Parte I

1. INTRODUCCIÓN Y OBJETIVOS

Esta práctica es una primera introducción en el uso de controles en las aplicaciones con interfaz gráfica basada en el uso de formularios. También se pretende que el alumno se familiarice con el IDE de Visual Studio .NET (opciones principales, ayuda dinámica, etc.). La práctica puede ser desarrollada sin problemas mediante el uso del emulador de Pocket PC instalado junto con Visual Studio .NET.

2. TAREAS DE LA PRÁCTICA

2.1. GENERACIÓN DE NÚMEROS ALEATORIOS

- Contruir una aplicación basada en un formulario, que genere números aleatorios:
  - El formulario ofrecerá dos TextBox que indicarán los valores máximo y mínimo que podrá tomar el número aleatorio a generar.
  - También ofrecerá un Button (botón) y una Label (etiqueta); al pulsar sobre el botón, la etiqueta ofrecerá un número generado aleatoriamente.
- Para la generación de números aleatorios, se utilizará la clase System.Random. Deberá crearse una instancia de esta clase y, a continuación, se llamará al método Next() de la clase. Este método genera un nuevo número aleatorio y permite, en algunas de sus versiones sobrecargadas, fijar el rango en el que estará el nuevo número aleatorio a generar.

2.2. USO DE TEMPORIZADORES

- Contruir una aplicación basada en un formulario que incluya un TextBox, una Label, un Timer (que es un control no visible, que representa a un temporizador) y dos Buttons.
- El TextBox indicará el periodo del Timer en milisegundos.
- Cada vez que venza el Timer, la etiqueta avisará al usuario.
- Los dos botones sirven para iniciar y para parar el Timer. Se deberá poder iniciar y parar el temporizador tantas veces como se desee. El objeto Timer ha sido comentado en clase de teoría.

2.3. PROGRAMACIÓN DE UN JUEGO SENCILLO

Se trata de crear una pequeña aplicación que simula el comportamiento del juego *cazabombas*. El juego consiste en un temporizador que controla la cuenta atrás de una bomba (que estallará a los 20 segundos, por ejemplo).

Pasado ese tiempo, la bomba estallará si no ha sido desactivada. Para desactivarla, habrá que pulsar un botón. Esto no será tan sencillo, porque otro timer se encargará de modificar aleatoriamente (cada ½ segundo, por ejemplo) las coordenadas X e Y del botón, haciendo más difícil la pulsación sobre el botón.

Si se logra detener la bomba, pulsando el botón que se va moviendo por la pantalla, aparecerá un MessageBox que mostrará el mensaje “Trabajo Conseguido”. Si pasan los 20 segundos sin que se logre pulsar el botón móvil, aparecerá otro MessageBox indicando el mensaje

"Lástima, fallaste".

Deberá incluirse un botón de inicio del programa que reiniciará los dos Timers y otro más para parar el juego. Evidentemente, en la zona de la pantalla donde estén estos dos botones de inicio y fin de partida, no podrá aparecer el botón móvil durante el juego. También es importante que el botón móvil aparezca por completo en la pantalla y ninguna parte de él se salga del formulario (juega para ello con las dimensiones del botón, esto es, Height y Width).

2.3.1. Ampliaciones

Se puede añadir un menú de programa para añadir una serie de opciones de menú. Esto se hace utilizando el objeto MainMenu (*mainMenu1*) que se incluye por defecto en los proyectos de Aplicaciones para Windows para Dispositivos Smart Device, que hemos estado creando en los ejercicios anteriores.

El MainMenu incorporará dos opciones principales: *General* y *Configuración*. El menú *General* permite *Iniciar* el juego, *Parar*lo y *Salir* de la aplicación. El menú *Configuración* permite seleccionar la *Dificultad*, a través de tres opciones (*Fácil*, *Difícil* y *Misión Imposible*). La selección de cualquiera de estas tres opciones modificará el tiempo de presencia en una misma posición del botón de desactivación de la bomba. También podría modificar las dimensiones de dicho botón.

Las distintas opciones de un MainMenu son muy fáciles de crear, utilizando el diseñador gráfico de formulario de Visual Studio .NET. Haciendo clic sobre el MainMenu, es posible ir añadiendo nuevas opciones; haciendo doble clic sobre una opción, se creará dinámicamente el método que se ejecutará cuando la aplicación detecte un evento de pulsación sobre esa opción de menú.

Otra posible ampliación sería el uso de un control ProgressBar para ir mostrando la evolución de la cuenta atrás de la bomba.

PARTE II

3. INTRODUCCIÓN Y OBJETIVOS

Esta parte es una ampliación del uso de controles en las aplicaciones con interfaz gráfica basada en el uso de formularios. Incorpora el uso de eventos de entrada mediante un dispositivo puntero y la visualización de mapas de bits.

4. TAREAS DE LA PRÁCTICA

4.1. APLICACIÓN SIMPLE PARA NAVEGACIÓN EN UN MAPA

- Hacer un programa que muestre un mapa en la pantalla. El programa se iniciará con una pantalla en blanco y tres opciones en el menú principal (*MainMenu*, siempre existe uno por defecto asociado a todo formulario): *Configurar*, *Salir* y *Ejecutar*. Al seleccionar *Configurar* se ofrece un *TabControl* con las siguientes opciones:

    - En la primera pestaña (*Mapas*) se mostrará un *ComboBox* que ofrecerá uno de los mapas disponibles (la gestión de los mapas a través del ComboBox y de la ImageList se comenta con mayor detalle al final de este guión, en el apartado 4.3). Una vez seleccionado, se hará aparecer el *InputPanel* y sobre un *TextBox* se solicitará un nombre al mapa. Se debe controlar que el SIP ocupa los 80 bits inferiores de la pantalla.

    - La segunda pestaña (*Calibración*) contiene los valores de calibración del mapa. Se deben dar las coordenadas X e Y de tres puntos (P1, P2 y P3) y la distancia en metros que hay entre ellos. P1 se puede modificar, vía NumericUpDown las dos coordenadas. De P2 sólo se puede modificar la coordenada Y, siendo la X la misma de P1 (debe modificarse a la vez). P3 puede modificar únicamente la coordenada X, siendo la Y la misma que la de P1. En la parte inferior de la pantalla debe haber dos TextBox que pregunten la distancia en metros entre P1-P2 y P1-P3.

- Al seleccionar *Ejecutar* se mostrará un *PictureBox* con la imagen del mapa seleccionado cargado en una *ImageList*. El tamaño del mapa es 240x272. Debe de aparecer una etiqueta con el nombre asignado al mapa, a modo de título. Cuando aparece el *PictureBox*, desaparecerá el *TabControl*. Cuando se seleccione de nuevo la opción *Configurar* en el menú principal, desaparecerá el *PictureBox* y aparecerá el *TabControl* con sus pestañas.

4.2. DETECCIÓN DE EVENTOS DE PULSACIÓN DE RATÓN

- Cuando se pulsen dos puntos del mapa, deberá aparecer la distancia (en metros) entre ellos, mediante una etiqueta sobre el mapa.

    - Para conocer las coordenadas de la pantalla de la posición de la pantalla sobre la que se pulsa con el puntero deben detectarse eventos de pulsación de puntero sobre la pantalla. Este evento es el mismo que el de clic de ratón.

    - Dicho evento es el evento *MouseDown* disponible entre los eventos de ratón

del formulario.

- Haciendo doble clic en la entrada *MouseDown* en el cuadro de eventos, asociará un método llamado *Form1_MouseDown* al evento *MouseDown* del Form. En el método *InitializeComponent()* del formulario, aparecerá el siguiente nuevo código:

```
this.MouseDown += new System.Windows.Forms.MouseEventHandler(this.Form1_MouseDown);
```

- Además, se añadirá el esqueleto de ese método al final del código fuente:

```
private void Form1_MouseDown(object sender, System.Windows.Forms.MouseEventArgs e)
        {

        }
```

- Dentro del cuerpo del método *Form1_MouseDown*, el argumento e proporciona en sus miembros *e.X* y *e.Y* las coordenadas X e Y del punto sobre el que se pulsó, dentro del área del formulario.

- Para capturar de forma más correcta la pulsación, en lugar de utilizar el gestor de eventos del formulario, se deberá utilizar el del objeto PictureBox donde se representa el mapa. Podemos modificar los dos trozos de código anteriores cambiando el texto *this.MouseDown* por *this.pictureBox1.MouseDown* y el nombre de método *Form1_MouseDown* por *pictureBox1_MouseDown*. Lamentablemente, el diseñador gráfico de VS.NET 2003 no ofrece la posibilidad de añadir "visualmente" manejadores de eventos de ratón asociados a un PictureBox, aunque éstos pueden ser creados "manualmente".

- La distancia entre dos puntos se debe indicar en metros. Evidentemente, lo que se conocerá directamente es la distancia en pixels (tanto en horizontal como en vertical). Pero en la pestaña de *Calibración* se han introducido las equivalencias de un avance en horizontal o en vertical en pixels a metros. Se deberá conocer la distancia en horizontal y en vertical entre los dos puntos en pixels y pasar cada una de ellas a metros. Finalmente, se calculará la distancia en metros entre ambos puntos mediante la fórmula que obtiene la hipotenusa de un triángulo rectángulo como la suma del cuadradro de sus catetos.

- Tras pulsar sobre el primero de los puntos, un *Timer* deberá controlar si se tarda en pulsar el segundo punto. En ese caso, se anulará la primera pulsación (es decir, si pasa mucho tiempo desde la primera pulsación, se deberá considerar que no se ha efectuado la pulsación del primer punto).

- Las coordenadas de los dos puntos pulsados (*X1* y *X2*) se deben de marcar con una etiqueta sobre las pulsaciones.

- La distancia entre ambos puntos se mostrará sobre una barra de estado (*StatusBar*) creada al efecto.

4.3. USO DE IMAGELIST Y MEJORA DE LA GESTIÓN DE MAPAS

En este apartado se pretende aclarar el funcionamiento de la gestión de los mapas a presentar. En la página de la asignatura, en la sección de Material de Prácticas, está disponiblr para esta

sesión práctica número 3 un fichero llamado *mapas.zip*, referenciado como *Mapas a utilizar*. Dicho fichero incluye en su interior dos ficheros gráficos en formato GIF (*mapa1.gif* y *mapa2.gif*), cuyas dimensiones en pixels son 240x272. Observa que estas dimensiones coinciden con las del *PictureBox* a utilizar en la práctica.

La idea es cargar esas dos imágenes mediante el diseñador gráfico de Visual Studio .NET en un objeto *ImageList*. Entre las propiedades de un objeto de esta clase, aparece Images. Si se pulsa sobre el botón con el texto "..." podremos seleccionar la colección de imágenes incluidas en el objeto *ImageList*, pulsando en el botón *Agregar*. Agregando los dos mapas, a partir de los ficheros *mapa1.gif* y *mapa2.gif*, estos estarán accesibles como objetos *System.Drawing.Bitmap* como *imageList1.Images[0]* e *imageList[1].Images[1]*.

El ComboBox ofrecerá como opciones los textos "*Mapa 1*" y "*Mapa 2*", y el *TextBox* a continuación contendrá el título o nombre que se dará al mapa al representarlo.

Es importante tener en cuenta que una *ImageList* se utiliza para redimensionar el tamaño de un objeto *Bitmap*. Desgraciadamente, el diseñador gráfico de VS.NET se suele quejar cuando alguna de las dos dimensiones de un *Bitmap* incluido en su colección Images excede los 256 pixels, redimensionándolo al tamaño 16x16. Esto puede solucionarse, redimensionando el bitmap en el método de gestión del evento *Load* del formulario (por defecto, *Form1_Load*). Para ello, hay que cambiar la propiedad *Size* del objeto *Bitmap*, asignándole un nuevo objeto *Size* del tamaño deseado (240x272). No tiene sentido hacerlo dentro de *InitializeComponent()*, puesto que su código es gestionado por el diseñador gráfico de VS.NET y podría en cualquier momento volver a poner las dimensiones al valor 16x16.

4.4. AMPLIACIÓN

En lugar de cargar los mapas en un objeto *ImageList* previamente (lo cual incluye los dos ficheros GIF dentro de los recursos de la aplicación, aumentando su tamaño), utiliza un objeto *OpenFileDialog* para ir cargando los mapas en el objeto ImageList en tiempo de ejecución. No se ha propuesto directamente esto en la práctica, puesto que la versión disponible del emulador (Pocket PC 2002) no permite copiar fácilmente ficheros al sistema de ficheros que utilizar el emulador. En cambio, puedes dejar lista una nueva versión de la aplicación para que funcione en una PDA real y probarla con una de las PDAs disponibles.

Nuevas versiones del emulador permiten mapear una unidad compartida dentro del sistema de ficheros del emulador y copiar los ficheros a esa unidad compartida y cargar entonces los ficheros GIF mediante el uso de *OpenFileDialog*.

OpenFileDialog contiene, entre otros, los siguientes miembros:

- Entre las propiedades accesibles desde el diseñador gráfico, pueden ser de interés *InitialDirectory* (para mostrar el directorio que mostrará inicialmente el diálogo; por ejemplo, */MyDocuments/*), y *Filter* y *FilterIndex* para filtrar los ficheros que dejará seleccionar el diálogo (por ejemplo, **.gif*).
- *ShowDialog()*, para mostrar el diálogo; puede llamarse a este método al pulsar un botón, por ejemplo.
- Al volver del método anterior, la propiedad *FileName* contendrá el nombre del fichero seleccionado.