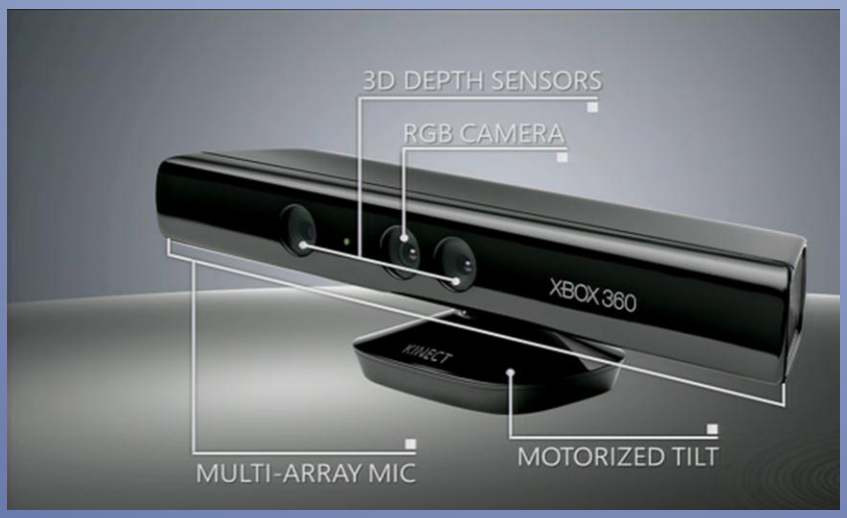


Nuevos Paradigmas de la Interacción

Practica 2 – Detección Movimiento Fitness

Desarrollo de una clase que controla el movimiento de Fitness de realizar estiramientos de los dos brazos por encima de la cabeza. Implementación de dicha clase y de un proyecto de demostración.

Realizado por:

José Delgado Dolset

Descripción del programa:

El programa consiste en una ventana que muestra los flujos de color y de Esqueleto del Kinect. De esta forma, el usuario puede verse a sí mismo en pantalla, así como el esqueleto artificial detectado por el Kinect. Se ha implementado una clase "P2_FitnessMove", que controla la realización de un movimiento de Fitness (estirar los brazos). Dicha clase, va controlando distintas posiciones para realizar el movimiento completo. En los comentarios del código se ha explicado con todo el detalle posible las distintas decisiones de implementación y qué se ha hecho para controlar los distintos movimientos. No obstante, aquí se explicarán algunos detalles que por su extensión o algún otro motivo no me ha parecido adecuado indicar en los comentarios, aparte de realizar un breve resumen de esa clase.

Implementación:

La clase consiste en una serie de miembros que controlan los parámetros del movimiento, así como las fases que ha realizado el movimiento definitivo. A estas variables, se suman distintos métodos que comprueban tanto el estado de estos miembros como los valores del esqueleto sobre el que se esté trabajando. Los más importantes son los siguientes:

```
/// <summary>
/// Porcentaje de error admitido en el movimiento.
/// </summary>
private double tolerancia;

/// <summary>
/// Valor que indica cuanto se tiene que mantener una postura para reconocerla:
/// </summary>
private int vecesNecesarias;

/// <summary>
/// Cadena de caracteres que se usa para almacenar el feedback del movimiento:
/// </summary>
private String feedBack;

/// <summary>
/// Valor que indica cuantas veces hay que hacer el estiramiento:
/// </summary>
private int repeticiones;
```

Como vemos, tenemos dos enteros que guardan parámetros necesarios para realizar cálculos referidos al tiempo que se debe mantener una pose y el número de veces que hay que realizar cada estiramiento. También se usa una cadena de texto para guardar el feedback, es decir, la información que permite a la persona que esté realizando el ejercicio recolocarse en una posición correcta o cuál es la siguiente posición. La tolerancia es un valor que nos sirve para indicar el error permitido en las posiciones en tanto por uno. Se han añadido los métodos Get y Set necesarios para controlar estos cuatro miembros clave de la clase (para el feedback solo el Get).

Se ha preparado la interfaz para que se puedan observar y modificar la tolerancia, el número de repeticiones y la duración de cada postura. Para el caso concreto de la tolerancia, se transforma el valor en un entero, que representa el porcentaje, para que sea más claro para el

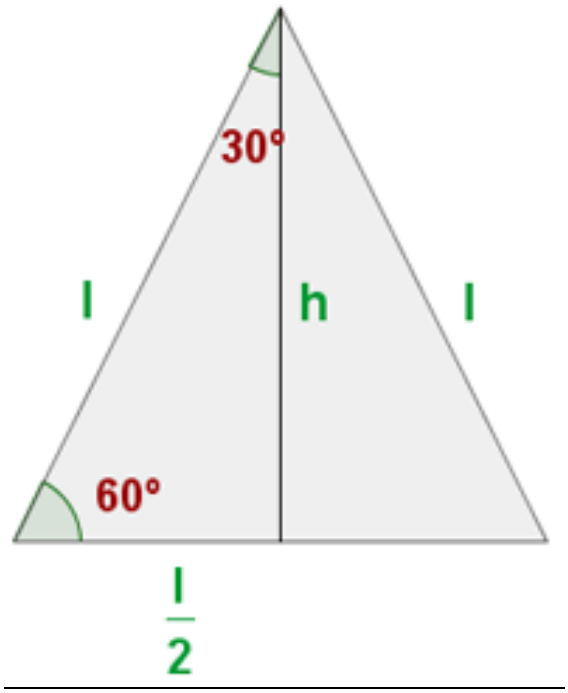
usuario. También se muestra el contenido del “feedback” de forma que el usuario pueda recolocarse para seguir realizando el movimiento.

Luego se ha dividido el código en distintos métodos que se usan para comprobar posturas distintas. Si se ha visto que una postura tenía en cuenta muchas partes del cuerpo, con el objetivo de facilitar la legibilidad del código se ha reestructurado en submétodos que controlan alguna parte concreta de forma independiente. Por ejemplo, la posición inicial tiene en cuenta por separado la espalda (que ha de estar recta) las piernas (deben de estar abiertas un ángulo de unos 30-60º) y los brazos (en cruz con la espalda). Luego se llama a estos métodos dentro del método posición inicial. Lo mismo se ha hecho con el método principal, pero aquí se ha añadido también el control para que la postura se mantenga durante un tiempo y para que se repitan la cantidad de veces necesaria.

El resto de miembros de la clase son booleanos que controlan las distintas fases del movimiento y contadores para el tiempo y las veces. También se ha añadido un método para reiniciar el movimiento (que pone todos estos miembros a false y los contadores a 0). Este es el funcionamiento de la clase. Sin embargo, creo conveniente explicar algunas decisiones de implementación, o la base teórica en la que se basan algunos métodos:

*bool piernasAbiertas(Joint LeftFoot, Joint RightFoot, Joint Hip):

Este método comprueba que las piernas estén abiertas un ángulo de entre unos 30º y unos 60. Pero para simplificar los cálculos, se ha utilizado una propiedad de los triángulos equiláteros: Estos triángulos se caracterizan por tener los tres lados iguales y sus tres ángulos son cada uno de 60º. De esta forma, la separación máxima de los pies ha de ser igual a la distancia que hay de un pie a la cadera (60º) y la mínima de la mitad de esa distancia (30º). Esto nos evita tener que calcular ángulos y simplifica los cálculos. En la siguiente imagen hay un diagrama de lo que he explicado.



*bool manoSobreCabeza(Joint Wrist, Joint Head) y

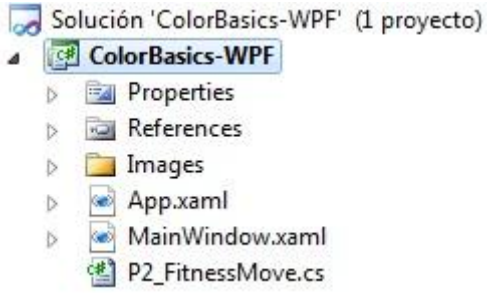
*bool manoEnCadera(Joint Wrist, Joint LeftHip, Joint RightHip):

Estos métodos funcionan para ambos brazos. Para ello, se han implementado de forma que inviertan los cálculos si nos encontramos en el caso de que sea un brazo u otro. De esta forma, no hay que desarrollar métodos repetidos. Esta misma estrategia se ha usado para calcular aplicar la tolerancia a un lado y al otro de los ejes X, Y y Z. El siguiente fragmento de código tiene un ejemplo de esto.

```
if (Wrist.JointType == JointType.WristLeft)
{
   x = (double)(LeftHip.Position.X - Wrist.Position.X);
   y = (double)(LeftHip.Position.Y - Wrist.Position.Y);
   z = (double)(LeftHip.Position.Z - Wrist.Position.Z);
}
else
{
   x = (double)(RightHip.Position.X - Wrist.Position.X);
   y = (double)(RightHip.Position.Y - Wrist.Position.Y);
   z = (double)(RightHip.Position.Z - Wrist.Position.Z);
}
```

Instrucciones de Uso de la Clase

1) Lo primero es colocar el archivo “P2_FitnessMove.cs” en la carpeta correspondiente del proyecto de Visual Studio. Hay que tener en cuenta que usa el namespace “namespace Microsoft.Samples.Kinect.SkeletonBasics” así que si no lo está ya, habrá que añadir las directivas correspondientes.

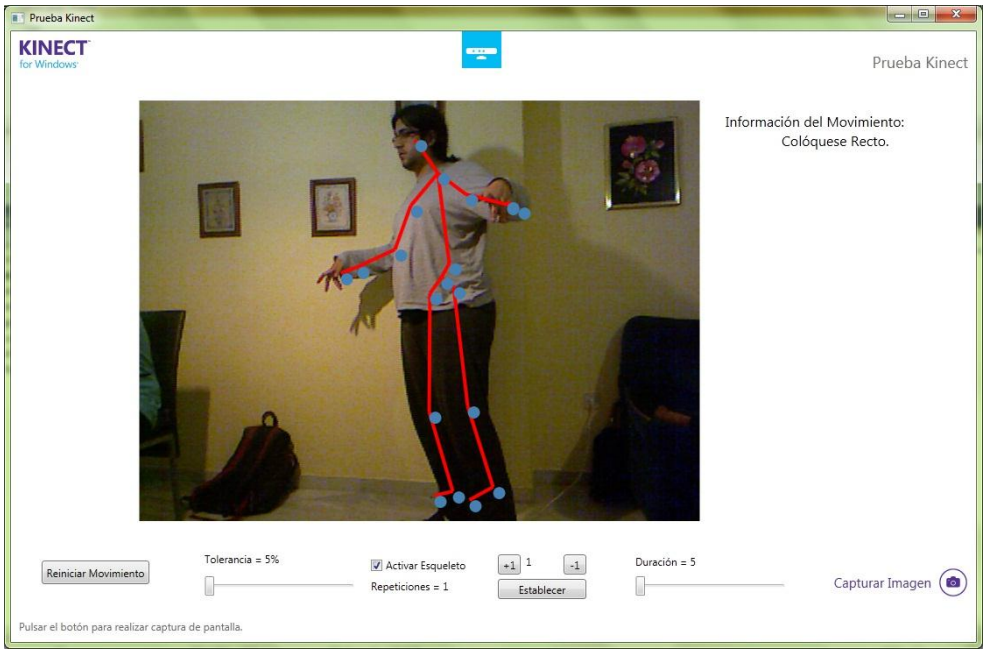


2) Hecho esto, hay que declarar un objeto de la clase “P2_FitnessMove” como miembro de la clase principal. Podemos inicializarlo directamente:

```
/// <summary>
/// Objeto que controla las posturas a realizar:
/// </summary>
private P2_FitnessMove controlMovimiento = new P2_FitnessMove(0.05, 10, 1);
```

Nota: Los parámetros que le he pasado al constructor son los que ya tiene por defecto, por lo que, en este ejemplo, es equivalente a **new P2_FitnessMove().**

3) Para controlar los parámetros de este objeto una vez creado, se puede invocar a los métodos Set implementados, ya sea usando variables para almacenar los valores a establecer o mediante un valor directamente. En nuestro caso, hemos creado controles en la interfaz para establecer estos valores y un manejador para cada controlador, así como mensajes de información para el usuario. Aquí una vista de la interfaz.

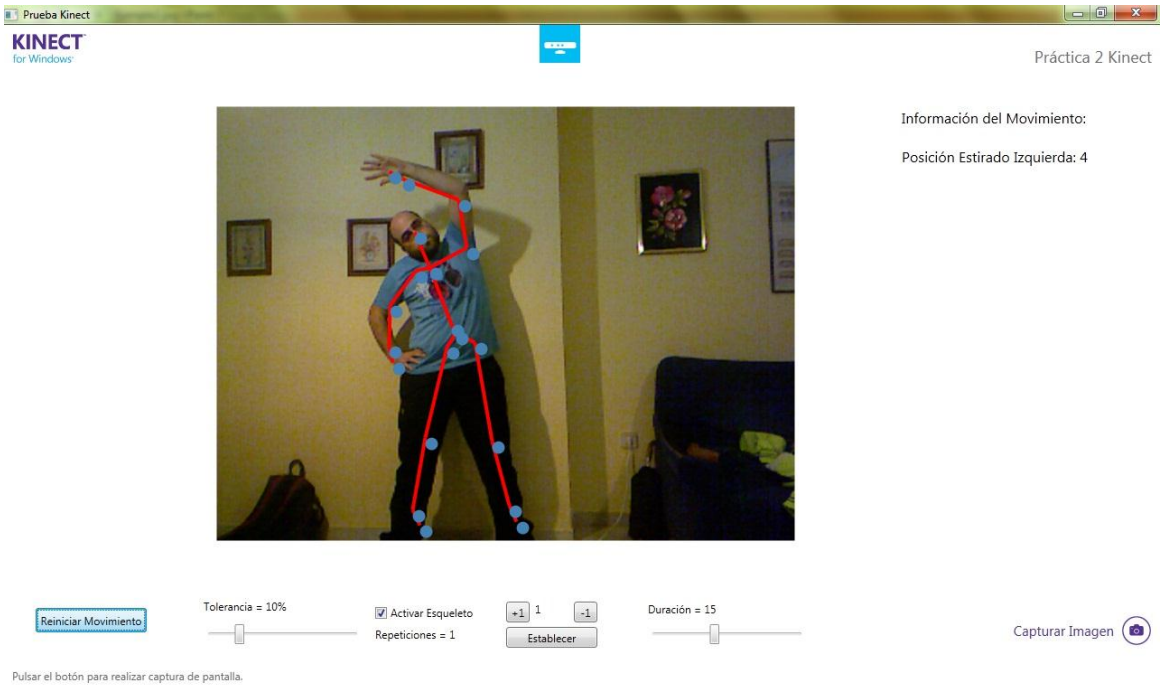


4) Una vez fijados los parámetros, solo tenemos que llamar a la función booleana que nos controla si se realiza bien el movimiento o no y usar ese dato para lo que queramos realizar. En nuestro caso, pintaremos el esqueleto en dos colores distintos. Un ejemplo del código para esto:

```
// Si todo el esqueleto está siendo detectado (TRACKED)
if (esq.TrackingState == SkeletonTrackingState.Tracked)
{
    // Si es correcto:
    if (controlMovimiento.movimientoRealizado(esq))
        pintarEsqueleto(esq, Colors.Green, Colors.Yellow);

    // Si no lo es:
    else
        pintarEsqueleto(esq, Colors.Red, Colors.SteelBlue);
}
```

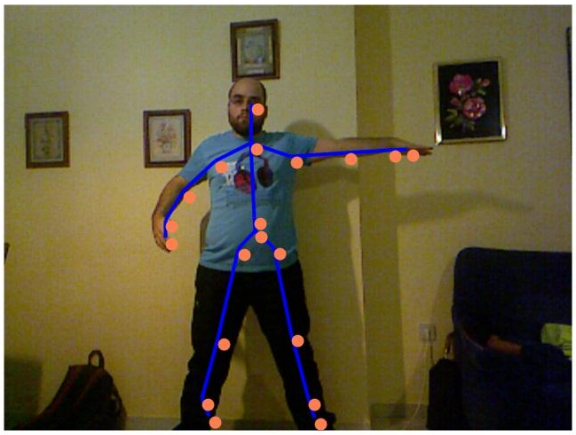
Aquí algunas capturas de un ejemplo de uso:



Prueba Kinect

KINECT for Windows

Práctica 2 Kinect



Información del Movimiento:

Coloque la mano izquierda
apoyada en su cadera.
Para ello siga bajando la mano
izquierda hacia ella.

Reiniciar Movimiento

Tolerancia = 10%

Activar Esqueleto

Repeticiones = 1

+1 1 -1

Establecer

Duración = 15

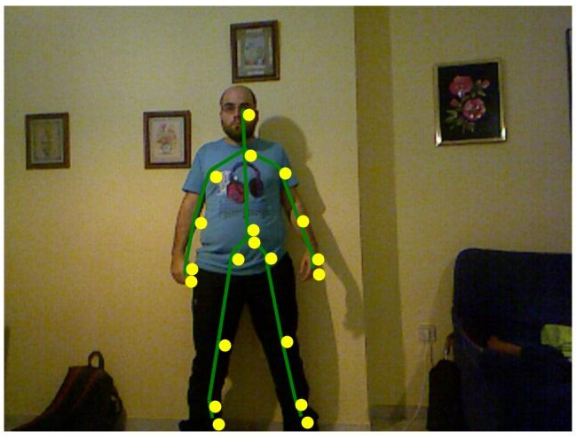
Capturar Imagen

Pulsar el botón para realizar captura de pantalla.

Prueba Kinect

KINECT for Windows

Práctica 2 Kinect



Información del Movimiento:

¡¡¡EJERCICIO TERMINADO!!!

Reiniciar Movimiento

Tolerancia = 10%

Activar Esqueleto

Repeticiones = 1

+1 1 -1

Establecer

Duración = 15

Capturar Imagen

Pulsar el botón para realizar captura de pantalla.

Bugs y Errores Detectados:

En ciertas condiciones de iluminación hay que aumentar la tolerancia hasta un nivel significativo para que detecte bien el movimiento, aunque no suele haber problema en ese aspecto si estamos en una habitación bien iluminada. En esta detección también influye de forma significativa la ropa que lleve el usuario, lo bien que se distinga el fondo e incluso el color de la piel. Pese a todo, el funcionamiento es bastante aceptable.

Se han encontrado algunas dificultades a la hora de pintar el esqueleto centrado, incluso después de mapear correctamente con la función adecuada del SDK los puntos de esqueleto a color. Sin embargo, tras algunas pruebas y cambios, la posición en la que se dibuja es bastante acertada.