Práctica 1- MS KINECT

Detección de posiciones y movimientos básicos

Autores:

- Miguel Sánchez Maldonado
  Correo: miguesanchez181@gmail.com
  Github: https://github.com/msmaldonado
- Cristina Zuheros Montes.
  Correo: zuhe18@gmail.com
  Github: https://github.com/cristinazuhe

Fecha de realización: 30/10/2015

ÍNDICE:

1- Descripción del problema que se aborda.

Nuestro objetivo será la detección de un movimiento de yoga, en particular, del movimiento que se puede ver en la imagen 1.1:

Fig1.1: final

Fig1.2: inicial

Para ello se tendrá que hacer primero un movimiento que llamaremos inicial (Fig1.2) que será simplemente elevar los brazos dejando las piernas relajadas (pero sin juntarlas) para concluir con el movimiento final que se ve en la imagen 1.1. Cabe destacar que para realizar correctamente el movimiento final, la pierna que se debe de levantar es la izquierda.

Para para asegurar de que el movimiento se hace de forma intencionada y no por casualidad, vamos a exigir al usuario que mantenga las posiciones inicial y final unos segundos.

2- Descripción de la solución que se aborda.

2.1- Primer contacto con la aplicación.

En primer lugar comenzamos trabajando únicamente con Skeleton Basics-WPF basándonos en el código de Javier Moreno (https://github.com/jmoreno).

Este código consistía en ser capaces de colocar los brazos hacia delante rectos. Cuando el usuario conseguía hacer este gesto, los brazos cambiaban de color para indicar que era correcto.

Siguiendo este esquema, nosotros fuimos capaces de hacer la detección del movimiento inicial y final:

- Movimiento inicial:

  Controlábamos las articulaciones de los brazos (manos, muñecas, codos y hombros) y mediante distintos colores indicábamos al usuario cuáles eran las articulaciones en posición incorrecta:

  ➢ Color verde: la articulación está en posición correcta.
  ➢ Color azul: los brazos están mal colocados en el eje x. Intenta abrir o cerrar los brazos para llegar a la correcta.
  ➢ Color naranja: los brazos están mal colocados en el eje z, es decir, en profundidad. Intenta acercar los brazos a ti mismo.
  ➢ Color morado: los brazos están hacia abajo. Intenta colocarlos por encima de tus hombros.

Destacar que para todas estas restricciones del movimiento, llevan consigo una tolerancia para que todo tipo de usuario pueda realizar el movimiento sin necesidad de hacer excesivos esfuerzos.

Una función llamada CheckPoints (Fig2.1.1) que irá devolviendo distintos enteros que representarán a los colores (lo iremos haciendo con bloques if que detectarán en todo momento como se encuentran las articulaciones), los cuales representados mediante la función getPen (Fig2.1.2) a la que a cada entero le asigna un color definido anteriormente. CheckPoints(Fig2.1.3) se ayuda de CheckArms para ir sacando las distancias que hay entre dos articulaciones adyacentes del vector.

```
private int checkPoints(SkeletonPoint P1, SkeletonPoint P2)
{
    //brazos arriba...
    if (P1.Y > P2.Y)
        return 2;

    //brazos en profundidad correcta...
    if (diffabs(P1.Z, P2.Z) > 2*ERROR)
        return 1;//rojo

    //En X...
    //hombro-codo
    if(P1 == skeleton.Joints[jointsTypes[0]].Position && P2 == skeleton.Joints[jointsTypes[1]].Position && diff(P1.X, P2.X) > ERROR)
        return -1;
    if (P1 == skeleton.Joints[jointsTypes[4]].Position && P2 == skeleton.Joints[jointsTypes[5]].Position && diff(P2.X, P1.X) > ERROR)
        return -1;

    //codo-muñeca
    if (P1 == skeleton.Joints[jointsTypes[1]].Position && P2 == skeleton.Joints[jointsTypes[2]].Position && P1.X > P2.X - 0.05)
        return -1;
    if (P1 == skeleton.Joints[jointsTypes[5]].Position && P2 == skeleton.Joints[jointsTypes[6]].Position && P2.X > P1.X - 0.05)
        return -1;

    //muñeca-mano
    if (P1 == skeleton.Joints[jointsTypes[2]].Position && P2 == skeleton.Joints[jointsTypes[3]].Position && diff(P1.X, P2.X) > 0.5*ERROR)
        return -1;
    if (P1 == skeleton.Joints[jointsTypes[6]].Position && P2 == skeleton.Joints[jointsTypes[7]].Position && diff(P2.X, P1.X) > 0.5*ERROR)
        return -1;

    else
            return 0;
}
```

Fig 2.1.1

```
public Pen getPen(JointType j)
{
    if (jointsTypes.Contains(j))
    {
        int i = jointsTypes.IndexOf(j);
        if (diff_positions[i] == 2)
            return failBonePenPurple;
        else if (diff_positions[i] == 1)
            return failBonePenRed;
        else if (diff_positions[i] == -1)
            return failBonePenBlue;
    }
    return trackedBonePen;
}
```

Fig2.1.2

Fig2.1.3

- Movimiento final: A la lista creada de articulaciones para el movimiento inicial, añadimos las articulaciones de las piernas (pie, talón, rodillas y caderas). Para detectar el movimiento, partimos del movimiento inicial y le añadimos condiciones sobre las piernas (empezamos trabajando con el movimiento posible de las piernas tanto derecha como izquierda). Siguiendo el esquema anterior mediante bloques if y salidas de enteros controlábamos los colores para indicar como llegar a la posición final:
  - ➢ Colores anteriores.
  - ➢ Color morado: de nuevo usamos este color para indicar que la pierna se debe levantar hacia arriba desplazándose en el eje x, pero sin extenderse demasiado en el eje z (seguimos permitiendo cierta tolerancia para la comodidad del usuario)

Al código anterior del método CheckArms le añadimos las restricciones sobre piernas:

```
if (P1 == skeleton.Joints[jointsTypes[9]].Position && P2 == skeleton.Joints[jointsTypes[10]].Position && diffabs(P1.X, P2.X) > 0.5 * ERROR)
    return 2;

else
    return 0;
```

En ambos movimientos, hacemos uso de la lista de articulaciones que se ve definida en la siguiente imagen:

```
private static List<JointType> jointsTypes = new List<JointType> { JointType.ShoulderLeft, JointType.ElbowLeft, JointType.WristLeft, JointType.HandLeft,
                                                                   JointType.ShoulderRight, JointType.ElbowRight, JointType.WristRight, JointType.HandRight,
                                                                   JointType.HipLeft, JointType.KneeLeft, JointType.AnkleLeft, JointType.FootLeft,
                                                                   JointType.HipRight, JointType.KneeRight, JointType.AnkleRight, JointType.FootRight};


/// <summary>
/// Lista de tamaño igual al número de JoinType que hay que comprobar,
/// en esta lista se almacenan los valores de error de cada posición.
/// </summary>
private List<int> diff_positions = new List<int>(new int[jointsTypes.Count]);
```

Llegados a este punto, ya se nos reconocían ambos movimientos perfectamente por separado.

2.2- Enlazando ambos movimientos.

Nuestro objetivo era realizar el movimiento inicial y, una vez realizado, se nos indique realizar el movimiento final. Haciendo uso de un booleano principal para distinguir si estamos en el movimiento inicial o final, y con booleanos auxiliares para las articulaciones los cuales se activan cuando este su articulación correspondiente correcta. Una vez que todos estén a true indica que hemos realizado dicho movimiento y podemos cambiar el booleano principal para entrar al siguiente movimiento.

Como el movimiento inicial forma parte de nuestro movimiento final, al realizar el movimiento final correctamente, no se podía volver a realizar el inicial, puesto que ya estaba realizado por nuestras restricciones. Para resolver esto, impusimos condiciones específicas sobre piernas también para el movimiento inicial. En concreto, pedíamos que en la posición inicial las piernas permanecieran en reposo.

Omitimos el código ya que posteriormente vimos que se podía hacer de forma más sencilla.

2.3- Haciendo uso de Skeleton Basics con cámara. Primer contacto.

Para esta implementación nos ayudamos del código de José Arcos Aneas (https://github.com/josearcosaneas).

Este código consistía en realizar un movimiento de abertura de manos en cruz, posteriormente levantarlas rectas por encima de los hombre y finalmente dejarlas en posición de reposo, todo esto ayudado por esferas indicándonos en todo momento la posición que debíamos de tener. Para pasar de un movimiento a otro, va haciendo uso de una variable llamada fase, la cual irá cambiando de valor cada vez que se realiza un movimiento correcto.

Olvidando un poco el código trabajado anteriormente basado en colores, buscamos la detección de nuestro movimiento inicial y movimiento final. En lugar de tener el gesto en cruz y el gesto de brazos arriba, declaramos el inicial y el final, para ello en lugar de ir declarando

variables para las articulaciones, como hace José Arcos en su práctica, nosotros seguimos con nuestra idea del vector de articulaciones declarado anteriormente. Mantenemos la idea de tener una variable fase la cual puede tomar tres valores, según estemos en el movimiento inicial, en final o se haya acabado el ejercicio.
Para detectar cuando un movimiento es correcto y podemos pasar al siguiente, lo que haremos será comprobar que todas las articulaciones están correctas, y no de forma incorrecta como hacíamos en un principio.

- ➢ Movimiento inicial: Haciendo uso de un sólo if verificamos que el movimiento sea correcto. Es decir, comprobamos que todas las articulaciones estén colocadas de forma correcta. Para ello hemos ido haciendo uso de los bloques if que definimos en el apartado 2.1. Podemos ver el código en la siguiente imagen.

```
public bool PosicionInicial(Skeleton esqueleto)
{

    this.FeedbackTexBlock.Text = "\t Coloquese en la posición inicial";
    bool enPosicion = false;

    if ((esqueleto.Joints[jointsTypes[0]].Position.Y < esqueleto.Joints[jointsTypes[1]].Position.Y) &&
         (esqueleto.Joints[jointsTypes[4]].Position.Y < esqueleto.Joints[jointsTypes[5]].Position.Y) &&
         (diff(esqueleto.Joints[jointsTypes[0]].Position.X, esqueleto.Joints[jointsTypes[1]].Position.X) < ERROR) &&
         (diff(esqueleto.Joints[jointsTypes[5]].Position.X, esqueleto.Joints[jointsTypes[4]].Position.X) < ERROR) &&
         (esqueleto.Joints[jointsTypes[1]].Position.X < (esqueleto.Joints[jointsTypes[2]].Position.X - 0.05)) &&
         (esqueleto.Joints[jointsTypes[6]].Position.X < (esqueleto.Joints[jointsTypes[5]].Position.X - 0.05)) &&
         (diff(esqueleto.Joints[jointsTypes[2]].Position.X, esqueleto.Joints[jointsTypes[3]].Position.X) < 0.5 * ERROR) &&
         (diff(esqueleto.Joints[jointsTypes[7]].Position.X, esqueleto.Joints[jointsTypes[6]].Position.X) < 0.5 * ERROR) &&
         (diff(esqueleto.Joints[jointsTypes[10]].Position.X, esqueleto.Joints[jointsTypes[9]].Position.X) < 0.02 * ERROR) &&
         (diff(esqueleto.Joints[jointsTypes[13]].Position.X, esqueleto.Joints[jointsTypes[14]].Position.X) < 0.02 * ERROR))
    {
        this.FeedbackTexBlock.Text = "\t Bien hecho . ";
        enPosicion = true;
        this.repeticiones++;
        this.FeedbackRepes.Text = " " + this.repeticiones;
        this.fase = 1;

    }
    else
    {
        this.FeedbackTexBlock.Text = "\t Mal. No esta en la posicion inicial.";
        enPosicion = false;
    }
    return enPosicion;
}
```

➢ Movimiento final: Del mismo modo que el movimiento inicial, añadimos un bloque if que nos asegure que el movimiento es correcto incorporando las condiciones de piernas que hemos comentando antes. El código queda del siguiente modo:

```
public bool PosicionFinal(Skeleton esqueleto)
{
    this.FeedbackTexBlock.Text = "\t Pongase en la posicion final";
    bool enPosicion = false;

    // dependiendo de la distancia el resultado sera true o false.
    if ((esqueleto.Joints[jointsTypes[0]].Position.Y < esqueleto.Joints[jointsTypes[1]].Position.Y) &&
                    (esqueleto.Joints[jointsTypes[4]].Position.Y < esqueleto.Joints[jointsTypes[5]].Position.Y) &&

                    (diff(esqueleto.Joints[jointsTypes[0]].Position.X, esqueleto.Joints[jointsTypes[1]].Position.X) < ERROR) &&
                    (diff(esqueleto.Joints[jointsTypes[5]].Position.X, esqueleto.Joints[jointsTypes[4]].Position.X) < ERROR) &&

                    (esqueleto.Joints[jointsTypes[1]].Position.X < (esqueleto.Joints[jointsTypes[2]].Position.X - 0.05)) &&
                    (esqueleto.Joints[jointsTypes[6]].Position.X < (esqueleto.Joints[jointsTypes[5]].Position.X - 0.05)) &&

                    (diff(esqueleto.Joints[jointsTypes[2]].Position.X, esqueleto.Joints[jointsTypes[3]].Position.X) < 0.5 * ERROR) &&
                    (diff(esqueleto.Joints[jointsTypes[7]].Position.X, esqueleto.Joints[jointsTypes[6]].Position.X) < 0.5 * ERROR) &&

                    (diff(esqueleto.Joints[jointsTypes[10]].Position.X, esqueleto.Joints[jointsTypes[9]].Position.X) > ERROR))
    {

        enPosicion = true;
        this.fase = 2;
    }
    else
    {
        this.FeedbackTexBlock.Text = "\t No esta en la posicion final.";
        enPosicion = false;
    }
    return enPosicion;
}
```

2.3- Skeleton Basics con cámara: Articulaciones con color.

Una vez que ya tenemos que se nos reconozca bien cuando hacemos el movimiento inicial, que se sea capaz de pasar al movimiento final y que éste también se reconozca bien, finalizando el ejercicio, vamos a hacerlo un poco más intuitivo. Para ello vamos a hacer uso de nuestro código visto en 2.1 para añadir color a los huesos en caso de que alguna esté mal colocada.

Seguimos manteniendo las funciones Check (nuestra antigua CheckArms) y getPen. Modificamos la función CheckPoints, para que dependiendo de la fase nos pinte los colores de forma correcta. Quedaría de la siguiente forma:

```
private int checkPosicion(SkeletonPoint P1, SkeletonPoint P2, Skeleton esqueleto)
{
    if (fase == 0)        {
        //brazos en profundidad correcta...
        if (diffabs(P1.Z, P2.Z) > 2 * ERROR)
            return 1;//naranja
        //brazos arriba...
        if (P1 == esqueleto.Joints[jointsTypes[0]].Position && P2 == esqueleto.Joints[jointsTypes[1]].Position && P1.Y > P2.Y)
            return 2;
        else if (P1 == esqueleto.Joints[jointsTypes[4]].Position && P2 == esqueleto.Joints[jointsTypes[5]].Position && P1.Y > P2.Y)
            return 2;
        //hombro-codo
        else if (P1 == esqueleto.Joints[jointsTypes[0]].Position && P2 == esqueleto.Joints[jointsTypes[1]].Position && diff(P1.X, P2.X) > ERROR)
            return -1;
        else if (P1 == esqueleto.Joints[jointsTypes[4]].Position && P2 == esqueleto.Joints[jointsTypes[5]].Position && diff(P2.X, P1.X) > ERROR)
            return -1;
        //codo-muñeca
        else if (P1 == esqueleto.Joints[jointsTypes[1]].Position && P2 == esqueleto.Joints[jointsTypes[2]].Position && P1.X > P2.X - 0.05)
            return -1;
        else if (P1 == esqueleto.Joints[jointsTypes[5]].Position && P2 == esqueleto.Joints[jointsTypes[6]].Position && P2.X > P1.X - 0.05)
            return -1;
        //muñeca-mano
        else if (P1 == esqueleto.Joints[jointsTypes[2]].Position && P2 == esqueleto.Joints[jointsTypes[3]].Position && diff(P1.X, P2.X) > 0.5 * ERROR)
            return -1;
        else if (P1 == esqueleto.Joints[jointsTypes[6]].Position && P2 == esqueleto.Joints[jointsTypes[7]].Position && diff(P2.X, P1.X) > 0.5 * ERROR)
            return -1;
        else
            return 0;
    }
    else if (fase == 1)    {
        if (diffabs(P1.Z, P2.Z) > 2 * ERROR)
            return 1;//naranja
        //brazos arriba...
```

```
        //brazos arriba...
        if (P1 == esqueleto.Joints[jointsTypes[0]].Position && P2 == esqueleto.Joints[jointsTypes[1]].Position && P1.Y > P2.Y)
            return 2;
        else if (P1 == esqueleto.Joints[jointsTypes[4]].Position && P2 == esqueleto.Joints[jointsTypes[5]].Position && P1.Y > P2.Y)
            return 2;
        //hombro-codo
        else if (P1 == esqueleto.Joints[jointsTypes[0]].Position && P2 == esqueleto.Joints[jointsTypes[1]].Position && diff(P1.X, P2.X) > ERROR)
            return -1;
        else if (P1 == esqueleto.Joints[jointsTypes[4]].Position && P2 == esqueleto.Joints[jointsTypes[5]].Position && diff(P2.X, P1.X) > ERROR)
            return -1;
        //codo-muñeca
        else if (P1 == esqueleto.Joints[jointsTypes[1]].Position && P2 == esqueleto.Joints[jointsTypes[2]].Position && P1.X > P2.X - 0.05)
            return -1;
        else if (P1 == esqueleto.Joints[jointsTypes[5]].Position && P2 == esqueleto.Joints[jointsTypes[6]].Position && P2.X > P1.X - 0.05)
            return -1;
        //muñeca-mano
        else if (P1 == esqueleto.Joints[jointsTypes[2]].Position && P2 == esqueleto.Joints[jointsTypes[3]].Position && diff(P1.X, P2.X) > 0.5 * ERROR)
            return -1;
        else if (P1 == esqueleto.Joints[jointsTypes[6]].Position && P2 == esqueleto.Joints[jointsTypes[7]].Position && diff(P2.X, P1.X) > 0.5 * ERROR)
            return -1;
        //posicion correcta de la pierna.
        else if (P1 == esqueleto.Joints[jointsTypes[9]].Position && P2 == esqueleto.Joints[jointsTypes[10]].Position && diff(P2.X, P1.X) < ERROR)
            return 2;
        else
            return 0;

    }
    else return 0;
```

2.4- Skeleton Basics con cámara: Añadimos esferas de guía.

Una vez que sabemos sabemos realizar el movimiento de forma correcta, vamos a darle unas guías al usuario mediante esferas para facilitarle de forma visual el tipo de movimiento a realizar.

Para ello declaramos un método pintaBolas para cada uno de los movimientos.

- ➢ Esferas para posición inicial (Fig2.4.1): vamos a pintar 3 esferas todas pintadas en rojo.
  Una situada por encima de la cabeza a una distancia equivalente a la longitud del codo-mano del usuario.
  Otra situada a la altura de la cabeza desplazada en el eje x la longitud del hombro-codo del usuario.
  Una tercera análoga al anterior pero en el otro hombro.
- ➢ Esferas para posición final (Fig2.4.2): vamos a pintar 4 esferas. Las 3 anteriores estarán en el mismo sitio pero pintadas en color naranja. La cuarta está situada a la altura de la rodilla izquierda desplazada en el eje x la longitud del rodilla-pie del usuario. También en color naranja.
- ➢ Esferas para ejercicio acabado: pintamos las 4 esferas anteriores en el mismo sitio pero en color verde para indicar que el movimiento está acabado correctamente.

Para trabajar con estas longitudes de extremidad a extremidad que usamos, creamos una función (Fig2.4.3) que en todo momento vaya calculándolas del siguiente modo: para cada dos extremidades que nos interesan, vemos cuál es su longitud mayor de hueso, y ésta será la longitud real de esas dos extremidades. De esta manera, las bolas se colocan a medida del tamaño del usuario.

```
private void pintaBolasInicial(Skeleton esqueleto, DrawingContext drawingContext)
{

    SkeletonPoint manosarriba = new SkeletonPoint();
    SkeletonPoint codod = new SkeletonPoint();
    SkeletonPoint codoi = new SkeletonPoint();

    //La X sera la distancia euclidea entre la muñeca y el hombro,
    // en el primer caso y el codo y el hombro en el segundo
    manosarriba.X = esqueleto.Joints[JointType.ShoulderCenter].Position.X;
    manosarriba.Y = esqueleto.Joints[JointType.ShoulderCenter].Position.Y + codoMano;
    manosarriba.Z = esqueleto.Joints[JointType.ShoulderCenter].Position.Z;


    codod.X = esqueleto.Joints[JointType.ShoulderCenter].Position.X + hombroCodo;
    codod.Y = esqueleto.Joints[JointType.ShoulderCenter].Position.Y + 0.05F;
    codod.Z = esqueleto.Joints[JointType.ShoulderCenter].Position.Z;


    codoi.X = esqueleto.Joints[JointType.ShoulderCenter].Position.X - hombroCodo;
    codoi.Y = esqueleto.Joints[JointType.ShoulderCenter].Position.Y + 0.05F;
    codoi.Z = esqueleto.Joints[JointType.ShoulderCenter].Position.Z;

    Point pos1 = this.SkeletonPointToScreen(manosarriba);
    drawingContext.DrawEllipse(Brushes.Red, null, pos1, 10, 10);
    Point pos2 = this.SkeletonPointToScreen(codod);
    drawingContext.DrawEllipse(Brushes.Red, null, pos2, 10, 10);
    Point pos3 = this.SkeletonPointToScreen(codoi);
    drawingContext.DrawEllipse(Brushes.Red, null, pos3, 10, 10);
```

Fig 2.4.1

```
private void pintaBolasFinal(Skeleton esqueleto, DrawingContext drawingContext)  {
    SkeletonPoint manosarriba = new SkeletonPoint();
    SkeletonPoint codod = new SkeletonPoint();
    SkeletonPoint codoi = new SkeletonPoint();
    SkeletonPoint rodillaizquierda = new SkeletonPoint();

    manosarriba.X = esqueleto.Joints[JointType.ShoulderCenter].Position.X;
    manosarriba.Y = esqueleto.Joints[JointType.ShoulderCenter].Position.Y + codoMano;
    manosarriba.Z = esqueleto.Joints[JointType.ShoulderCenter].Position.Z;

    codod.X = esqueleto.Joints[JointType.ShoulderCenter].Position.X + hombroCodo;
    codod.Y = esqueleto.Joints[JointType.ShoulderCenter].Position.Y +0.05F;
    codod.Z = esqueleto.Joints[JointType.ShoulderCenter].Position.Z;

    codoi.X = esqueleto.Joints[JointType.ShoulderCenter].Position.X - hombroCodo;
    codoi.Y = esqueleto.Joints[JointType.ShoulderCenter].Position.Y + 0.05F;
    codoi.Z = esqueleto.Joints[JointType.ShoulderCenter].Position.Z;

    rodillaizquierda.X = esqueleto.Joints[JointType.KneeRight].Position.X - rodillaPie;
    rodillaizquierda.Y = esqueleto.Joints[JointType.KneeRight].Position.Y + 0.1F;
    rodillaizquierda.Z = esqueleto.Joints[JointType.KneeRight].Position.Z;

    Point pos1 = this.SkeletonPointToScreen(manosarriba);
    drawingContext.DrawEllipse(Brushes.Orange, null, pos1, 10, 10);
    Point pos2 = this.SkeletonPointToScreen(codod);
    drawingContext.DrawEllipse(Brushes.Orange, null, pos2, 10, 10);
    Point pos3 = this.SkeletonPointToScreen(codoi);
    drawingContext.DrawEllipse(Brushes.Orange, null, pos3, 10, 10);

    Point pos4 = this.SkeletonPointToScreen(rodillaizquierda);
    drawingContext.DrawEllipse(Brushes.Orange, null, pos4, 10, 10);
}
```

Fig 2.4.2

```
public void TomaMedidas(Skeleton esqueleto)
{
    if (hombroCodo < diffabs(esqueleto.Joints[jointsTypes[0]].Position.X, esqueleto.Joints[jointsTypes[1]].Position.X))
        hombroCodo = diffabs(esqueleto.Joints[jointsTypes[0]].Position.X, esqueleto.Joints[jointsTypes[1]].Position.X);
    if (hombroCodo < diffabs(esqueleto.Joints[jointsTypes[0]].Position.Y, esqueleto.Joints[jointsTypes[1]].Position.Y))
        hombroCodo = diffabs(esqueleto.Joints[jointsTypes[0]].Position.Y, esqueleto.Joints[jointsTypes[1]].Position.Y);
    if (hombroCodo < diffabs(esqueleto.Joints[jointsTypes[0]].Position.Z, esqueleto.Joints[jointsTypes[1]].Position.Z))
        hombroCodo = diffabs(esqueleto.Joints[jointsTypes[0]].Position.Z, esqueleto.Joints[jointsTypes[1]].Position.Z);

    if (codoMano < diffabs(esqueleto.Joints[jointsTypes[1]].Position.X, esqueleto.Joints[jointsTypes[2]].Position.X))
        codoMano = diffabs(esqueleto.Joints[jointsTypes[1]].Position.X, esqueleto.Joints[jointsTypes[2]].Position.X);
    if (codoMano < diffabs(esqueleto.Joints[jointsTypes[1]].Position.Y, esqueleto.Joints[jointsTypes[2]].Position.Y))
        codoMano = diffabs(esqueleto.Joints[jointsTypes[1]].Position.Y, esqueleto.Joints[jointsTypes[2]].Position.Y);
    if (codoMano < diffabs(esqueleto.Joints[jointsTypes[1]].Position.Z, esqueleto.Joints[jointsTypes[2]].Position.Z))
        codoMano = diffabs(esqueleto.Joints[jointsTypes[1]].Position.Z, esqueleto.Joints[jointsTypes[2]].Position.Z);


    if (caderaRodilla < diffabs(esqueleto.Joints[jointsTypes[8]].Position.X, esqueleto.Joints[jointsTypes[9]].Position.X))
        caderaRodilla = diffabs(esqueleto.Joints[jointsTypes[8]].Position.X, esqueleto.Joints[jointsTypes[9]].Position.X);
    if (caderaRodilla < diffabs(esqueleto.Joints[jointsTypes[8]].Position.Y, esqueleto.Joints[jointsTypes[9]].Position.Y))
        caderaRodilla = diffabs(esqueleto.Joints[jointsTypes[8]].Position.Y, esqueleto.Joints[jointsTypes[9]].Position.Y);
    if (caderaRodilla < diffabs(esqueleto.Joints[jointsTypes[8]].Position.Z, esqueleto.Joints[jointsTypes[9]].Position.Z))
        caderaRodilla = diffabs(esqueleto.Joints[jointsTypes[8]].Position.Z, esqueleto.Joints[jointsTypes[9]].Position.Z);


    if (rodillaPie < diffabs(esqueleto.Joints[jointsTypes[9]].Position.X, esqueleto.Joints[jointsTypes[10]].Position.X))
        rodillaPie = diffabs(esqueleto.Joints[jointsTypes[9]].Position.X, esqueleto.Joints[jointsTypes[10]].Position.X);
    if (rodillaPie < diffabs(esqueleto.Joints[jointsTypes[9]].Position.Y, esqueleto.Joints[jointsTypes[10]].Position.Y))
        rodillaPie = diffabs(esqueleto.Joints[jointsTypes[9]].Position.Y, esqueleto.Joints[jointsTypes[10]].Position.Y);
    if (rodillaPie < diffabs(esqueleto.Joints[jointsTypes[9]].Position.Z, esqueleto.Joints[jointsTypes[10]].Position.Z))
        rodillaPie = diffabs(esqueleto.Joints[jointsTypes[9]].Position.Z, esqueleto.Joints[jointsTypes[10]].Position.Z);
}
```

Fig 2.4.3

2.5- Skeleton Basics con cámara:Imágenes de guía.

Para hacer nuestra aplicación más intuitiva, hemos añadido a la interfaz imágenes que representen el movimiento inicial y el final que se deben de realizar. Las imágenes son las que vimos en el apartado 1.
Para incorporarlas añadimos al archivo MainWindow.xaml el siguiente código:

```
<DockPanel Grid.Row="2" Margin="0,0,0,0" Grid.ColumnSpan="2">
    <Image DockPanel.Dock="Left" Source="Images\inicial.png" Stretch="Fill" Height="300" Width="150" Margin="0 10 0 5"/>
    <TextBlock DockPanel.Dock="Bottom" Margin="0 0 -1 0" VerticalAlignment="Bottom" FontSize="25">inicial</TextBlock>
</DockPanel>

<DockPanel Grid.Row="3" Margin="0,0,0,0" Grid.ColumnSpan="3">
    <Image DockPanel.Dock="Left" Source="Images\final.png" Stretch="Fill" Height="300" Width="150" Margin="0 10 0 5"/>
    <TextBlock DockPanel.Dock="Bottom" Margin="0 0 -1 0" VerticalAlignment="Bottom" FontSize="25">final</TextBlock>
</DockPanel>
```

Además se ha incorporado mensajes de salida en los que se le va indicando en todo momento al usuario el movimiento que tiene que realizar y si lo está realizando de forma correcta.
El resultado visual lo podremos ver en el siguiente apartado.

3- Errores frecuentes o aspectos destacados.

Uno de los aspectos que hay que destacar es detectar que el movimiento se está realizando correctamente y no se ha realizado por casualidad, y para ello lo que hacemos es pedirle al usuario que se mantenga en la posición un determinado tiempo.

Para detectar que el usuario está en una posición fija en determinado tiempo usamos frame, (1 segundo =30 frame).El concepto de frame se comentó en clase por varios compañeros e investigamos para abordar el problema.

Vamos a comprobar si estoy en la posición indicada, si estamos en la posición correcta y es la primera vez que entra en el método lo indica booleano y actualiza el frame al frame actual, si seguimos en la posición correcta ya la siguientes veces que entra comprueba si el frame actual es mayor y ponemos el tiempo que queramos. Si en algún momento nos movemos de la posición correcta pondremos el booleano a false de esta manera cuando hagamos el movimiento correcto se vuelve a actualizar el frame para así tener que mantenernos en la posición el tiempo que se había considerado desde un principio. Es importante tener en cuenta que hay que usar el booleano ya que en caso de no mantenerse en la posición correcta, vuelver a reiniciar el frame.De no ser así podríamos realizar multiples repeticiones del movimiento correcto sin matenernos en la posición correcta y que nos lo diera por bueno. Podemos ver el código en (Fig 3.1)

```
        (esqueleto.Joints[jointsTypes[1]].Position.X < (esqueleto.Joints[jointsTypes[2]].Position.X - 0.05)) &&
        (esqueleto.Joints[jointsTypes[6]].Position.X < (esqueleto.Joints[jointsTypes[5]].Position.X - 0.05)) &&
        (diff(esqueleto.Joints[jointsTypes[2]].Position.X, esqueleto.Joints[jointsTypes[3]].Position.X) < 0.5 * ERROR) &&
        (diff(esqueleto.Joints[jointsTypes[7]].Position.X, esqueleto.Joints[jointsTypes[6]].Position.X) < 0.5 * ERROR) &&
        (diff(esqueleto.Joints[jointsTypes[10]].Position.X, esqueleto.Joints[jointsTypes[9]].Position.X) < 0.02 * ERROR) &&
        (diff(esqueleto.Joints[jointsTypes[13]].Position.X, esqueleto.Joints[jointsTypes[14]].Position.X) < 0.02 * ERROR))
    {
        this.FeedbackTexBlock.Text = "\t Bien hecho. Manténgase. ";
        if(contador == false)
        {
            contador = true;
            frame_aux = frame;
        }
        else
        {
            if (frame_aux + 45 < frame)
            {
                this.fase = 1;
                contador = false;
            }
        }

        enPosicion = true;

    }
    else
    {
        if (contador == true)
           contador = false;

        this.FeedbackTexBlock.Text = "\t No esta en la posicion inicial.";
        enPosicion = false;
    }
    return enPosicion;
}
```

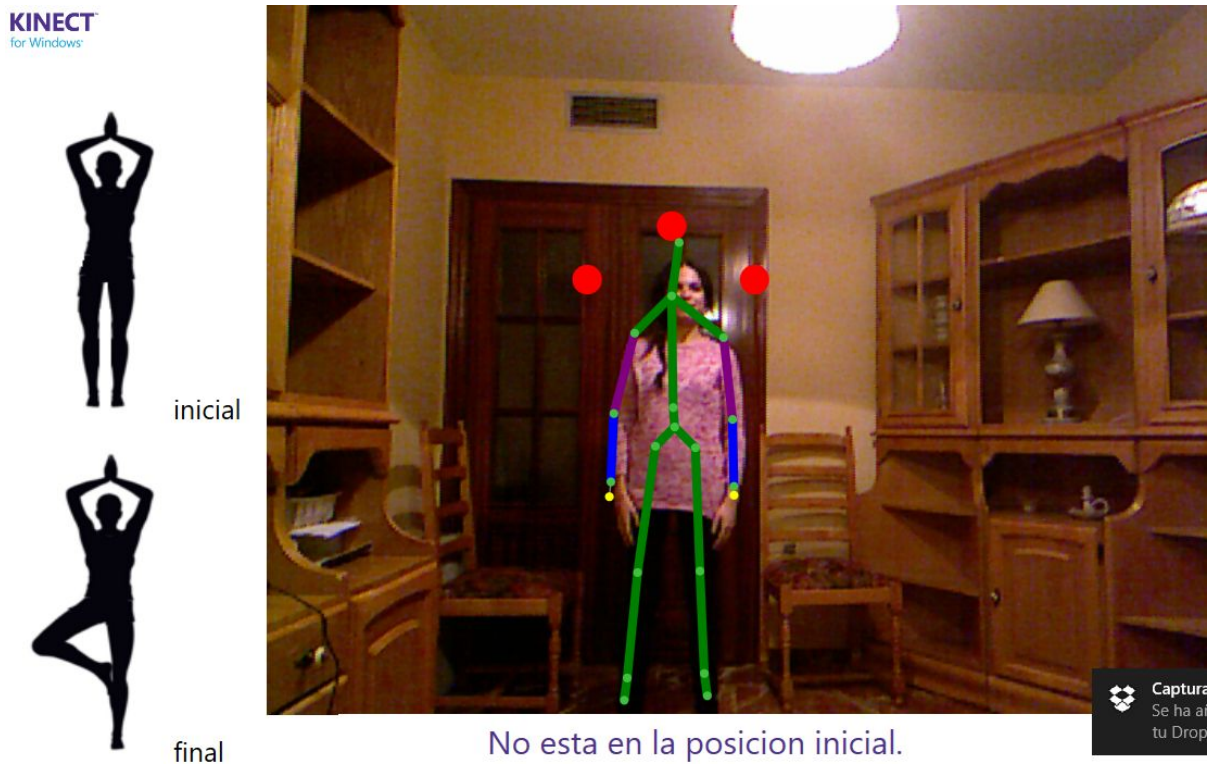
Fig 3.1

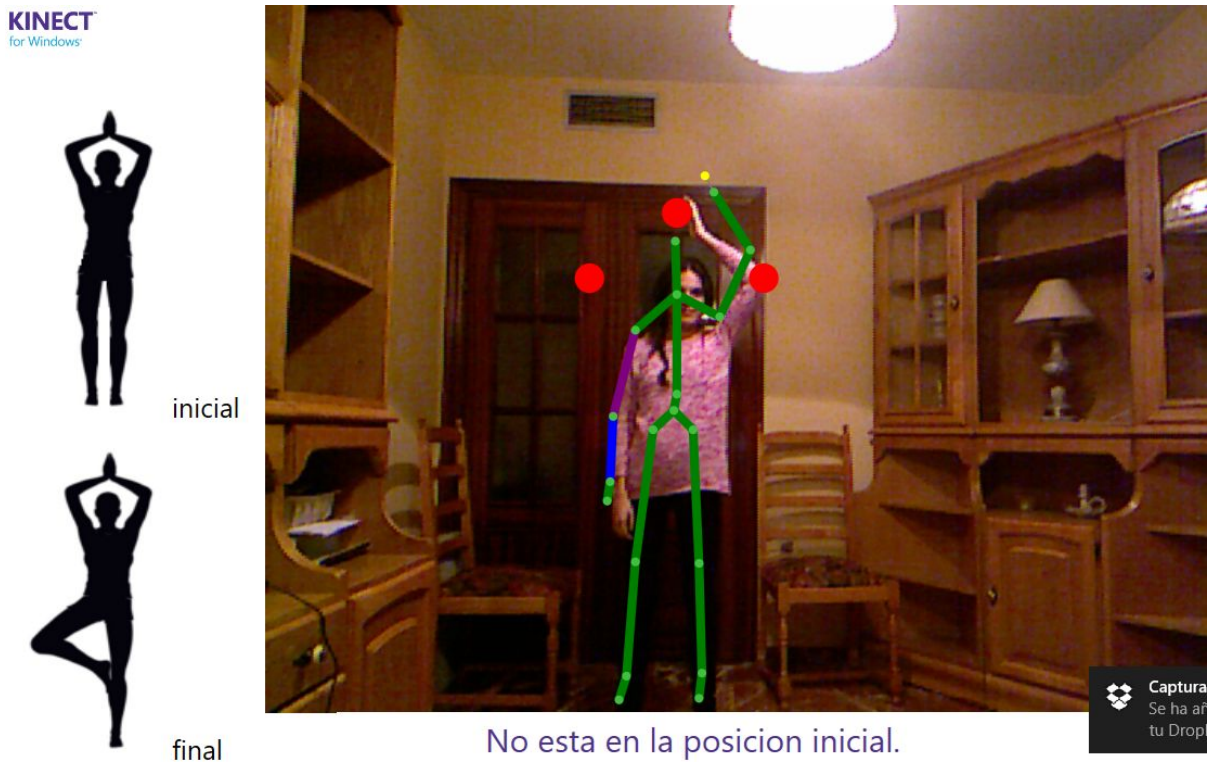
4- Probando la aplicación.

En este apartado vamos a ver el correcto funcionamiento de nuestra aplicación. En resumen de todo lo explicado anteriormente, lo que el usuario tendrá que hacer será:

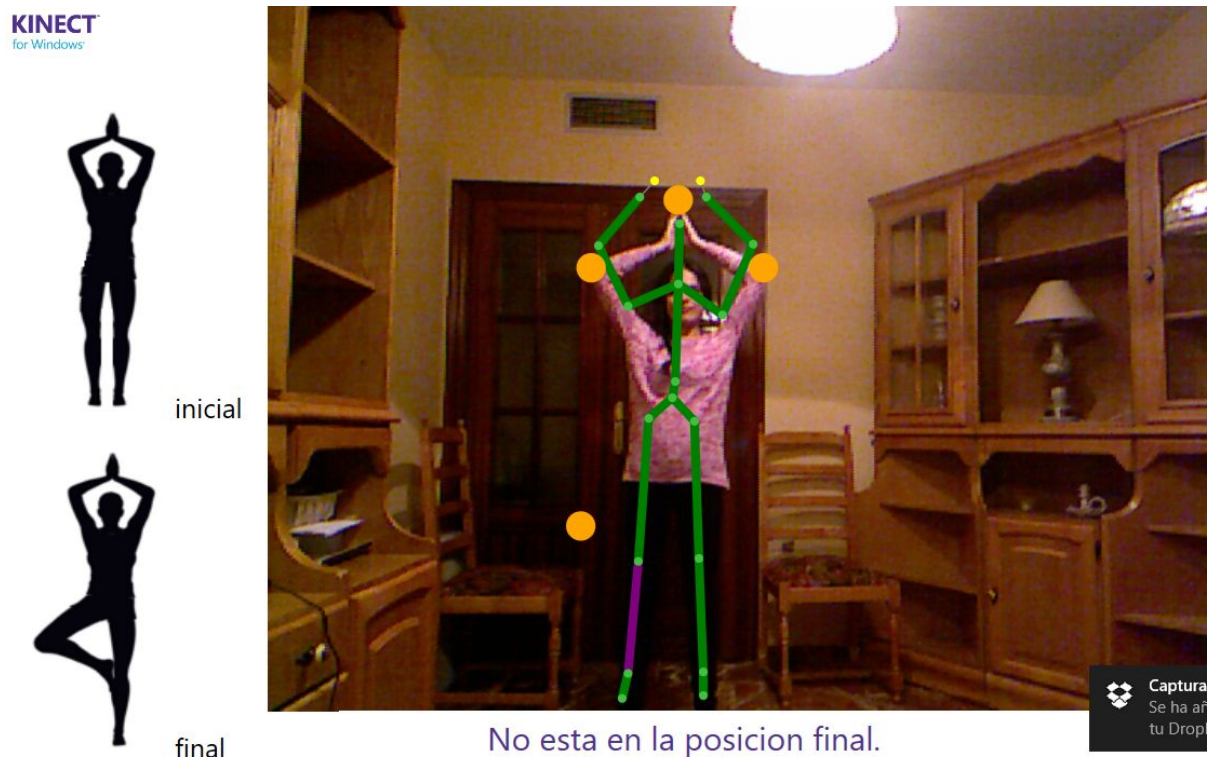
- ➢ Colocarse en la posición de Inicio, colocando los brazos donde indican las bolas rojas.
- ➢ Aguantar segundo y medio para asegurar que el movimiento es a posta.
- ➢ Colocarse en la posición final, colocando los brazos y las piernas donde indican las bolas naranja.
- ➢ Aguantar segundo y media para asegurar que el movimiento es a posta.
- ➢ El ejercicio termina cuando todas las bolas y el esqueleto están en verde.

Veámoslo:

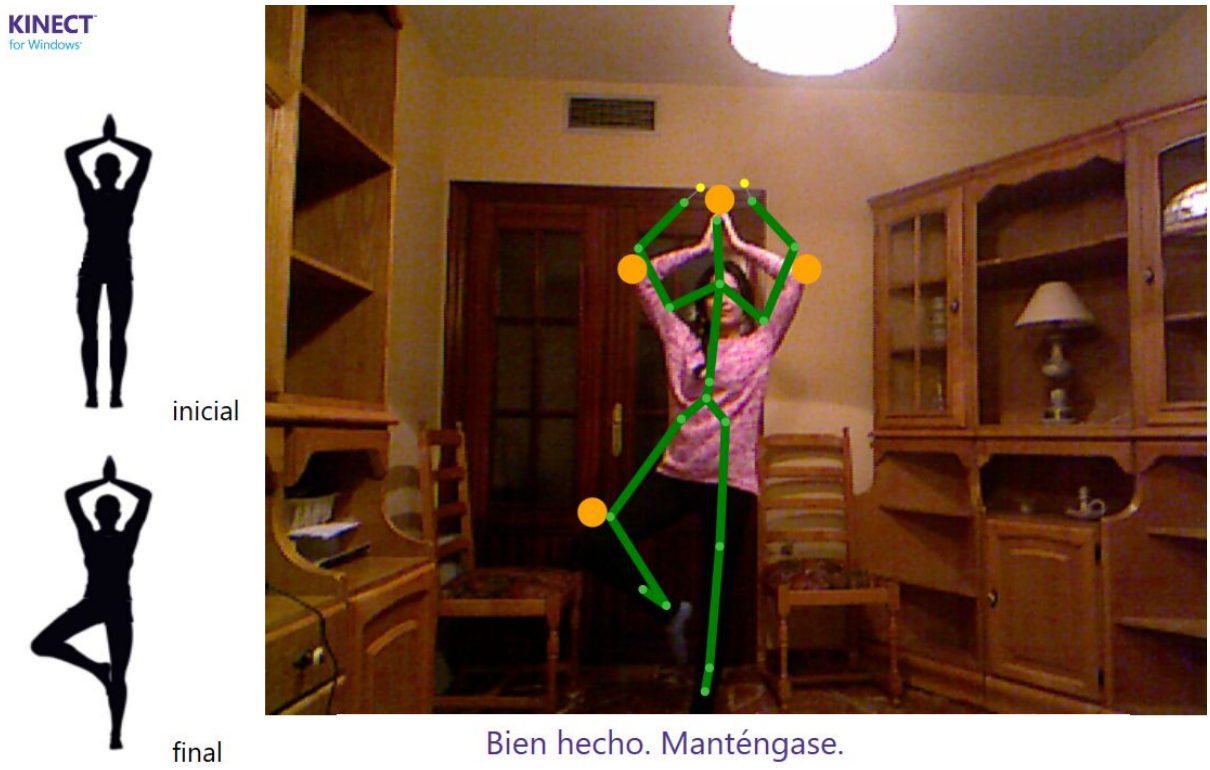




Aunque un brazo y las piernas estén correctas, el movimiento no es válido. Necesitamos colocar el otro brazo.



Hemos superado los segundos de la posición inicial y nos piden que nos coloquemos en la posición final. Como la pierna no está en posición incorrecta, se marca en color morado.



Una vez hecho el movimiento final, nos piden que nos mantengamos, al igual que ocurre con el movimiento inicial.

El movimiento es final, y el ejercicio finaliza.

5- Lecturas recomendadas y referencias.

Como ya hemos indicado nos basamos en:

- Ejemplos C# de Kinect for Windows SDK.
- Práctica del curso pasado de Javi Moreno.
- Práctica del curso pasado de José Arcos.