1. INTRODUCCIÓN Y OBJETIVOS

Esta práctica finaliza el uso de controles en aplicaciones basadas en formularios y pretende utilizar algunas de las características de dibujo disponibles en .NET Compact Framework.

2. TAREAS DE LA PRÁCTICA

2.1. AMPLIACIÓN DE LA APLICACIÓN DE MAPAS

- Debe modificarse la forma de calibrar los mapas, haciendo desaparecer la pestaña de calibración. Para calibrar, será necesario usar la opción del menú principal, que hará que aparezca el mapa seleccionado en la pestaña de mapas y comience el proceso de calibración sobre el mapa. Se mostrará el mapa con su título y una barra de estado irá guiando el proceso de calibración, pidiendo que se hagan los siguientes pasos:

    - Pulsar sobre el mapa en la posición de P1.

    - Pulsar sobre el mapa en la posición de P2. P2 deberá estar en la horizontal de P1, es decir, tendrá una coordenada Y igual a la de P1. Como puede ser realmente difícil hacer clic justo en la horizontal de P1, admitiremos cualquier pulsación y haremos su componente Y igual a la de P1.

    - Introducir, mediante un *TextBox* y un botón que aparecerán en pantalla, la distancia en metros entre P1 y P2. El botón servirá para indicar que hemos introducido un dato válido en el *TextBox*. Si el dato es un número mayor que cero, tanto el *TextBox* como el botón desaparecerán, y el valor se considerará válido. Debería aparecer un *InputPanel* en cuanto se muestra en pantalla el *TextBox*.

    - Pulsar sobre el mapa en la posición de P3. P2 deberá estar en la vertical de P1, es decir, tendrá una coordenada X igual a la de P1. Actuaremos de forma análoga a como se hizo con P2, pero con la coordenada X.

    - Introducir, mediante un *TextBox* y un botón, la distancia en metros entre P2 y P3. El comportamiento de ambos será análogo al comentado anteriormente para obtener la distancia entre P1 y P2.

- Después de esto, el mapa esperará dos pulsaciones sobre la pantalla y se indicará la distancia entre ambos puntos, igual que se hacía en la versión anterior del programa.

2.2. USO DE LAS FUNCIONES DE DIBUJO EN PANTALLA

La siguiente modificación consiste en utilizar un objeto *System.Drawing.Graphics* en lugar de un *PictureBox*. *System.Drawing.Graphics* es una de las clases alrededor de la cual gira *GDI+*, el API de dibujo sobre dispositivos gráficos e impresoras de la plataforma Win32 y sus variantes.

El objeto *Graphics* en nuestra aplicación se creará usando el método *CreateGraphics()* de la clase *Form*. Para dibujar el mapa, se utilizará el método *Graphics.DrawImage()*. Existen muchas versiones sobrecargadas de este método, pudiendo utilizarse la que recibe como primer parámetro un objeto *Image* (*ImageList. Images[i]*) y dos enteros con las coordenadas a

partir de los que se dibujará. Ambos parámetros serán 0 para dibujar a partir de la esquina superior izquierda de la pantalla.

Una vez se muestra el mapa mediante el objeto *Graphics*, podría mejorarse la aplicación introduciendo las siguientes mejoras:

- Según se vayan marcando los puntos P1, P2 y P3 durante la calibración, se dibujará un pequeño punto (círculo de radio no muy grande) en la posición donde está el punto. En el caso de P2 y P3, el punto no se dibuja exactamente en las coordenadas de cada una de las pulsaciones, sino en las posiciones finales calculadas para P2 y P3 (justo en la horizontal y en la vertical de P1, respectivamente). Además de los puntos, se dibujará junto a cada uno de ellos el texto P1, P2 y P3, según corresponda. Para implementar todo esto, habrá que utilizar los métodos *Graphics.FillEllipse()* y *Graphics.DrawString()*. Utiliza la ayuda de Visual Studio .NET 2003 para conocer cómo funcionan estos métodos y qué significado tienen sus parámetros.

- Los puntos P1, P2 y P3 aparecerán siempre dibujados en pantalla y la barra de estado (*StatusBar*) mostrará siempre las distancias P1-P2 y P2-P3 en metros.

- A la hora de calcular la distancia entre dos puntos, cuando se pulse sobre el mapa para determinar la posición del primer punto, se dibujará en pantalla el mismo y, junto a él, el texto X1. Lo mismo se hará con el segundo punto (X2). Recuerda que, si pasa un determinado tiempo desde la pulsación que determina la posición de X1, habrá que rechazar ese punto. Esto implica repintar toda la pantalla sin el círculo y el texto de X1. Una vez pulsadas las posiciones de X1 y X2, se dibujará una línea en pantalla de X1 a X2 y, junto a su punto medio, un texto que indique la distancia en metros entre ambos.

- Si se vuelve a seleccionar otro punto, se repintará la pantalla sin los antiguos X1 y X2 y con el nuevo punto X1.

- En cualquier momento, se podrá seleccionar una nueva calibración.

2.3. ACTUALIZACIÓN DE UN GRAPHICS: EL EVENTO PAINT

Si mientras estuvieramos visualizando un objeto *Graphics* en pantalla, se mostrase un *MessageBox* ocultando parte de la superficie del *Graphics*, al cerrar ese *MessageBox* veríamos esa parte del objeto *Graphics* vacía. Esto se debe a que, cuando cualquier otra ventana oculta todo o parte de un *Graphics*, hay que repintar explícitamente la zona que quedó oculta. Siempre que esto ocurra, el objeto *Form* recibirá un evento *Paint*, siendo labor del programador la implementación de un manejador de este evento.

Es posible, según el alumno haya programado la aplicación que, incluso, a la hora de ordenar el dibujo del mapa en pantalla, éste no aparezca. Comprueba cómo, en cualquiera de los dos casos (aparece el mapa pero se muestra un *MessageBox* y, tras cerrarlo, falta la parte del mapa que el *MessageBox* ocultaba; o, directamente, no aparece el mapa en pantalla), la aplicación muestra correctamente el mapa al implementar el manejador del evento *Paint*.

Mientras que para los controles típicos (*Label*, *TextBox*, …) están definidas por defecto las

acciones a efectuar cuando hay que repintar, en el caso de un nuevo tipo de control creado a medida o de una superficie gestionada mediante un objeto *Graphics*, estas acciones son desconocidas para el sistema operativo. Esto es completamente lógico, puesto que un objeto de esta clase podría mostrar cualquier imagen cargada desde fichero o dibujada directamente por el programador, y sólo éste sabrá qué es lo que pretende mostrar en pantalla.

Debe tenerse en cuenta que, en esta aplicación, el evento *Paint* se podrá recibir tanto cuando estemos mostrando la pestaña de selección de mapas como cuando estemos mostrando el mismo (calibrando, seleccionando la posición de X1 y X2, etc.). Es labor del programador determinar, por lo tanto, si se está mostrando el objeto *Graphics* con el mapa y el resto de elementos[1], siendo necesario entonces repintar, o la pestaña de mapas (para la cual no es necesario repintar nada).

Haz doble clic sobre el evento *Paint* del formulario de la aplicación e implementa el código necesario del nuevo método *Form1_Paint()*:

```
private void Form1_Paint(object sender,
                                    System.Windows.Forms.PaintEventArgs e)
{
   if (this.tabControl1.Visible==false)
      this.dibujarObjetos();
      // Será necesario conocer en qué estado está el programa y qué
      // elementos hay que dibujar y cuáles no en dibujarObjetos()
}
```

Podría optimizarse la gestión del evento *Paint*, de forma que repinte sólo el rectángulo necesario y no todo el objeto *Graphics*, aunque no se abordará cómo hacerlo, por exceder el propósito de la asignatura.

[1] Los círculos correspondientes a P1, P2, P3, X1 y X2, los textos que indican qué es cada punto, la línea entre X1 y X2, la distancia en metros entre X1 y X2, etc. Es necesario controlar en qué momento de la ejecución se encuentra la aplicación y cuál de estos elementos gráficos hay que dibujar en el instante de recepción del evento Paint.