Memoria Técnica

Aplicación controlada mediante gestos detectados con Kinect 360

Autores: Antonio Jesús Heredia Castillo
Jose Manuel Perez Lendinez

Índice

Dispositivo usado

El dispositivo usado es un Kinect v1. Este kinect es la versión creada para la consola Xbox 360. Al ser el primer modelo creado, cuenta con unas características reducidas en comparación con dispositivos similares o su versión v2.

Sus características más destacadas son que puede detectar hasta 6 personas y 20 articulaciones por cada una. Su distancia de funcionamiento es de como mínimo de 2 metros para que funcione correctamente. También cuenta con un sensor de distancia mediante infrarrojos.

Implementación

Como base del proyecto hemos usado un proyecto existente anterior, el cual era una aplicacion de pintura. Aunque hemos tenido que añadir la mayoría de funcionalidades que usamos, pero nos facilitaba el la obtención de algunos datos.

Mover objetos por la pantalla

La mayoría de ejemplos que vimos, no tenían la opción de mover un objeto por la pantalla, teniendo la mano cerrada. Nosotros hemos añadido la opción de que al cerrar la mano, el programa detecte lo que se encuentra debajo del puntero y lo mueva con el. Tenemos una lista de botones (imágenes), predefinidos, los cuales clonamos para poder añadir mas cuadros al canvas.

```
if (isHandOver(oldPoint, buttons))
{
    stopDraw = false;
    primeraProfundidad = getprofundidad();

    if(selected.Uid != "elegido")
    {
        stopDraw = true;

        var nuevo = ClonarBoton(selected);
        Canvas.SetLeft(nuevo, newPoint.X - selected.Width / 2);
        Canvas.SetTop(nuevo, newPoint.Y - selected.Height / 2);

        this.canvasKinect.Children.Add(nuevo);
        selected = nuevo;
        buttons.Insert(0, nuevo);

    }

}
```

Una vez que esta cogido el objeto hasta que no haga el movimiento de abrir la mano, no dejaría de seguir al puntero.

```
InformacionObra(selected.Name);
Canvas.SetLeft(selected, newPoint.X-selected.Width/2);
Canvas.SetTop(selected, newPoint.Y-selected.Height/2);
profundidad = getprofundidad();
double widthNuevo = selected.Width - (primeraProfundidad - profundidad) * 1.5;
double heightNuevo = selected.Height - (primeraProfundidad - profundidad) * 1.5;

if (widthNuevo > 75 && heightNuevo > 75 && widthNuevo < 500 && heightNuevo < 500)
{
    selected.Width = widthNuevo;
    selected.Height = heightNuevo;
}
```

Una vez que abre la mano, suelta el objeto en la posición donde se encuentre. Además comprueba si está dentro de los márgenes y si no lo está lo metería dentro.

Aumentar o disminuir el tamaño de una imagen

El proceso de aumentar o disminuir una imagen se puede diferenciar en dos partes. La primera es cuando el objeto es “agarrado”, en ese momento capturamos la profundidad a la que se encuentra la segunda mano (la que no está como primaria).

```
if (isHandOver(oldPoint, buttons))
{
    stopDraw = false;
    primeraProfundidad = getprofundidad();
```

La segunda parte es cuando se está arrastrando la imagen y además aumenta o disminuye la profundidad de la segunda mano. En ese momento se cambia las dimensiones de la imagen.

```
profundidad = getprofundidad();
double widthNuevo = selected.Width - (primeraProfundidad - profundidad) * 1.5;
double heightNuevo = selected.Height - (primeraProfundidad - profundidad) * 1.5;

if (widthNuevo > 75 && heightNuevo > 75 && widthNuevo < 500 && heightNuevo < 500)
{
    selected.Width = widthNuevo;
    selected.Height = heightNuevo;
}
```

También tenemos unos tamaños mínimos establecidos, para que no pueda hacerse o demasiado pequeña o demasiado grande.

Deslizar hacia derecha o izquierda

Esta funcionalidad nos permite cambiar entre las distintas salas del museo. La implementación de esta funcionalidad ha sido algo más costosa que las demás. Al no existir documentación oficial de kinect, no hemos podido encontrar si existía o no. Por ello la hemos implementado nosotros desde 0. Para comprobar si el deslizamiento lo hacía de forma horizontal, pero a su vez tener un poco de margen para que fuera un movimiento “humanamente” de realizar, hemos puesto que no pudiera superar hacia arriba o hacia abajo una determinada distancia, respecto del inicio de la acción.

```
{
    if (Math.Abs(puntoActual.Y - puntoInicial.Y) > 0.1)
    {
        reconociendo = false;
    }
    else
    {
```

Si se pasaba paraba de reconocer la acción.

Lo siguiente que comprobamos para ver que la acción se realiza de forma adecuada es que sea de una determinada distancia, para que no fuera demasiado sensible y con cualquier movimiento lo hiciera. Para ver si es izquierda o derecha, la distancia recorrida será positiva o negativa respectivamente, teniendo esto en cuenta actuamos en consecuencia.

```
{
    if (puntoActual.X - puntoInicial.X > 0.4)
    {
        TimeSpan stop_local = new TimeSpan(DateTime.Now.Ticks);
        if ((stop_local - start).TotalMilliseconds < 200)
        {
            CambiarPagina(1);
            reconociendo = false;
        }
        else
        {
            reconociendo = false;
        }
```

Y por último la última comprobación que hacemos es que la acción se realice en un determinado tiempo, para que así, se hiciera con una cierta velocidad(recordad que antes medimos la distancia que recorría).

Acción cuando se toca la cabeza

Otro movimiento introducido, es que cuando se toca la cabeza elimina todo lo presente en la zona de trabajo. Para ello, hemos comprobado la distancia que hay entre el punto de la cabeza y el de la mano derecha. Teniendo un pequeño margen. La función creada, permite elegir si quieres comprobar la mano izquierda o la derecha.

```
    }

    Joint cabeza = esqueleto.Joints[JointType.Head];
    Joint mano;
    if (derecha)
        mano = esqueleto.Joints[JointType.HandRight];
    else
        mano = esqueleto.Joints[JointType.HandLeft];

    return distanciaPuntosSkeleton(cabeza.Position, mano.Position);

}
```

En la función distanciaPuntosSkeleton simplemente calculamos la distancia entre 2 puntos en 3 dimensiones, teniendo en cuenta los ejes x,y,z.
Para que haya un poco de margen, la distancia necesaria para que se active la función es mayor que 0 y menor que 0.2.

```
if (distanciaManoCabeza(true) > 0 && distanciaManoCabeza(true) < 0.2)
    vaciarCanvas();
```

Métodos usados de la liberia de kinect

De la liberia de Kinect, lo único que hemos usado ha sido el onclick de los botones de kinect. Estos botones funcionan cuando tienes el cursor sobre ellos y además se acerca la mano al sensor. En estos botones hemos implementado funcionalidades tales como borrar todo lo que hay en la pantalla, reorganizar todas las imágenes, borrar todas las imágenes de un tipo y mostrar los cuadros que hay en la sala de un determinado cuadro.