Práctica 2-

MS KINECT

Interacción gestual con kinect

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: 30/11/2015

ÍNDICE:

1- Descripción del problema que se aborda.

Nuestro objetivo será la creación de un juego de preguntas en las que el usuario tendrá que interaccionar a través de kinect para marcar la solución correcta.

El juego comenzará solicitando al usuario que elija un nivel de dificultad y posteriormente elegir un tema de una lista de temas (hemos incorporado los siguientes temas: Arte, literatura, deporte, ciencias, historia y geografía). Una vez elegido el tema y la dificultad, comenzarán las preguntas. Cada vez que nos aparece una pregunta el usuario debe pulsar una de las 3 posibles soluciones y al lado aparecerá un icono indicando si es la correcta o no. En caso de ser correcta el contador que muestra el resultado de aciertos se aumentará, después de cada pregunta el usuario debe pulsar el botón de siguiente pregunta que se activará al contestar la pregunta anterior. Después de cinco preguntas el programa vuelve a empezar y por tanto vuelve a pedir un nivel de dificultad, elegir un tema y vuelven a comenzar las preguntas.

2- Descripción de la solución que se aborda.

Para el proyecto hemos usado el ejemplo ControlBasics-WPF que nos ofrece kinect, el cuál nos permite movernos por botones y seleccionar alguno de ellos haciendo un impulso hacia delante con la mano.

Para adaptarlo a nuestro juego, lo primero fué crear una lista de botones con los temas en cuestión. Creamos el método KinectTileButtonClick para establecer las funcionalidades de estos botones. Veámoslo:

```
private void KinectTileButtonClick(object sender, RoutedEventArgs e)
{

    var button = (KinectTileButton)e.OriginalSource;
    if (tema == null && dificultad != null)
    {
        tema = (button.Label as string);
        this.Imagen_pregunta.Source = new BitmapImage(new Uri(System.IO.Path.GetFullPath("S_PREGUNTA.png")));
        this.fondo.Visibility = Visibility.Visible;
        this.fondo.Source = new BitmapImage(new Uri(System.IO.Path.GetFullPath(tema + ".png")));
    }
    e.Handled = true;
}
```

Si aún no tenemos un tema de la lista de temas seleccionado, pero sí que tenemos una dificultad establecida por nosotros mismo, haremos que el tema de trabajo sea aquél que corresponda a la etiqueta que tiene el botón que pulsamos. Ya tendremos el tema realmente asignado.

Además, establecemos el fondo del juego con la imagen del tema, para que el usuario sea consciente en todo momento del tema en el que está. Una vez elegido un tema, no se podrá cambiar hasta que finalice la ronda de 5 preguntas, con la dificultad ocurre igual que el tema. Esta restricción se tiene en cuenta ya que cuando seleccionamos un tema se activa el botón de fórmula pregunta indicándonos que pulsemos para que empiecen las preguntas, por tanto no tendría sentido que se pulsara otro tema.
Asimismo, en la zona de preguntas, se muestra una imagen sobre la que el usuario tendrá que pulsar para que vayan apareciendo las preguntas. De este modo, puede tomarse el tiempo necesario que considere para prepararse para jugar.

Al pulsar una respuesta, que corresponde a KinectTileButtonClick2 nos almacenará en un variable la opción,(una vez que seleccionamos un respuesta no nos permite cambiar de respuesta, ya que al pulsar automáticamente comprueba si esa respuesta es correcta). Veamos con más detalle el código:

```
private void KinectTileButtonClick2(object sender, RoutedEventArgs e)       {
    var button = (KinectTileButton)e.OriginalSource;
    if (tema == null && dificultad == null && opcion == null)           {
        dificultad = (button.Label as string);
        this.Imagen_pregunta.Source = new BitmapImage(new Uri(System.IO.Path.GetFullPath("S_Tema.png")));
    }
    else if (tema != null && opcion == null && dificultad != null && !primera)           {
        opcion = (button.Label as string);
        comprobarPregunta();

        this.Imagen_pregunta.Source = new BitmapImage(new Uri(System.IO.Path.GetFullPath("S_Pregunta.png")));
        if (n_pregunta >= 5 && tema != null && dificultad != null)              {
            this.Imagen_pregunta.Source = new BitmapImage(new Uri(System.IO.Path.GetFullPath("DIFICULTAD.png")));
            this.fondo.Visibility = Visibility.Hidden;
            tema = null;
            dificultad = null;
            n_correctas = 0;
            this.ContadorResultados.Text = n_correctas + "/5";
            n_pregunta = 1;
            this.correcto.Visibility = Visibility.Hidden;
            opcion = null;
            primera = true;
        }
    }
    e.Handled = true;
}
```

Vemos que primero hace una comprobación de si el tema es null ya que de está manera lo que nos estará dando los botones ABC será una dificultad y no un respuesta a un pregunta. Por tanto si se trata de dificultad, almacenamos el valor de la dificultad y pedimos que se seleccione un tema, en otro caso almacenamos opción (A, B o C) y llamamos a la función Comprobarpregunta() la cuál nos dirá si la respuesta es correcta. Este método vamos a omitirlo por su excesiva extensión, pero básicamente hace lo siguiente:
Vemos la dificultad con la que estamos trabajando, seguidamente vemos el tema de trabajo, y por último vemos la pregunta que se hace al usuario. Si la respuesta del usuario (su opción) corresponde con la opción correcta de la pregunta, se mostrará un tick de que la respuesta es correcta y se contabiliza una respuesta correcta más en un contador

de respuestas acertadas. Dicho contador, se irá actualizando cada vez que respondamos a una pregunta y cuando tengamos hecha la ronda de 5 preguntas, se pondrá de nuevo a 0. En caso contrario, se mostrará un tick de que la respuesta es incorrecta. Se ve más claro con un ejemplo, así que vamos a ver un trozo de código:

```
else if(dificultad == "B")
{
    if (tema == "DEPORTES")
    {
        if (n_pregunta == 1)
        {
            if (opcion == "B")
            {
                n_correctas++;
                this.ContadorResultados.Text = n_correctas + "/5";
                this.correcto.Visibility = Visibility.Visible;
                this.correcto.Source = new BitmapImage(new Uri(System.IO.Path.GetFullPath("correcto.png")));
            }
            else
            {
                this.correcto.Visibility = Visibility.Visible;
                this.correcto.Source = new BitmapImage(new Uri(System.IO.Path.GetFullPath("incorrecto.png")));

            }

        }
        else if (n_pregunta == 2)
```

Retomando con KinectTileButtonClick2, por último tiene en cuenta si se trata de la respuesta a la última pregunta, para de esta manera “reiniciar el programa” pone los contadores a valores iniciales y las variables al valor inicial, quita el fondo y nos pone que seleccionemos una dificultad. y volvería a comenzar el programa.

Estas son las funcionalidades de los botones, ahora vamos a ver las características de los mismos. Para ello creamos otros dos métodos, veamos botón para los temas:

Lo que hacemos es establecer una imagen a cada tema, es decir, a cada uno de los botones de la lista de temas. Al botón, le podemos el nombre del tema, para que sea más claro para el usuario. Establecemos el tamaño de la letra, así como el tamaño de los botones.
Para los botones A, B y C creamos un método análogo que vamos a omitir pues es realmente similar, simplemente vamos cambiando los tamaños, textos e imágenes a nuestra conveniencia.

Para crear un botón con alguno de los temas hacemos lo siguiente:

```
var button = crearBoton("HISTORIA");
this.wrapPanel.Children.Add(button);
```

y para las opciones A, B, C, sería análogo:

```
var button2 = crearBoton2("A");
this.wrapPanel2.Children.Add(button2);
```

Y por último creamos el método FormulaPregunta() que se lanza cuando pulsamos para que nos salga una pregunta. Si tenemos un tema seleccionado, una dificultad y va a ser la primera pregunta se nos muestra, en otro caso estamos en cualquier otra pregunta, aumentamos el contador el número de preguntas , mostramos la siguiente pregunta y ocultamos el icono de correcto o incorrecto de la pregunta anterior. Además ponemos la opción a null para esperar una nueva respuesta del usuario.

```
private void FormulaPregunta(object sender, RoutedEventArgs e)
{
    var button = (KinectTileButton)e.OriginalSource;

    if (tema != null && primera && dificultad != null)
    {
        this.Imagen_pregunta.Visibility = Visibility.Visible;

        this.Imagen_pregunta.Source = new BitmapImage(new Uri(System.IO.Path.GetFullPath(tema + n_pregunta + dificultad + ".png")));
        primera = false;
    }
    else if (n_pregunta < 5 && tema != null && opcion != null && dificultad != null)
    {
        n_pregunta++;
        this.Imagen_pregunta.Source = new BitmapImage(new Uri(System.IO.Path.GetFullPath(tema + n_pregunta + dificultad + ".png")));
        this.correcto.Visibility = Visibility.Hidden;
        opcion = null;

    }

    e.Handled = true;
}
```

3- Errores frecuentes o aspectos destacados.

Como aspectos destacados podemos ver dos:

3.1 Fusionar botón de correcto e incorrecto.

Al principio en la aplicación teníamos dos botones, uno que mostraba la imagen de correcto si la respuesta era acertada, o el otro que mostraba el icono de incorrecto si la respuesta era errónea. Lo que hicimos fue dejar un único botón con una imagen y en el método comprobarPregunta() dependiendo si se acierta o no la pregunta manda la imagen adecuada al botón(correcta o incorrecta)

3.2 Especificar la respuesta correcta.

Otro aspecto que se tuvo en cuenta que al final no se usó pero que creemos importante de considerar fue el que apareciese tras pulsar una respuesta cuál era la correcta. Pero al ser un programa “corto” ,ya que solo hay 5 preguntas por nivel y tema las cuales aparecen en el mismo orden tanto de pregunta como de respuestas, si nos muestra la respuesta correcta rápidamente sabríamos todas las respuestas y por tanto no tendría sentido seguir jugando. De este modo si hay preguntas que fallas puedes volver luego a escoger el mismo tema y dificultad para poder jugar sin saber cuál era la respuesta correcta.

Como errores podemos ver:

3.3 Cerrar la aplicación.

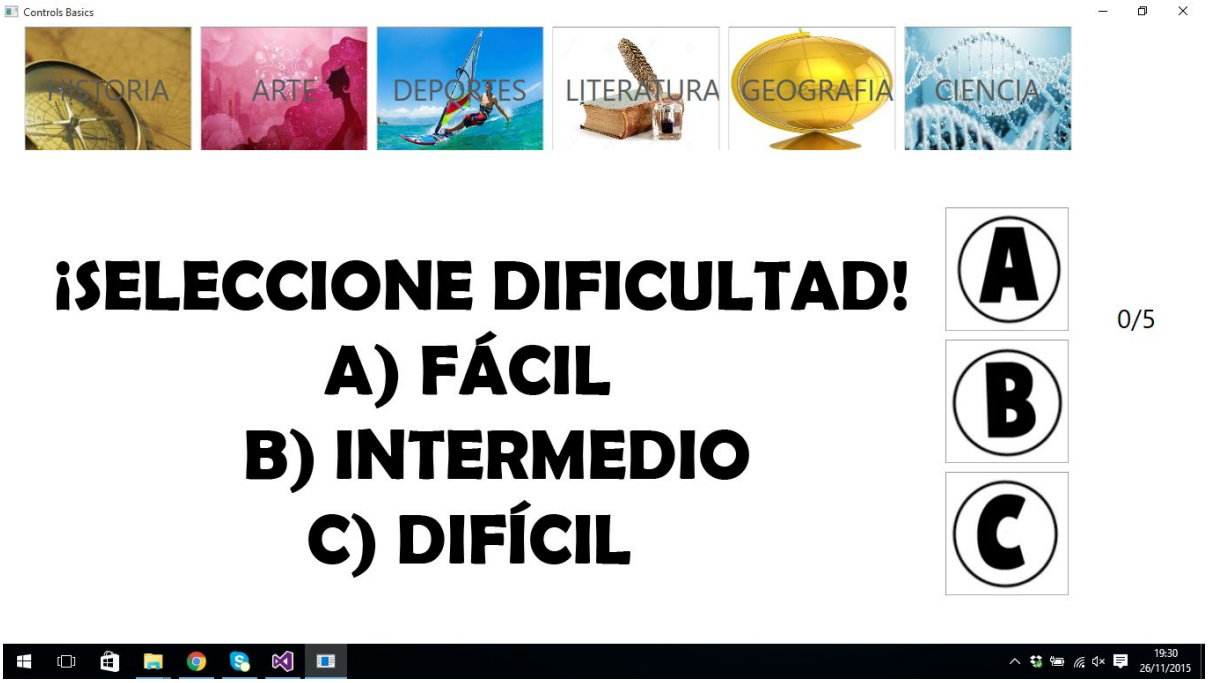
Intentamos colocar un botón para finalizar la aplicación y que se cerrara. Para ello usamos windowsClossing pero se cerraba porque daba violación de segmento al igual que a varios compañeros. Por tanto finalmente se decidió no usar ese botón ya que no se consiguió arreglar ni averiguar a qué era debido el problema. Para esta función nos basamos en el siguiente método.

```
private void Window_Closed(object sender, EventArgs e)
{
    // Turn off timers.
    RefreshTimer.IsEnabled = false;
    RefreshTimer.Stop();
    UpdateTimer.IsEnabled = false;
    UpdateTimer.Stop();
    // Turn off Kinect
    if (this.mainKinect != null)
    {
        try
        {
            this.mainKinect.Stop();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
        this.TxtBx_KinectStatus.Text += "\n[" + DateTime.Now.TimeOfDay.ToString() + "] " +
        + this.mainKinect.UniqueKinectId.ToString() + " has been turned off.";
    }

    // Shut down application
    Application.Current.Shutdown();
}
```

4- Probando la aplicación.

Comenzamos el juego y nos encontramos con la siguiente pantalla:



Seleccionamos un nivel de dificultad, A B o C y nos pide que seleccionemos un tema:

Ahora nos obliga a seleccionar el tema que queramos, con la dificultad que nos hemos auto asignado. En nuestro caso, seleccionamos literatura y vemos la siguiente pantalla:

Como podemos ver, el fondo es acorde a nuestra elección y nos pide que pulsemos para empezar a jugar. Impulsamos con la mano y vemos la pregunta:

Tenemos todo el tiempo que queramos para pensarlo, la idea es aprender y no vernos presionados. Una vez tengamos clara la respuesta, seleccionamos la opción. En nuestro caso seleccionamos la opción C:

Se nos indica que la respuesta es correcta, y se nos indica que pulsemos para continuar aprendiendo. Una vez acabadas las 5 preguntas de la ronda, volvemos a la imagen inicial para elegir dificultad y, posteriormente, tema.

5- Lecturas recomendadas y referencias.

Como ya hemos indicado nos basamos en:

- Ejemplos C# de Kinect for Windows SDK.
- Juego Atríviate para conseguir más preguntas.