer los cálculos necesarios para pasar de coordenadas de dispositivo a coordenadas de pantalla como puede verse en la

fórmula (8), siendo w el ancho de la ventana y k una constante que nos permite adaptar la sensibilidad.

$$positionX = k \cdot centre\_palmell\_dreta.x + \frac{w}{2} \qquad (8)$$

4 RESULTADOS

4.1 Pruebas de las funcionalidades

Una vez expuesto el desarrollo de la aplicación, seguidamente se detallan los resultados obtenidos para cada funcionalidad.

4.1.1 Transformación del Punto de Vista

Para activar esta funcionalidad se debe estar conectado al dispositivo y tener la mano izquierda dentro de la zona de interacción de Leap Motion. En la Fig.10a puede verse el objeto originalmente y en la Fig.10b el objeto después de aplicarle una transformación del Punto de Vista.

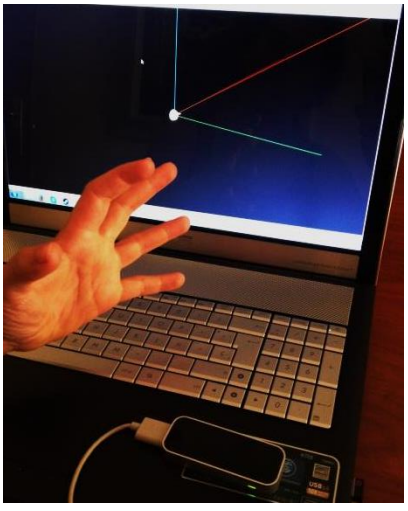

Fig.10a. Objeto previo a transformación Punto de Vista.

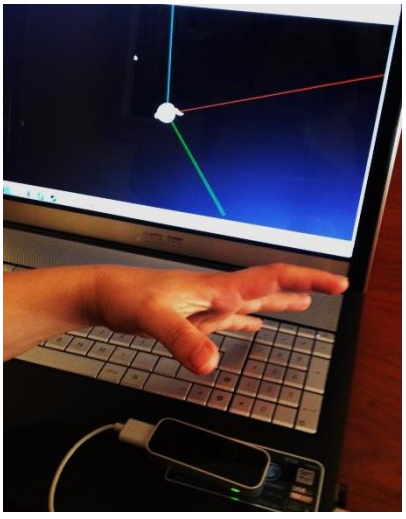

Fig.10b. Objeto posterior a transformación Punto de Vista.

4.1.2 Traslación, Escalado y Rotación

Para trasladar un objeto se debe estar conectado con el dispositivo, tener el objeto cargado en perspectiva y seleccionar dentro del desplegable de menú "Transformaciones" la opción Traslación. El objeto original (Fig.11a) se trasladará tal y como puede verse en la Fig.11b.

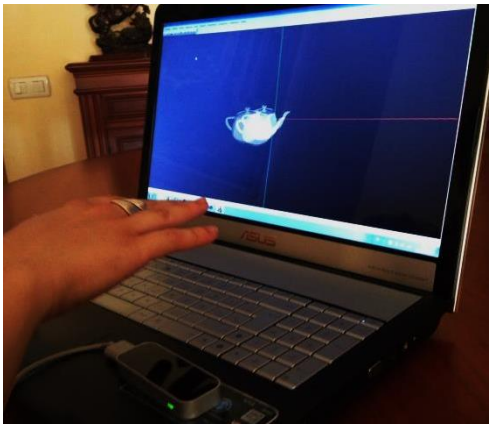

Fig.11a. Objeto original.

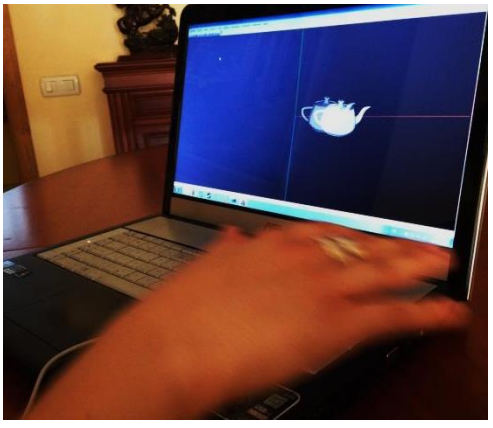

Fig.11b. Objeto trasladado.

Para escalar un objeto se debe estar conectado con el dispositivo, tener el objeto cargado en perspectiva y seleccionar dentro del desplegable de menú "Transformaciones" la opción Escalado. A continuación puede verse la ampliación del objeto (Fig.12a) y su reducción (Fig.12b).

Fig.12a. Objeto ampliado

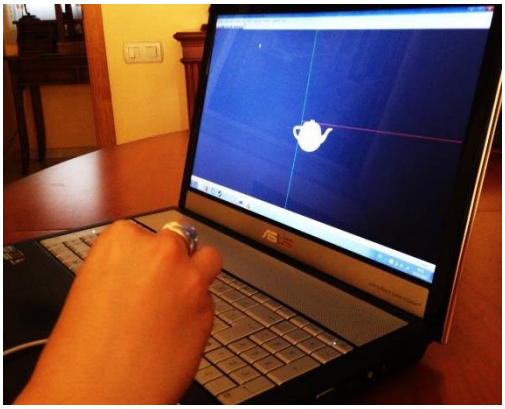

Fig12b. Objeto reducido.

Para rotar un objeto se debe estar conectado con el dispositivo, tener el objeto cargado en perspectiva y seleccionar dentro del desplegable de menú "Transformaciones" la opción Rotación. El objeto original se rotará tal y como puede verse en la Fig.13a y 13b.

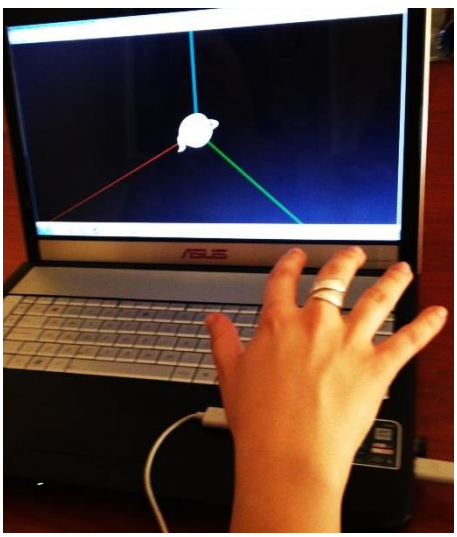

Fig.13a.Objeto original.

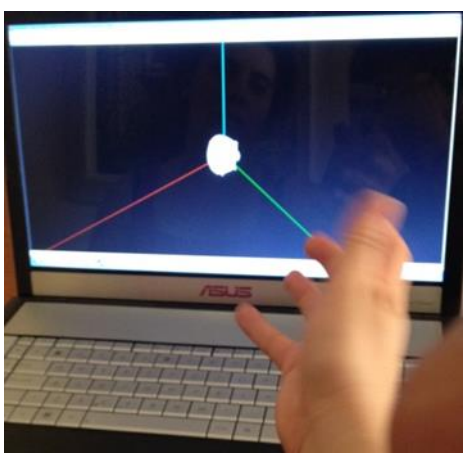

Fig.13b. Objeto rotado respecto eje Z

4.1.3 Juego de Esferas

Para poder iniciar el juego se debe estar conectado con el dispositivo y seleccionar dentro del desplegable de menú "Objeto" la opción Simon->"Inicia". Se visualizará la escena 3D del juego

Para visualizar la secuencia a memorizar se debe estar conectado con el dispositivo y seleccionar dentro del desplegable de menú "Objeto" la opción Simon->"Visualiza". Se visualizará la animación de la escena 3D del juego (Fig.14a y 14b).

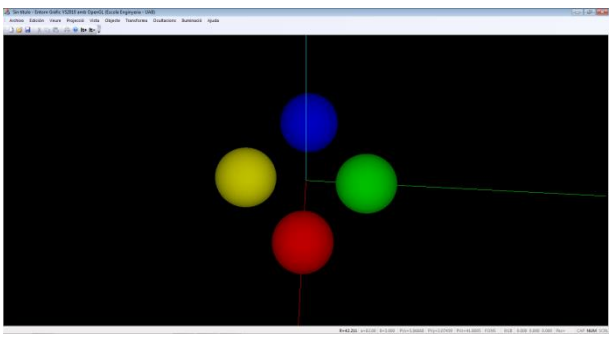

Fig.14a. Escena con esfera sin iluminar

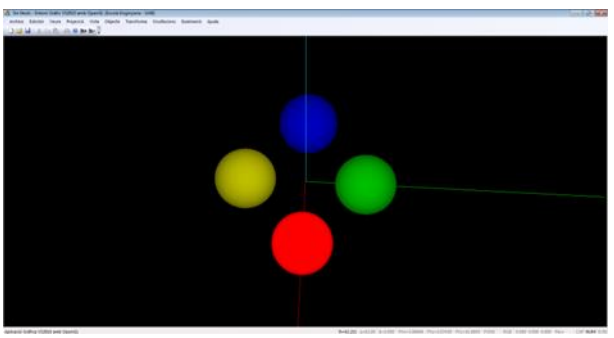

Fig.14b. Escena con esfera roja iluminada

Para jugar al juego, una vez vista la secuencia a memorizar, se debe estar conectado con el dispositivo y seleccionar dentro del desplegable de menú "Objeto" la opción Simon->"Juega". Se podrá seleccionar la esfera que nos convenga posicionado el puntero con el movimiento de la mano encima de la esfera que nos interese y seleccionándola haciendo el gesto keyTap. Si la esfera seleccionada no es correcta se visualizará la animación de nuevo y se volverá al modo juego.

4.2 Escenarios de pruebas

Se han creado diferentes escenarios de pruebas para asegurar un buen funcionamiento de la aplicación. También se ha probado en profundidad la respuesta del dispositivo de entrada Leap Motion, y al ser éste un dispositivo que captura imágenes de las cámaras se ha visto necesario tener en cuenta otros factores externos como la iluminación ambiental.

4.2.1 Interacción con Leap Motion

Para probar los límites de funcionamiento del dispositivo se ha generado un código que muestra por consola la información de la mano siguiente: id frame, cantidad de manos, posición del centro de la palma de la mano, si es mano derecha o izquierda, vector de rotación, vector de traslación, radio de la esfera que cabría dentro de la mano y los gestos que se han reconocido junto con su tipo.

Se han hecho las pruebas en diferentes condiciones de luz ambiental y en equipos con diferentes características. En condiciones de luz intensa y en equipos más básicos se han encontrado pequeños retrasos en el refresco ya que los fps bajaron. Estos problemas de performance se han solucionado ejecutando la aplicación en un equipo de características avanzadas (Procesador i7 y tarjeta gráfica Nvidia con 2GB de GPU) en un entorno menos iluminado.

La distancia con el dispositivo también ha sido un factor clave en un buen rendimiento ya que posicionarse en áreas más lejanas al dispositivo hace que deje de reconocer las manos, volviendo a capturarse de nuevo la mano una vez vuelve a estarse dentro de la zona de interacción. La posición de la mano correcta para el uso del dispositivo es con la palma de la mano paralela al dispositivo, para evitar posibles oclusiones.

Se han probado diferentes estilos interacción, reaccionando con suavidad y también a la inversa, haciendo gestos secos y rápidos. Tal y como se esperaba, cuando se hacen gestos secos el dispositivo no se comporta con tanta precisión. Se pueden ver situaciones como que el dispositivo deje de reconocer el dedo como estirado, especialmente el dedo anular.

Se ha observado que los gestos circulares (circle) y de arrastre (swipe) son muy robustos, a diferencia de los gestos que simulan tocar la pantalla (screenTab) o teclas (keyTap). Para estos últimos es necesario partir de un estado en que el dedo esté quieto. Para solucionarlo se ha tenido en cuenta este detalle y se ha configurado la velocidad mínima de movimiento del dedo y la distancia mínima que debe recorrer el dedo.

Para el caso del escalado, se ha observado que el cambio en el valor del radio de la esfera que cabe dentro de la palma de la mano, no era demasiado continuo en relación al movimiento de apertura y cierre de la mano. Había cierto estancamiento en algunos instantes sobre todo en el gesto de apertura. El radio de apertura mínimo oscilaba entre 30 y 40 unidades, estando el radio de apertura máxima entre 80 y 100 unidades. Para manos más grandes, se ha observado valores superiores a 100.

4.2.2 Visualización de la secuencia

Las pruebas que se han hecho han sido siempre para secuencias de tamaño 10 generadas aleatoriamente mediante la fórmula expresada anteriormente en su sección de desarrollo.

También se ha tenido en cuenta el caso en que la secuencia tenga valores repetidos secuencialmente. La primera implementación de la animación de la secuencia no tenía en cuenta los casos en que se iluminaba una esfera de forma continuada, véase una secuencia del estilo: Azul-azul-rojo-verde, y por lo tanto se mantenía iluminada la esfera azul hasta que se iluminase el rojo. La solución fue crear un estado intermedio entre cada paso de la animación de forma que cuando se encontraba en este estado intermedio se pintaba la escena 3D sin iluminar. De esta forma se diferenciaba a la perfección que esfera estaban iluminadas sea cual sea la secuencia a mostrar.

4.2.3 Modo Juego

Siendo ésta una de las partes más complejas del proyecto, es la que más tiempo de pruebas ha consumido. Se podrían separar en 2 partes: algoritmo juego y Picking.

Se ha probado el algoritmo del juego, siguiendo la traza de la ejecución mediante debugging, consultando el valor de las variables locales en toda la ejecución, incluida la secuencia que se está jugando. Éste algoritmo se ha probado tanto al haber seleccionado las esferas mediante ratón como haciendo uso del gesto de keyTap en el dispositivo.

Se ha probado el picking con uso del ratón para así aislar defectos propios del picking. Se ha probado con diferentes técnicas de iluminación y con iluminación deshabilitada. Se ha observado que en la escena simplificada es necesario trabajar sin iluminación y con polígono lleno tanto por la cara delantera como para la trasera, para así obtener unos resultados correctos. Se ha probado también que el reconocimiento de la esfera seleccionada funcionaba de forma correcta tanto en el centro de las esferas como en los márgenes de éstas.

5 CONCLUSIONES

- Se ha definido un lenguaje gestual para la interacción 3D mediante LeapMotion. Se ha implementado y testeado diferentes interacciones 3D, como transformaciones de punto de vista y transformaciones geométricas.
- Se ha diseñado, implementado y testeado un juego interactivo con el dispositivo.
- Se recomienda el uso de un ordenador potente para obtener un buen performance del controlador de Leap Motion.
- Como posibles mejoras, se podría implementar una calibración, especialmente para el caso de escalado.
- Se reserva también para trabajos futuros, el crear gestos nuevos y de esta manera no será necesario el uso del ratón en ninguna zona de la aplicación.

REFERENCIAS Y BIBLIOGRAFÍA

[1] Leap Motion Official Website. URL: https://www.leapmotion.com (last visit April 2015).

[2] Review sobre Leap Motion. Título: Leap 3D Out-Kinects Kinect . Autor: David Zax– URL: http://www.technologyreview.com/view/427981/leap-3d-out-kinects-kinect-video/ (last edition May 2012).

[3] Developer Events & Hackathons Guide. URL: https://developer.leapmotion.com/events (last visit June 2015).

[4] J.Han and N.Gold, "Lessons Learned in Exploring the Leap Motion Sensor for Gesture-based Instrument Design", Proceedings of the International Conference on New Interfaces for Musical Expression. URL: http://nime2014.org/proceedings/papers/485_paper.pdf.

[5] K. Aldar et al, "Leap Motion", Weekly Science Research Journal, Vol 2, Issue 25, 1st Jan 2015. URL: http://www.weeklyscience.org/UploadedArticle/129.pdf.

[6] S.Jimenez, "Physical interaction in augmented environments", Master Thesis Report, 2014, URL: http://master-3dmt.eu/wp-content/uploads/2014/07/ JIMENEZ_Samuel_MasterThesis_comp.pdf.

[7] R.Burton, "Motion Controlled graphics Applications", URL:http://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1031&context=cscsp.

[8] E. Martí. Apuntes de las asignatura Visualització Gràfica Interactiva. (UAB), 2014.

[9] Leap Motion API - URL: https://developer.leapmotion.com/ documentation/cpp/devguideLeap_Tracking.html (last visit June 2015).

[10] Blog de William A Adams, donde se explica la implementación de las funcionalidades del ratón haciendo uso del Leap Motion. URL: https://williamaadams.wordpress.com/2013/03/20/ handy- mouse-using-the-leap-as-a-mouse-controller/ (last visit May 2015).

[11] Respuesta a una consulta a Stackoverflow, sobre el uso de SetCursor-Pos. http:// stackoverflow.com/ questions/22259936/c-move-mouse-in-windows-using-setcursorpos (last visit May 2015).

[12] Web UCanCode.net donde se explican las funciones de tratamiento de los eventos del Mouse en una aplicación MFC. URL:http://www.ucancode.net/ Visual_C_Source_Code/MFC-Sample-Code-TrackMouseEvent-GetCapture-SetCapture-ReleaseCapture -GetCursorPos.htm (last visit May 2015).