Programación Visual en C#.NET

Guion de la práctica 2

OBJETIVOS

- Asignar funcionalidad a los botones del ratón.
- Modificar el aspecto de las aplicaciones
- Aprender a utilizar cajas de diálogo estándar

TEMPORIZACIÓN

Recogida del enunciado:	Semana del 27 de septiembre
Desarrollo de la práctica:	**1 semana**
Fecha de entrega:	Semana del 25 de octubre junto con las prácticas 1, 3 y 4
Fecha límite de entrega:	Semana del 1 de noviembre

PRÁCTICA 2

Fuentes y color, cajas de diálogo estándar

TABLA DE CONTENIDOS:

Introducción

Se continuará trabajando con el programa desarrollado en la práctica 1.

Como ampliación al apartado 2.11, se mejorará la interfaz de usuario en lo que respecta a cambiar la hora con el ratón. El botón derecho servirá para arrastrar la manecilla de los minutos y se recurrirá al estado de la tecla *Ctrl* para que el botón izquierdo adquiera la funcionalidad que antes tenía el botón derecho.

En el punto 3 se modificará el aspecto del reloj digital. En primer lugar, se cambiará la fuente del texto. Se mostrará la existencia de distintas unidades de medida (puntos, píxeles…). En segundo lugar, se cambiará el color del texto y del fondo.

En el punto 4 se usarán algunas cajas de diálogo estándar. Mediante estas cajas de diálogo, que el sistema proporciona ya implementadas, el usuario podrá escoger el color de la fuente de una paleta de colores, así como el tipo de fuente.

En el punto 5, se sustituirán los botones por opciones de menú. Se comprobará que este cambio no afecta significativamente al funcionamiento del programa.

2.12 Ajustar los minutos

Se pretende refinar la interfaz de usuario para que sea más fácil cambiar la hora del reloj. Actualmente resulta muy difícil ajustar los minutos. Se usará para ello el botón derecho del ratón. La funcionalidad actual del botón derecho será proporcionada por la tecla *Ctrl* más el botón izquierdo. Recapitulando, la funcionalidad del ratón deberá ser:

Eventos del ratón	*Acción*
Pulsación/arrastre con el botón **izquierdo**	Ajuste **de horas y minutos**. **PM** si tecla *Ctrl* pulsada; **AM** en caso contrario
Pulsación/arrastre con el botón **derecho**	Ajuste de **minutos.**

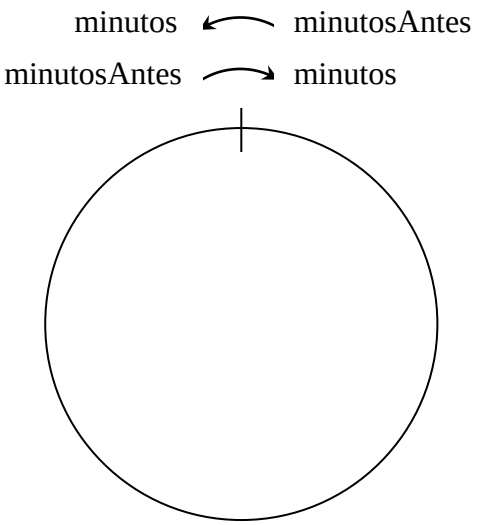
Pista: En la biblioteca Windows.Forms es posible conocer el estado (pulsado/no pulsado) de cualquier tecla modificadora utilizando la propiedad estática `ModifierKeys` de la clase `Control`, o bien los eventos `KeyDown/KeyUp` del `control`.

Cuando el usuario esté arrastrando la aguja de los minutos, y dicha aguja pase por la marca de las 12, deberán incrementarse o disminuirse, según corresponda, las horas. El código listado a continuación da una pista de cómo puede conseguirse este efecto:

```
minutosAntes = ...;  // <--Minutos antes de la actualización actual
minutos = ...;       // <--Calcular según la posición del ratón

if (minutos > minutosAntes &&          // Si pasamos por las 12
       minutos - minutosAntes > 30)    // hacia atrás...
   horas--;                            // ...una hora menos
else if (minutos < minutosAntes &&
       minutosAntes - minutos > 30)    // Y hacia delante...
   horas++;                            // ...una hora más

if (horas == -1)
   horas = 23;
else if (horas == 24)
   horas = 0;
```



El código de la función arrastrar agujas queda como sigue, rellene las sentencias que faltan.

```
private void ArrastrarAgujas(object sender, MouseEventArgs e)
{
  bool botonDerIzq = e.Button == MouseButtons.Right ||
                     e.Button == MouseButtons.Left;
  bool pulsadoCentro = (e.X == m_Centro.X && e.Y == m_Centro.Y);

  if (m_Radio < 10 || !botonDerIzq || pulsadoCentro)
     return;

  double alfa;
  int horas = m_Hora.Hour, minutos=m_Hora.Minute;
  int minutosAntes = m_Hora.Minute;

  alfa = Math.Atan2(m_Centro.X - e.X, e.Y - m_Centro.Y);

  // Si está pulsado el botón izquierdo del ratón
  if (.....................)
  {
    minutos = (int)((12 * alfa / (2 * Math.PI) + 6) * 60);
    horas = (minutos / 60) % 12;
    minutos %= 60;

    //Si esta pulsada la tecla control pasar a PM sumando 12 horas
    if (.....................)
       horas += 12;
   }

   // Si está pulsado el botón derecho del ratón
   if (.....................)
   {
     // Calcular según la posición del ratón
     minutos = (int)(60 * alfa / (2 * Math.PI) + 30) % 60;
     if (minutos > minutosAntes &&          // Si pasamos por las 12
         minutos - minutosAntes > 30)      // hacia atrás...
       horas--;                             // ...una hora menos
     else if (minutos < minutosAntes &&
              minutosAntes - minutos > 30)     // Y hacia delante...
       horas++;
                       // ...una hora más
        /* 30 es un valor para conocer que nos estamos moviendo
           alrededor de las 12 (podría ser 5, 50 u otro).
           Alrededor de las 12 las diferencias van a ser de 58 o 59
           entre cada dos eventos MouseMove consecutivos.
           Alrededor de las 3 la diferencia sería de 0 o 1.*/

     if (horas == -1)
       horas = 23;
     else if (horas == 24)
       horas = 0;
   }

   /*Actualice la hora del reloj digital de acuerdo a los cambios
     realizados. Utilice para ello un objeto del reloj digital

   ...................
   ...................

}
```

3. Fuentes y color

3.1 Fuentes

Añada en el constructor de la clase `RelojDigital`, antes de la llamada a la función `MostrarHoraActual`, la siguiente línea de código:

```
ct_HoraActual.Font =
        new Font("Times New Roman", 16, FontStyle.Regular);
```

Observe que la caja de texto se redimensiona automáticamente para adaptarse a la nueva fuente.

Compilar y ejecutar para ver el resultado.

Por defecto, el constructor de la clase `Font` interpreta en puntos el tamaño suministrado. Esta unidad es la más habitual, y es la que se utiliza cuando se especifica un tamaño de fuente en un editor de textos como Microsoft Word. Sin embargo, en ocasiones nos puede interesar especificar el tamaño en otras unidades, como por ejemplo en píxeles. Para ello podemos emplear una forma alternativa de dicho constructor:

```
ct_HoraActual.Font = new Font("Times New Roman",16,
                             FontStyle.Regular, GraphicsUnit.Pixel);
```

Compilar y ejecutar para ver el resultado.

Modifique la línea anterior para obtener una letra Arial de 12 puntos en negrita.

Compilar y ejecutar para ver el resultado.

3.2 Color

Añada al constructor de la clase `RelojDigital`, tras la línea introducida anteriormente:

```
ct_HoraActual.BackColor = Color.White;
```

Modifique el color del texto de la caja para que sea azul, sabiendo que la propiedad que se encarga de esta tarea es **ForeColor**.

Compilar y ejecutar para ver el resultado.

Nota**:** Las propiedades **Font**, **ForeColor** y **BackColor** de la caja de texto pueden modificarse también a través de la ventana de propiedades de la misma. Compruebe que, al hacerlo así, el código correspondiente se añade en el método `InitializeComponent` del archivo "RelojDigital.Designer.cs".

4. Cajas de diálogo estándar

La biblioteca System.Windows.Forms pone a nuestra disposición las siguientes clases de diálogo estándar:

Clase	***Propósito***
`OpenFileDialog`	Abrir un fichero
`SaveFileDialog`	Guardar un fichero
`ColorDialog`	Seleccionar un color
`FontDialog`	Seleccionar un tipo de letra
`PageSetupDialog`	Establecer las características de la página a imprimir
`PrintDialog`	Imprimir un documento

Las cajas de diálogo estándar se suelen usar como se indica en el siguiente fragmento de código:

```
private void bt_Prueba_Click(object sender, EventArgs e)
{
   OpenFileDialog dlgAbrir = new OpenFileDialog();

   //... Inicialización del diálogo

   if (dlgAbrir.ShowDialog() == DialogResult.OK)
   {
     // ... Extracción de los datos introducidos por el usuario
   }
}
```

Siguiendo el esquema del ejemplo anterior, se deberá añadir al programa la funcionalidad necesaria para que el usuario pueda cambiar el color de fondo de la caja de texto `ct_HoraActual` del reloj digital, así como el tipo de letra y su color. Para ello se añadirán dos botones “Color fondo” y “Fuente” al formulario.

Observe en el código anterior que el diálogo debe mostrar inicialmente los valores actuales.

5. Menús

Sustituya todos los botones de la ventana principal por los elementos de un menú "Opciones" que incluya una orden por cada botón eliminado. La opción "Mostrar analógico" deberá estar marcada (*checked*) cuando el reloj analógico esté visible. Para ello, maneje el evento **DropDownOpened** del menú "Opciones".