Código fuente de la aplicación “Screen Colors”

En este documento se describe brevemente una sencilla aplicación de demostración escrita en C# que emplean la tecnología de los cubos Sifteo para ilustrar algunos conceptos importantes de la POO, como *clase*, *objeto*, *atributo*, *método* e incluso *herencia*. Asimismo, ciertos conceptos de la programación basada en eventos como los *eventos* y los *tipos delegados* de C#, pueden ilustrarse de igual modo.

Esta aplicación muestra secuencias ordenadas de colores en las pantallas de los cubos (“Screen Colors”). Dichas secuencias progresan de modo independiente en cada cubo por medio de pulsaciones en su pantalla (*ButtonEvent*). Después del último color se muestra de nuevo el primero de la secuencia. La acción de agitar un cubo (*ShakeStartedEvent* y *ShakeStoppedEvent*) reinicia la secuencia de colores para dicho cubo, cuya pantalla vuelve a mostrar el primer color. El resto del documento describe el código del programa paso a paso:

El ensamblado (unidad de empaquetado de software para la plataforma .NET) *System* contiene las APIs fundamentales de C#, mientras que el ensamblado *Sifteo* incluye todas las APIs para comunicarse con los cubos Sifteo.

```
using System;
using Sifteo;
```

La lógica de la aplicación debe escribirse dentro de una clase que herede la clase *BaseApp*. Dado que las clases de aplicación - que son subclases de *BaseApp* - deben ejecutarse bajo el entorno de ejecución *Siftrunner*, a veces se necesita un proceso de arranque para iniciar el programa manualmente desde fuera del entorno *Siftrunner*, para propósitos de depuración. La clase *Bootstrap* es una simple envoltura con un método *Main()* que inicia la ejecución de la aplicación: En primer lugar se obtiene una instancia (objeto) de la clase de aplicación y, a continuación, se invoca su método *run()*.

```
public class Bootstrap {
  public static void Main() {
    ScreenColorsApp myScreenColorsApp = new ScreenColorsApp();
```

```
    myScreenColorsApp.Run();
  }
}
```

ScreenColorsApp es una clase derivada de *BaseApp*. *Siftrunner* buscará esta subclase al iniciar la ejecución de la aplicación. El primer atributo que se declara es un vector al que después se asignarán los colores a mostrar en las pantallas de los cubos. A los atributos *initialColorSound* y *nextColorSound* se asignarán dos objetos *Sound* que se usarán cada vez que la pantalla de un cubo muestre el primer color de la secuencia, y cuando ésta cambie su color actual por el próximo en la secuencia, respectivamente.

El método *Setup()* es invocado al comenzar la ejecución de la aplicación. Por tanto, toda la lógica necesaria para iniciar ésta se debe incluir en dicho método. En este caso, se inicializa la estructura de datos que da soporte a la secuencia de colores: Un vector de un tipo de dato *struct*[1] denominado *Color* y definido en la API de Sifteo. Cada elemento del vector se inicializa usando el constructor del tipo *struct Color*, el cual acepta un color específico proporcionado en formato RGB: Un valor comprendido entre 0 y 255 para cada color básico (rojo, azul y verde).

La variable *colorIndex* se inicializa a cero para que señale el primer color de la secuencia. Este valor se almacenará en el atributo público *userData* de cada objeto cubo. Dicho atributo es a su vez un objeto que proporciona un modo conveniente de etiquetar una instancia de la clase *Cube* con algunos datos adicionales necesarios para los propósitos de la aplicación.

La propiedad *BaseApp.Sounds* permite acceder al conjunto de sonidos disponibles para una aplicación concreta. La invocación del método *CreateSound()* crea una instancia de reproducción de audio a partir de una secuencia de sonido registrada en formato PCM (*modulación por codificación de pulsos*), que

[1] El tipo de dato estructurado *struct* del lenguaje C# está basado en el tipo del mismo nombre que existe en C++. Puede contener métodos y constructores. El tipo *struct* se diferencia de una *clase* en el modo de almacenamiento en la memoria: El primero es un tipo de *valor*, mientras que el segundo es un tipo de *referencia*. Además, el tipo *struct* no soporta la herencia.

corresponde a un fichero ".wav" de Ms-Windows, o MP3. Se dispone, por tanto, de una *factoría de objetos* que permite la creación de instancias *Sound*, las cuales representan secuencias de sonido. Para esta aplicación se han utilizado dos ficheros de sonido denominados "intialcolor.mp3" y "nextcolor.mp3", a partir de los cuales se crean respectivamente los objetos *initialColorSound* y *nextColorSound*.

Cada instancia de la clase *Cube* representa un cubo Sifteo "físico". La clase *CubeSet* representa el conjunto de cubos actualmente conectados al ordenador a través del dispositivo radiotransmisor USB de Sifteo. La propiedad[2] *BaseApp.CubeSet* proporciona una instancia de *CubeSet* para el uso de los desarrolladores, que puede indexarse para acceder a un objeto concreto de la clase *Cube*: Tales objetos se almacenan en el vector en el orden en que han sido conectados al sistema, de modo que *CubeSet[0]* proporciona acceso al primer objeto cubo que se conectó.

La sentencia *foreach* permite iterar sobre todos los cubos de *CubeSet* para realizar el mismo conjunto de acciones sobre cada cubo que se halla conectado al sistema. En primer lugar, se asigna el valor actual de la variable que indexa el vector de colores (*colorIndex*) al atributo público *userData* de cada objeto cubo. A continuación, se colorea la pantalla completa de cada cubo con el color RGB del vector cuya posición está señalada por el índice, que en este caso vale cero (el primer color de la secuencia). No obstante, nada se dibujará en la pantalla del cubo hasta que el método *paint()* sea invocado (ver descripción del método *Tick()* en un próximo párrafo). En las últimas líneas del bloque de código vinculado a la sentencia *foreach*, se añaden manejadores de eventos[3] a cada objeto cubo para gestionar las acciones del usuario

[2] El lenguaje C# define con el término *propiedad* a un miembro que da acceso a una característica de un objeto o una clase. Para acceder a una propiedad se usa la misma sintaxis que para acceder a un campo de datos (denominación de atributo en C#), pero el acceso es traducido por el compilador en llamadas a métodos de consulta y actualización *get()* y *set()* definidos en dicha propiedad.

[3] Las clases de la API Sifteo, en particular *BaseApp*, *CubeSet* y *Cube*, generan eventos para notificar a una aplicación de ciertos sucesos, incluyendo las manipulaciones que el usuario efectúa sobre los cubos.

sobre el botón y los acelerómetros. Para esta aplicación se consideran los eventos generados cuando el usuario pulsa y libera la pantalla de un cubo, y cuando lo agita.

Finalmente, el método *Play()* es invocado sobre el objeto *initialColorSound* para indicar que el primer color de la secuencia aparece por primera vez en la pantalla de cada cubo conectado al sistema anfitrión y, por tanto, la aplicación se ha iniciado. El parámetro que se usa en la llamada al método referido es un número real que indica el volumen del sonido, y está comprendido entre cero (el volumen más bajo) y uno (el más alto) para el sonido a reproducir.

```
public class ScreenColorsApp : BaseApp {

  Color [] colors;
  Sound initialColorSound, nextColorSound;

  public override void Setup() {

    int colorIndex = 0;

    // colors: red, green, blue, yellow, cyan, magenta and white
    colors = new Color[] { new Color(255, 0, 0), new Color(0, 255, 0),
             new Color(0, 0, 255), new Color(255, 255, 0),
             new Color(0, 255, 255), new Color (255, 0, 255),
             new Color(255, 255, 255) };

    initialColorSound = Sounds.CreateSound("initialcolor");
    nextColorSound = Sounds.CreateSound("nextcolor");

    foreach (Cube cube in CubeSet) {
            cube.userData = colorIndex;
            cube.FillScreen(colors[colorIndex]);
            cube.ButtonEvent += OnButton;
            cube.ShakeStartedEvent += OnShakeStarted;
            cube.ShakeStoppedEvent += OnShakeStopped;
            }

    initialColorSound.Play(1);
  }
```

La palabra clave *override* permite que el método *Tick()* original de la clase base *BaseApp* se sustituya en la clase derivada *ScreenColorsApp* por una implementación más adecuada para los propósitos de la

---

Para gestionar las ocurrencias de un evento concreto, se debe añadir a éste un método manejador, el cual especifica las acciones a llevar a cabo cuando suceda el evento.

aplicación. Por omisión es invocado cada 1/20 de segundo (la frecuencia es ajustable) y, en este caso, itera sobre los objetos que representan a los cubos conectados al sistema anfitrión para invocar a su método *paint()*. Dicho método actualiza la imagen que se muestra en la pantalla de un cubo, de modo que todos los gráficos previamente dibujados a través de los correspondientes métodos aparecen ahora en su pantalla.

```
public override void Tick() {
  foreach (Cube cube in CubeSet) cube.Paint();
}
```

Éste es un manejador para el evento *Button*, que se origina cuando la pantalla de un cubo es pulsada o liberada. El parámetro *pressed* es verdadero en el primer caso, y falso en el segundo. Obviamente, el parámetro *cube* representa el cubo Sifteo cuyo botón se ha accionado. La propiedad *UniqueID* da como resultado el *identificador único* del objeto cubo cuando ésta es consultada. El valor de dicha propiedad permanece constante frente a desconexiones entre el cubo y el sistema anfitrión, e inicios de nuevas sesiones de juego, de modo que siempre puede utilizarse este valor para identificar a un objeto cubo concreto de forma unívoca.

```
private void OnButton(Cube cube, bool pressed) {
  Console.Write("Cube: {0}. ", cube.UniqueId);
  if (pressed) {
     Console.WriteLine("Button pressed.");
     nextColorSound.Play(1);
     }
  else {
     Console.WriteLine("Button released.");
     int colorIndex = (int) cube.userData;
     colorIndex++;
     if (colorIndex >= colors.Length)
        colorIndex = 0;
     cube.userData = colorIndex;
     cube.FillScreen(colors[colorIndex]);
     }
}
```

Cuando se pulsa la pantalla de un cubo - también actúa como botón -, se reproduce el sonido que corresponde a un cambio de color. Cuando la pantalla se libera, el valor del índice correspondiente al color actual de ésta se recupera a partir de la lectura del atributo público *userData* del objeto cubo, y dicho valor se asigna a la variable local *colorIndex*. A continuación, se incrementa en una unidad el valor

de dicha variable. Si éste alcanza la longitud del vector que da soporte a la secuencia de colores, a la variable se le asigna el valor cero para que indique de nuevo el primer color de la secuencia. Finalmente, al atributo *userData* del objeto cubo se le asigna el valor actualizado de la variable *colorIndex* y se colorea la pantalla del cubo con el color RGB que indica el elemento del vector indexado por la variable.

Éste es un manejador para el evento *ShakeStarted*, que se origina cuando el usuario comienza a agitar un cubo. El parámetro *cube* representa el cubo Sifteo que está siendo agitado, y su identificador único se muestra en el monitor del ordenador para facilitar la trazabilidad de la ejecución. La acción concreta a realizar en este caso consiste en reproducir el sonido correspondiente a un nuevo comienzo de la secuencia de colores.

```
  private void OnShakeStarted(Cube cube) {
    Console.WriteLine("Cube: {0}. Shake start.", cube.UniqueId);
    initialColorSound.Play(1);
  }
```

Este es un manejador para el evento *ShakeStopped*, que se origina cuando el usuario deja de agitar un cubo. El parámetro *cube* representa el cubo Sifteo que está siendo agitado, y su identificador único se muestra en el monitor del ordenador para facilitar la trazabilidad de la ejecución. El parámetro *duration* indica el tiempo que el cubo ha sido agitado, y se mide en milisegundos. En primer lugar, se declara una variable local denominada *colorIndex* y se inicializa a valor cero. A continuación, se asigna este valor al atributo público *userData* del objeto *cube* y se colorea la pantalla del cubo con el primer color de la secuencia de colores, indicado por el primer elemento del vector que da soporte a la misma.

```
  private void OnShakeStopped(Cube cube, int duration) {

    int colorIndex = 0;

    Console.Write("Cube: {0}. ", cube.UniqueId);
    Console.WriteLine("Shake stop: {0} seconds.", duration / 1000.0);
    cube.userData = colorIndex;
    cube.FillScreen(colors[colorIndex]);
  }
} // end of class "ScreenColorsApp"
```