Pequeña introducción al proyecto

Desde hace años estamos acostumbrados al uso de gráficos 3d en múltiples áreas de la informática. A pesar de que normalmente el modelo mostrado esté diseñado realmente en 3d, el hecho de tener que mostrarlo en un display 2d, como un monitor, hace que el usuario perciba que lo que se le muestra no está realmente en 3d.

Para mejorar esta ilusión de profundidad, normalmente se usa alguna solución basada en visión estereoscópica, pero el uso de esta técnica suele venir acompañada de limitaciones importantes: normalmente requiere el uso de hardware adicional, o las configuraciones son estáticas y hacen que el efecto solo sea realista desde un punto de vista específico.

En el mundo real, cuando observamos un objeto y cambiamos de posición cambia nuestro punto de vista, y por tanto la imagen que observamos de dicho objeto. Aunque no se ofrezca a cada ojo una imagen separada, podemos lograr que el usuario perciba un efecto de tridimensional si conseguimos replicar este hecho. Para poder conseguirlo, es necesario que el sistema conozca la posición relativa del observador (o mejor dicho, de sus ojos) respecto al objeto (y al display en el que se le muestra) y que con esta información calcule como modificar la cámara para conseguir que se muestre una imagen coherente del objeto tal y como lo vería el observador si el objeto estuviese en el mundo real y no fuese una imagen en una pantalla.

Sin duda, una de las principales causas de que aún no se haya popularizado ésta clase de tecnología es la falta de un periférico o una serie de periféricos de coste asequible para el público, una plataforma estándar que permita a los desarrolladores liberarse de la complejidad de tener que crear una solución individual para cada producto en el que quieran implementar este seguimiento de la posición del usuario (el head track o face track). Si por el contrario, para el programador fuese tan fácil obtener la información necesaria para realizar estos cambios de cámara (básicamente, la posición del observador respecto a la pantalla) como lo es obtener la que hace falta para colocar el puntero en la pantalla, esto simplificaría enormemente el dilema, y sin duda veríamos un rápido auge en el uso de esta técnica.

Ahí es donde intenta encajar este proyecto. No creando un dispositivo hardware para realizar el head tracking, sino implementando una plataforma que permita usar dispositivos de los que el usuario ya dispone, aunque en un principio no estén diseñados para realizar esta función, y que partiendo de esta base heterogénea de dispositivos de entrada, se realice el head track y se reporten los datos al sistema en un formato que a cualquier desarrollador le sea fácil usar, en este caso, y siguiendo con el ejemplo de "tan fácil como usar un ratón", en forma de un dispositivo Linux (/dev/ht por poner un ejemplo).

Debido a la naturaleza del proyecto, éste estará compuesto por dos partes bien diferenciadas: un módulo central (el dispositivo de tracking) y un conjunto de piezas independientes, de tamaño mucho menor, que serán las demostraciones que ilustrarán las funcionalidades y diversos usos que se le puede dar al dispositivo de tracking.

Sistema centralizado de Tracking para Linux

Ésta será la pieza central del proyecto: una plataforma software que permitirá al usuario crear uno o varios dispositivos virtuales de tracking, a partir de otros dispositivos de los que ya dispone, y que no están, en un principio, diseñados para ésta función.

Algunos de los requisitos y funcionalidades del sistema serán:

- El resultado final (los datos que se vuelquen en /dev/ht) deben ser lo más homogéneos posible, haciendo transparente a la aplicación que los esté usando cual es el dispositivo real de entrada (para la aplicación simplemente se estará usando un dispositivo de track).
- Añadir compatibilidad para nuevos dispositivos de entrada (como la cámara del Project natal de Microsoft planeada para el año que viene) debe ser lo más simple posible.
- La configuración del dispositivo debe ser sencilla para el usuario.
- El uso de recursos del sistema no debe ser demasiado elevado.
- Se debe poder usar triangulación (visión estereoscópica) para mejorar la precisión en el cálculo de la profundidad.
- Se debe soportar sistemas multipantalla (es decir, se reportarán los datos para cada una de las pantallas.
- Cuando sea posible, se debe poder calibrar automáticamente un segundo dispositivo de entrada gracias al input generado por un primer dispositivo de entrada. (Ejemplo: si tenemos configurado el sistema con un wiimote, y añadimos una webcam, después de un tiempo de uso deberíamos haber podido obtener la posición de la webcam)
- Integración con VRPN: debe ser posible exportar un dispositivo de HT, así como importar un wiimote para usarlo de input
- Para mejorar el ratio de FPS, o simplemente suavizar el movimiento o eliminar vibraciones, se implementará un predictor/suavizador de movimientos.
- Se ha de poder aplicar más de un tracking (incluso de diferentes técnicas) a una misma imagen (aplicable a webcams) (por ejemplo trackear 2 personas)
- El posicionamiento de los dispositivos de entrada (cámaras, wiimote) debe ser lo más flexible posible. (que no sea obligatorio que, por ejemplo, la cámara esté centrada y encima de la pantalla)
- Debe ser posible usar más de una cámara para mejorar el campo de visión total del dispositivo virtual. (Si una cámara cubre 30 grados de ángulo de visión en horizontal, usando dos de esas cámaras deberá poder cubrirse una zona de más grados).

Para completar el proyecto será interesante también hacer una comparativa de todos los métodos implementados, comparando rendimiento, fps, precisión, pros y contras de cada sistema y conclusiones.

En un principio, el sistema soportará wiimotes (tanto en local como vía VRPN), y como webcams. En las siguientes subsecciones se detallan algunas características e ideas sobre las implementaciones concretas de ambos tipos de input.

Wiimotes

El mando de la Wii dispone de una cámara infrarroja capaz de dar la posición de hasta cuatro puntos móviles con una resolución de 1024x768 a una frecuencia de 100Hz. La cámara es en realidad de 128x96, pero el mando realiza un pre proceso que permite afinar esta resolución en un factor de 8. Debido a este pre proceso, la posición de los 4 puntos es la única información disponible para el usuario, impidiendo usar la cámara para recobrar, por ejemplo, la imagen sin procesar. El campo de visión de la cámara es de 33 grados en horizontal y 23 en vertical. Si se quita el filtro infra-rojo de la cámara, puede usarse para dar información sobre cualquier punto brillante en luz visible, no solo infrarroja.

Además de la cámara, el input del mando se completa con tres acelerómetros (uno para cada eje, con un rango de 3 a -3 g con un 10% de precisión que se reporta en 8 bits) que funcionan también a 100Hz. Otros componentes, que en un principio no son tan relevantes para el proyecto, son 11 botones de uso general más uno de encendido/apagado que no puede ser reprogramado con los drivers actuales.

El mando se puede conectar a cualquier pc gracias a la tecnología Bluetooth. En un principio, se usará el driver cwiid (http://github.com/abstrakraft/CWiid), ya que de entre los drivers de wiimote para Linux, es el que dispone de un mantenimiento mas activo.

Más información sobre las características técnicas del wiimote en http://johnnylee.net/projects/wii/ y http://wiibrew.org/wiki/Wiimote .

Viendo el video de Johnny Lee (http://www.youtube.com/watch?v=Jd3-eiid-Uw), es fácil entender cómo se puede usar el wiimote para realizar head track. Básicamente, el wiimote se coloca de cara al usuario, que usando como marcadores leds infrarrojos (en el caso de Johnny Lee, montados en unas gafas) podrá ser localizado.

Webcams

A estas alturas, una webcam no necesita presentación. Es un periférico bastante común en cualquier entorno doméstico (muchos portátiles lleva incluso una integrada. Suelen ofrecer resoluciones de unos 640x480 a 30 frames por segundo.

Para obtener los datos necesarios para el tracking usando webcams, en un principio se usarán 2 librerías: OpenCV y ARToolkit.

OpenCV

OpenCV es una librería orientada a la visión por computador. Entre las funcionalidades que ofrece podemos encontrar la detección de caras. Usando esta funcionalidad podemos determinar en qué punto de la imagen obtenida por la cámara se encuentra la cara del usuario, y de ahí calcular la posición relativa del usuario a la pantalla.

Otra opción sería la de emular el comportamiento del wiimote, procesando la imagen para obtener la posición de los leds infrarrojos en las gafas del usuario.

Posiblemente durante el desarrollo del proyecto, se intenten otras aproximaciones (distinguir siluetas con respecto a los cambios en el fondo de la imagen, por ejemplo). En posteriores documentos se

definirán con mucho más detalle los detalles de la implementación del tracker con OpenCV, ya que es, a priori, una de las partes del proyecto que más flexibilidad ofrece a la hora de programar diversas opciones y variantes.

ARToolkit

Aunque en un principio ARToolkit está pensado para aplicaciones de realidad aumentada y los ejemplos que podemos encontrar por internet introducen elementos generados por ordenador en una imagen capturada por una cámara, en la página oficial encontramos:

"The ARToolkit video tracking libraries calculate the real camera position and orientation relative to physical markers in real time."

"Las librerías de seguimiento de ARToolkit calculan la posición real de la cámara y la orientación relativa respecto a los marcadores físicos en tiempo real."

Los marcadores de ARToolkit son sencillos rectángulos con diseños en blanco y negro que pueden ser fácilmente identificados. Si el usuario lleva alguno de ellos (en una gorra, por ejemplo), y sabemos la posición relativa de la cámara a la pantalla, al calcular la posición del usuario respecto a la cámara podremos obtener la posición del usuario respecto a la pantalla, que es exactamente lo que necesitamos para realizar el head track.

Esta técnica posiblemente sea la única de las usadas que dé información sobre los 6 grados de libertad.

Demostración de las funcionalidades

Para ilustrar mejor los requisitos y las capacidades que ha de tener el dispositivo de head track implementado, lo mejor es, sin duda, describir las demostraciones que debe ser capaz de realizar.

Usos simples: ajustar la perspectiva usando Head Track

Esta es la idea principal que se nos presenta en la mayoría de videos en internet (como por ejemplo el de Johnny lee): Usando head track sabemos la posición relativa del observador respecto a la pantalla. Sabiendo esto, podemos ajustar la perspectiva consiguiendo transmitir la sensación de que estamos viendo un objeto tridimensional, a pesar de estar usando un display 2d. Aunque esta técnica tiene sus limitaciones: normalmente solo sirve para un usuario, y el efecto solo se aprecia realmente en movimiento (cuando el usuario no se está moviendo la imagen queda fija, y aunque la perspectiva sea correcta, al no estar en movimiento, el efecto de profundidad se pierde).

Posibles implementaciones:

- Visor de modelos en 3d: Sin duda una de las demostraciones más básicas sería la de conseguir el código de algún visor simple de modelos en 3d, y adaptarlo al ajuste de perspectiva usando HT.
- Google Street view: El uso de fotos de 360 grados de ésta aplicación puede adaptarse bien al uso de HT, simulando que nuestra pantalla es en realidad la ventana del coche que ha tomado las fotos
- Escritorio en 3d: Aparte de las adaptaciones obvias de usar modelos 3d con Head Track para el fondo o los salvapantallas, el uso de Head track puede ser interesante en un caso de escritorio

3d, donde las ventanas estén situadas en capas a diferentes niveles de profundidad. Idealmente se podría adaptar esta idea a entornos de escritorio reales, como compiz, aunque una simple demo simulando como podrían integrarse estos efectos en el escritorio puede ser muy ilustrativa

Uso avanzado

El head track nos permite dar un paso más allá que el de simplemente ajustar la perspectiva para dar un efecto de profundidad a un display 2d, algunos ejemplos que sería interesante demostrar:

- Reajuste de elementos dependientes del usuario, no de la escena: El caso más ilustrativo de esto es una interfaz de usuario en un display virtual. Dar la ilusión de profundidad a una escena y después superponer a ésta un HUD o una Interfaz en el plano del display no ayuda a mantener la ilusión de que el display es una ventana a una escena que realmente está en 3d. Reajustar este tipo de elementos según la perspectiva del usuario puede ser la solución.
- Interacción del sistema con el usuario: Conocer la posición del usuario respecto a la pantalla puede ayudar a mejorar la inmersión del usuario en otras formas además del ajuste de perspectiva. Por ejemplo, que un personaje de la escena mire directamente al usuario (o a otras personas en caso de haberlas), o el ajuste de la iluminación si las luces dependen de la posición del usuario (por ejemplo simular que está llevando una linterna).
- Modo sin pantalla o efecto ventana: El uso normal del head track es el de ajustar la perspectiva de una escena 3d para simular profundidad. Como se puede ver en este video (http://www.youtube.com/watch?v=RycnyBCrEFQ&feature=response_watch) es posible también hacer lo mismo con escenas reales, pudiendo simular que la pantalla es transparente (si la escena es del mismo lugar) o que tenemos una ventana a otro lugar (en caso de no serlo).
- Usar pantalla partida para más de un usuario: Una solución típica para entornos multiusuario con distintos puntos de vista en un solo display es la de usar una porción del display para cada usuario. Uno de los puntos flojos del head track es que solo da la ilusión de profundidad al usuario que se trackea. Con una demo de este estilo se demostraría la capacidad de la plataforma para trackear más de un usuario a la vez.
- Cálculos en tiempo real de las cámaras necesarias para visión estereoscópica: Una parte bastante engorrosa de la configuración en visión estereoscópica es el configurar correctamente las cámaras. Normalmente en las pocas soluciones que hay para pcs domésticos (básicamente el uso de drivers 3d de nvidia y similares), esto se soluciona con una configuración estática, lo que provoca que el efecto 3d sea correcto solo dependiendo de la posición del usuario, y que la imagen se vea incorrectamente si el usuario se mueve. Usando head track se puede conocer (o deducir) la posición real de los ojos, y configurar adecuadamente las cámaras para conseguir crear una imagen estereoscópica correcta.
- Adaptación al uso con más de una pantalla: El uso de más de una pantalla para mostrar una imagen o una escena no es nuevo. El problema es que normalmente, a no ser que las pantallas estén contiguas y en el mismo plano, la imagen no es del todo coherente. Usando head track en ambas pantallas con una configuración individual para cada una de ellas se puede solucionar el problema, consiguiendo una mayor inmersión.

Usando el Tracking como input

El tracking (no necesariamente de la cabeza del usuario) no solo se puede usar para conseguir mejorar la inmersión del usuario en una escena 3d, sino que también puede servir para mejorar las capacidades de interactuación con el sistema.

- Pantalla o superficie táctil: Hoy en día se usan cada vez más los dispositivos con pantallas táctiles, y su adaptación a todos los medios es cada vez más completa. Sin embargo muchos usuarios no disponen aún de una pantalla táctil en su pc. Si somos capaces de conocer la posición de los dedos del usuario con suficiente precisión, podremos (usando 2 puntos de vista) triangular su posición, y mediante un simple calibrado, extrapolar esa posición a la pantalla.
- charIO: Seria interesante poder reproducir los resultados de este proyecto (http://imve.informatik.uni-hamburg.de/projects/chairio) usando tracking, posiblemente mediante el método de ARToolkit
- Integración con Blender: Seria interesante mostrar cómo puede integrarse head track en otras soluciones software, como por ejemplo usar el ajuste de perspectiva para simplificar el hacer pequeños ajustes mientras se modela con blender
- Reconocimiento básico de gestos, como asentir y negar con la cabeza deberían ser relativamente fáciles de reconocer usando head track, y puede ser útil para mejorar una interacción más natural con el pc.
- Apuntar en 3d: Posiblemente esta sea la demo más ambiciosa del proyecto, en ella se unen head track, tracking de otro objeto y el uso tradicional del wiimote como apuntador. Conociendo como ajustar el punto de vista gracias al head track, si conocemos la posición del wiimote mediante tracking y a qué punto está apuntando el wiimote, podemos trazar una trayectoria y ver en qué punto de la escena 3d intersecta. La idea es que el usuario no apunte a la parte del display donde se muestra lo que quiere apuntar, sino que apunte al objeto o zona que le interesa como si de verdad existiese.

Otros puntos a tener en cuenta

Para la presentación del software generado en el proyecto, sería muy interesante lograr incluir todo el material listo para funcionar en un LiveCD Linux. Posiblemente esto se haga con el nuevo servicio de Novell, Suse Studio (http://susestudio.com/), para el cual conseguí una invitación este verano tras explicar el uso que quería darle en este proyecto.

Todo el proyecto será, en un principio de código abierto. La licencia concreta aun no está decidida, pero gran parte del proyecto posiblemente se aloje en un repositorio público, para permitir el libre acceso del público a él, y tener la seguridad de tener un sistema robusto de backup.

Por último, antes de empezar el proyecto pensaba crear un diario sobre su desarrollo. Finalmente éste diario tomará la forma de un weblog, ya que ésta alternativa ofrece varias ventajas interesantes. Este weblog se puede encontrar en http://plagaslair.blogspot.com .

Control de cambios

v1.0 - Primera versión

v1.0.1 – Completada la introducción, cambiada la cabecera

v1.1 – Correcciones de estilo, añadidos más detalles en varias secciones