UserForms extremos

Sergio Propergol

ayudaexcel
Formación y recursos para Excelers

Contenido

Técnicas de UserForm avanzadas

Esta publicación complementa al manual “De 0 a 100 con macros y VBA”. En él encontrarás más ejemplos de UserForms.

- Usar UserForm no modales.
- Mostrar un indicador de progreso (tres técnicas).
- Crear un asistente (una serie de cuadros de diálogo interactivos).
- Crear una función que imita a la función `MsgBox` de VBA.
- Permitir a los usuarios mover los controles del UserForm.
- Controlar varios objetos con un solo controlador de objetos.
- Mostrar un UserForm sin barra de titulo.
- Simular una barra de herramientas con un UserForm.
- Permitir a los usuarios modificar el tamaño de un UserForm.
- Controlar varios controles con un solo controlador de eventos.
- Usar un cuadro de diálogo para seleccionar un color.
- Mostrar un gráfico en un UserForm.
- Mostrar una hoja de cálculo completa en un UserForm.

La mayoría de estos ejemplos son más avanzados, pero todos se basan en aplicaciones prácticas. Sin embargo, incluso los ejemplos que no son tan prácticos muestran técnicas muy útiles.

Un cuadro de diálogo no modal

La mayoría de los cuadros de diálogo que encontrarás, serán modales. Un cuadro modal significa que debes hacerlo desaparecer de la pantalla antes de poder realizar cualquier acción con Excel. Sin embargo, algunos cuadros de diálogo son no modales, lo que significa que el usuario puede seguir trabajando en la aplicación mientras se muestra en la pantalla.

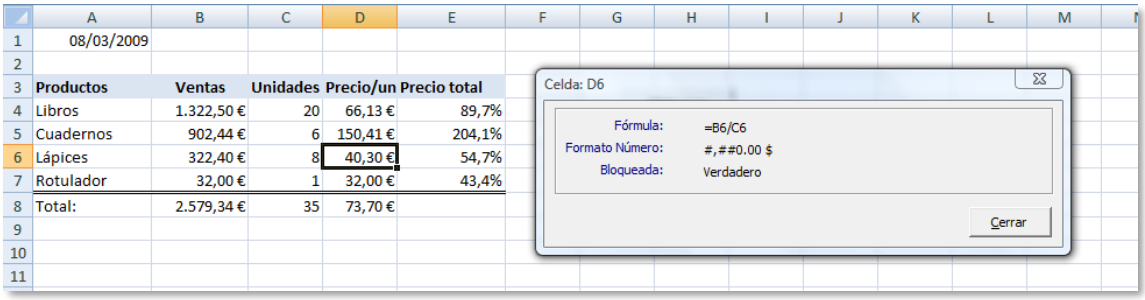
Para mostrar un `UserForm` no modal, se utiliza una instrucción como la siguiente:

```
UserForm1.Show vbModeless
```

La palabra `vbModeless` es una constante integrada que tiene un valor predeterminado de 0. Por lo tanto, la siguiente instrucción funciona de forma idéntica:

```
UserForm1.Show 0
```

El siguiente ejemplo muestra un cuadro de diálogo no modal que contiene información sobre la celda activa. Cuando se muestra el cuadro de diálogo, el usuario puede mover el cursor de la celda, activar otras hojas y realizar otras acciones con Excel.



La clave está en determinar cuándo actualizar la información del cuadro de diálogo. Para ello, el ejemplo supervisa dos eventos del libro: **SheetSelectionChange** y **SheetActivate**. Estos procedimientos de control de eventos se encentran en el módulo de código para el objeto **ThisWorkbook**.

Los procedimientos de control de eventos son los siguientes:

```
Private Sub Workbook_SheetSelectionChange _
  (ByVal Sh As Object, ByVal Target As Range)
    Call UpdateBox
End Sub

Private Sub Workbook_SheetActivate(ByVal Sh As Object)
    Call UpdateBox
End Sub
```

Estos procedimientos invocan al procedimiento **UpdateBox**, que se muestra a continuación:

```
Sub UpdateBox()
    With UserForm1
'       Nos aseguramos de que hay una hoja activa
        If TypeName(ActiveSheet) <> "Worksheet" Then
            .lblFormula.Caption = "N/A"
            .lblNumFormat.Caption = "N/A"
            .lblLocked.Caption = "N/A"
            Exit Sub
        End If

        .Caption = "Celda: " & ActiveCell.Address(False,
False)
'       Formula
        If ActiveCell.HasFormula Then
            .lblFormula.Caption = ActiveCell.Formula
        Else
            .lblFormula.Caption = "(ninguna)"
        End If
'       Formato número
        .lblNumFormat.Caption = ActiveCell.NumberFormat
'       Bloqueada
```

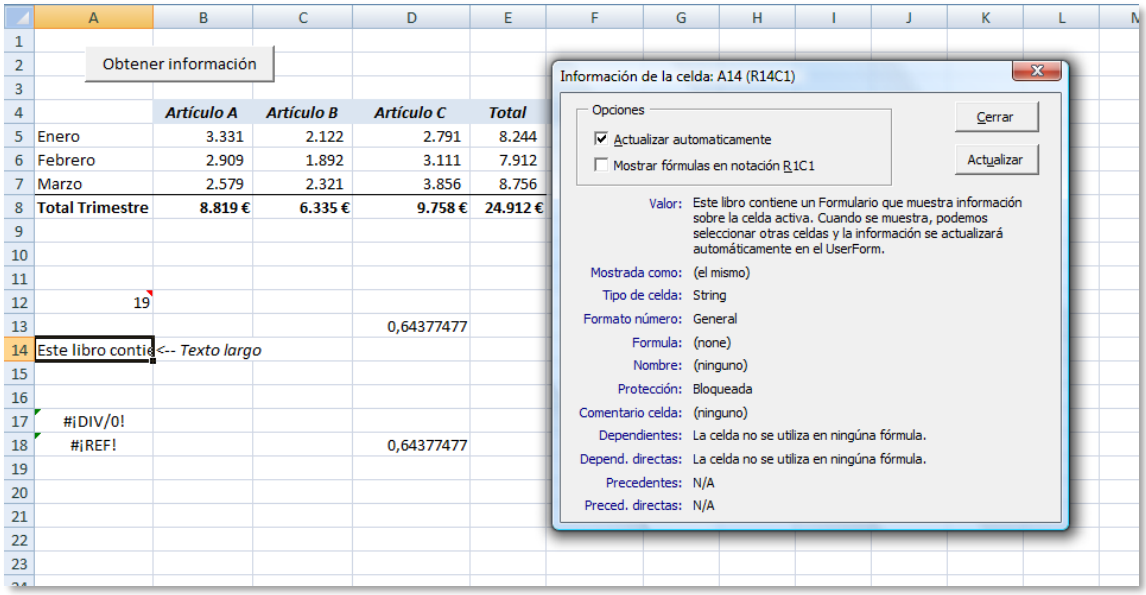
```
            .lblLocked.Caption = ActiveCell.Locked
        End With
    End Sub
```

El procedimiento **UpdateBox** cambia el título del UserForm para que muestre la dirección de la celda activa; después actualiza los tres controles de etiqueta (**lblFormula**, **lblNumFormat** y **lblLocked**).

A continuación, se muestran algunos puntos que te ayudarán a comprender el funcionamiento de este ejemplo:

- El UserForm no es modal para hacer posible el acceso a la hoja mientras se muestra.
- El código de la parte superior del procedimiento te asegura que la hoja activa es una hoja de trabajo. Si la hoja no es una hoja de trabajo, es decir, una hoja de gráfico o macro, se asigna a los controles de etiqueta el texto N/A.
- El libro supervisa la celda activa usando un evento **Selection_Change** (que se encuentra en el módulo de código de **ThisWorkbook**).
- La información se muestra en los controles de etiqueta del UserForm.

La siguiente imagen muestra una versión mucho más sofisticada de este ejemplo. Esta versión contiene un poco más de información acerca de la celda seleccionada. Los usuarios más experimentados notarán las similitudes con la ventana Info (una función que fue eliminada de Excel hace varios años).



A continuación, se muestran algunos puntos clave de esta versión más sofisticada.

- El UserForm tiene una casilla de verificación (**Actualizar automáticamente**). Cuando esta casilla de verificación está activada, el UserForm se actualiza automáticamente. Cuando no lo está, el usuario puede pulsar el botón **Actualizar** para actualizar la información.
- El libro utiliza un módulo de clase para supervisar dos eventos para todos los libros abiertos: el evento **SheetSelectionChange** y el evento **SheetActivate**.

Como resultado, el código que muestra la información de la celda se ejecuta automáticamente cada vez que se producen estas acciones en cualquier libro (suponiendo que la opción Actualizar automáticamente está activa). Algunas acciones (como cambiar el formato de número de la celda) no desencadenan ninguna de estas acciones. Por lo tanto, el UserForm también contiene un botón **Actualizar**.

- El recuento que se muestra para el campo de precedentes y dependientes de la celda incluye únicamente las celdas de la hoja activa. Esto es una limitación de las propiedades `Precedents` y `Dependents`.
- Como la longitud de la información variará, se utiliza código VBA para modificar el tamaño de las etiquetas y espaciarlas verticalmente (y también para cambiar la altura del UserForm si es necesario).

Mostrar un indicador de progreso

Una de las necesidades más comunes de los programadores de Excel tiene que ver con los indicadores de progreso. Un indicador de progreso es un gráfico parecido a un termómetro que muestra el progreso de una tarea, como una macro, que tarde mucho en ejecutarse.

La creación de un indicador de progreso es una tarea relativamente sencilla. En esta sección se muestra cómo crear tres tipos de indicadores de progreso para:

- Una macro que no se inicia mediante un UserForm (un indicador de progreso independiente).
- Una macro iniciada por un UserForm. En este caso, el UserForm usa el control de página múltiple, que muestra un indicador de progreso mientras se está ejecutando la macro.
- Una segunda macro iniciada por un UserForm. En este caso, se aumenta la altura del UserForm y el indicador de progreso aparece en la parte inferior del cuadro.

Para utilizar un indicador de progreso tienes que (de alguna manera) calcular cuánto tardará la macro en completar su trabajo. Esto se puede hacer de varias formas, dependiendo de la macro. Por ejemplo, si escribe datos en celdas (y sabes el número de celdas que se van a rellenar), basta con escribir código que calcule el porcentaje completado. Incluso si no puedes calcular de forma precisa el proceso de una macro, es una buena idea indicar al usuario que la macro todavía está en ejecución.

Un indicador de progreso ralentizará un poco (imperceptible en la mayoría de las ocasiones) la macro debido al esfuerzo extra de tener que actualizarla. Si la rapidez es absolutamente crucial en el libro, quizás sea mejor renunciar al indicador de progreso.

Mostrar el progreso en la barra de estado

Un modo simple de mostrar el progreso de una macro es utilizar la barra de estado de Excel. La ventaja es que es muy fácil de programar. Sin embargo, el inconveniente es que la mayoría de los usuarios no están acostumbrados a mirar la barra de estado y preferirían algo más visual.

Para escribir texto en la barra de estado, utilizamos una instrucción como ésta:

```
Application.StatusBar = "Por favor, espere…"
```

Por supuesto, se puede actualizar la barra de estado mientras progresa la macro. Por ejemplo, si tienes una variable llamada Pct que representa el porcentaje completado, puedes escribir código que periódicamente ejecute una instrucción como esta:

```
Application.StatusBar = "Procesando... " & Pct & "% _
completado."
```

Cuando la macro finaliza, se devuelve la barra de estado a su estado normal con la siguiente declaración:

```
Application.StatusBar = False
```

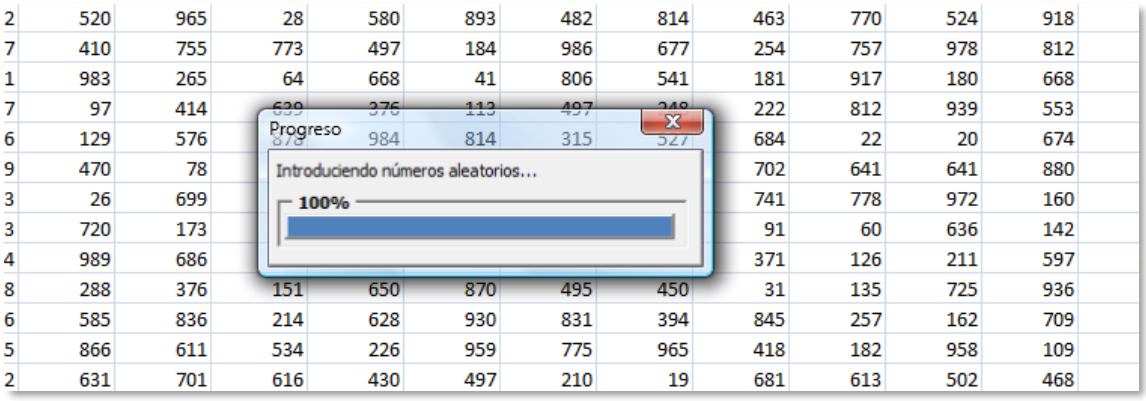
Si no restableces la barra de estado, continuará mostrándose el mensaje final.

Crear un indicador de progreso independiente

Esta sección describe cómo configurar un indicador de progreso independiente (es decir, que no comienza mostrando un UserForm) para mostrar el progreso de una macro. La macro simplemente deja la hoja en blanco y escribe 20.000 números al azar en un rango de celdas.

```
Sub GenerateRandomNumbers()
'   Inserta números al azar en la hoja activa
    Dim Counter As Integer
    Const RowMax As Integer = 500
    Const ColMax As Integer = 40
    Dim r As Integer, c As Integer
    Dim PctDone As Single

    If TypeName(ActiveSheet) <> "Worksheet" Then Exit Sub
    Cells.Clear
    Counter = 1
    For r = 1 To RowMax
        For c = 1 To ColMax
            Cells(r, c) = Int(Rnd * 1000)
            Counter = Counter + 1
        Next c
        PctDone = Counter / (RowMax * ColMax)
        Call UpdateProgress(PctDone)
    Next r
    Unload UserForm1
End Sub
```

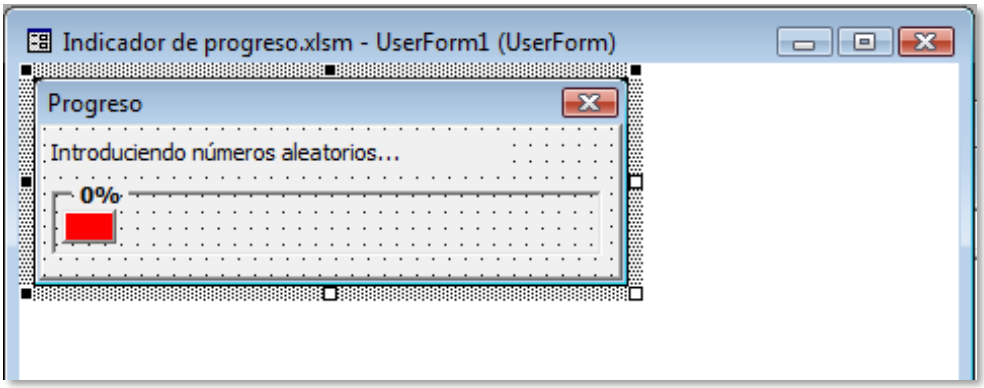
Después de realizar unas pocas modificaciones en la macro, que veremos en la siguiente sección, el UserForm mostrará su progreso.



Construir el UserForm indicador de progreso independiente

Sigue estos pasos para crear el UserForm que se utilizará para mostrar el progreso de la tarea:

1. Inserta un nuevo UserForm y cambia su propiedad **`Caption`** a **`Progreso`**.
2. Añade un control de macro y llámalo **`MarcoProgreso`**.
3. Añade un control de etiqueta dentro del marco y llámalo **`LabelProgress`**. Elimina el título de la etiqueta y haz que su color de fondo (**`BackColor`**) sea rojo. El tamaño de la etiqueta y su colocación no son importantes por el momento.
4. Añade otra etiqueta sobre el marco para describir qué está pasando (opcional).
5. Configura el UserForm y los controles para que aparezcan como en el dibujo.



Por supuesto, puedes aplicar cualquier otro tipo de formato a los controles. Por ejemplo, se puede cambiar la propiedad **`SpecialEffect`** del control de marco que aparece en el dibujo anterior.

Crear los procedimientos de controlador de eventos para el indicador de progreso independiente

En este caso, el truco consiste en ejecutar un procedimiento automáticamente cuando se muestra el UserForm. Una manera de hacerlo es utilizar el evento **`Initialize`**. Sin embargo, este evento tiene lugar antes de que el UserForm se muestre realmente, por lo que no resulta útil. Por otro lado, el evento **`Activate`** se desencadena cuando se muestra el UserForm, así que es perfecto para esta aplicación.

Inserta el siguiente procedimiento en la ventana Código del UserForm. Este procedimiento simplemente invoca a un procedimiento llamado **GenerateRandomNumbers** cuando se muestra el UserForm. Este procedimiento, que se almacena en un módulo VBA, es la macro que se ejecuta realmente mientras se muestra el indicador de progreso.

```
Private Sub UserForm_Activate()
    Call GenerateRandomNumbers
End Sub
```

A continuación, se muestra el procedimiento **GenerateRandomNumbers**, que viste anteriormente, con algunas modificaciones. Observa que el código adicional realiza un seguimiento del progreso y lo almacena en una variable llamada **PctDone**.

```
Sub GenerateRandomNumbers()
'   Inserta números al azar en la hoja activa
    Dim Counter As Integer
    Const RowMax As Integer = 500
    Const ColMax As Integer = 40
    Dim r As Integer, c As Integer
    Dim PctDone As Single

    If TypeName(ActiveSheet) <> "Worksheet" Then Exit Sub
    Cells.Clear
    Counter = 1
    For r = 1 To RowMax
        For c = 1 To ColMax
            Cells(r, c) = Int(Rnd * 1000)
            Counter = Counter + 1
        Next c
        PctDone = Counter / (RowMax * ColMax)
        Call UpdateProgress(PctDone)
    Next r
    Unload UserForm1
End Sub
```

El procedimiento **GenerateRandomNumbers** contiene dos bucles. Dentro del bucle central hay una llamada al procedimiento **UpdateProgress**. Este procedimiento toma un argumento: la variable **PctDone**, que representa el progreso de la macro. **PctDone** tendrá un valor entre **0** y **100**.

```
Sub UpdateProgress(Pct)
    With UserForm1
        .FrameProgress.Caption = Format(Pct, "0%")
        .LabelProgress.Width = Pct * (.FrameProgress. _
          Width - 10)
        .Repaint
    End With
End Sub
```

Crear un procedimiento de inicio para un indicador de progreso independiente

Todo lo que falta es un procedimiento que muestre el UserForm. Introducimos el siguiente procedimiento en un módulo VBA:

```
Sub ShowUserForm()
    With UserForm1
        .LabelProgress.BackColor = ActiveWorkbook.Theme. _
            ThemeColorScheme.Colors(msoThemeAccent1)
        .LabelProgress.Width = 0
        .Show
    End With
End Sub
```

Cómo funciona el indicador de progreso independiente

Cuando se ejecuta el procedimiento **ShowUserForm**, se establece que el ancho del objeto **Label** sea **0**. Después, el método **Show** del objeto **UserForm1** muestra el UserForm (que es el indicador de progreso). Cuando se muestra el UserForm, se desencadena su evento **Activate**, lo que ejecuta el procedimiento **GenerateRandomNumbers**. Este procedimiento contiene el código que invoca al procedimiento **UpdateProgress** cada vez que el contador de la variable del bucle r se actualiza. Observa que el procedimiento **UpdateProgress** utiliza el método **Repaint** del objeto UserForm. Sin esta instrucción, no se actualizarían los cambios de la etiqueta. Antes de que finalice el procedimiento, la última declaración descarga el UserForm.

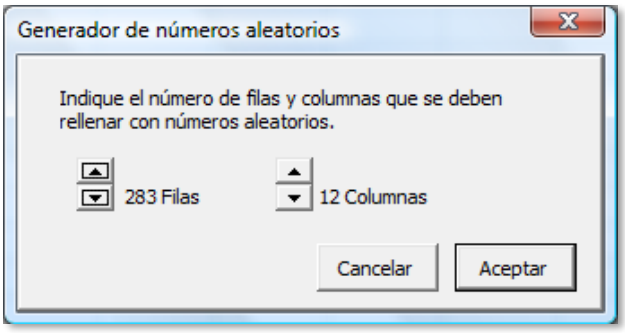
Para personalizar esta técnica, tendrás que averiguar cómo determinar el porcentaje completado y asignárselo a la variable **PctDone**. Esto variará dependiendo de la aplicación. Si el código se ejecuta en un bucle (como en este ejemplo), determinar el porcentaje completado es sencillo. Si el código no está en un bucle, quizás tengas que calcular el progreso completado en varios puntos del código.

Mostrar un indicador de progreso utilizando un control de página múltiple

En el ejemplo anterior, la macro no estaba iniciada por un UserForm. En muchos casos, una macro que tarda mucho en ejecutarse se inicia cuando el usuario hace clic en el botón Aceptar de un UserForm. La técnica descrita en esta sección será una solución más adecuada y presupone lo siguiente:

- El proyecto está completo y sin errores.
- El proyecto utiliza un UserForm (sin controles de página múltiple) para iniciar una macro que tarda mucho en ejecutarse.
- Hay un modo de calcular el progreso de la macro.

Como en el ejemplo previo, éste introduce números aleatorios en una hoja. La diferencia es que esta aplicación contiene un UserForm donde el usuario puede especificar la cantidad de filas y columnas en las que insertar los números aleatorios.

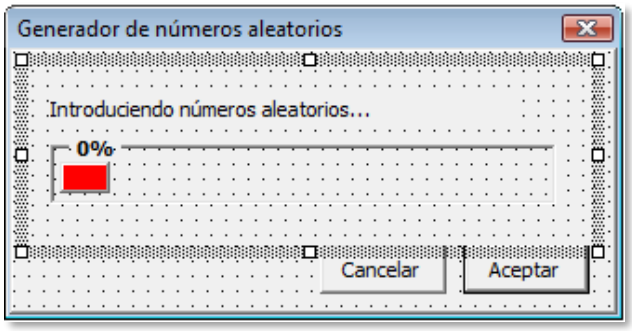


Modificar un UserForm para un indicador de progreso con un control de página múltiple

Este paso presupone que existe un UserForm completamente configurado. Se añadirá un control de página múltiple. La primera página del control de página múltiple contendrá todos los controles originales. La segunda contendrá los controles que muestran el indicador de progreso. Cuando la macro comience a ejecutarse, el código VBA cambiará la propiedad **`Value`** del control de página múltiple. Esto ocultará efectivamente los controles originales y mostrará el indicador de progreso.

El primer paso es añadir un control de página múltiple a un UserForm. A continuación, se mueven todos los controles del UserForm y se copian en la página 1 del control de página múltiple.

El siguiente paso es activar la página 2 y configurarla como la que se puede ver en la siguiente imagen. Es básicamente la misma combinación de controles utilizados en el ejemplo de la sección anterior.



1. Añade un control de marco y llámalo **`FrameProgress`**.
2. Añade un control de etiqueta y llámalo **`LabelProgress`**. Elimina el título de la etiqueta y haz que su fondo sea de color rojo.
3. Añade otra etiqueta para describir qué está pasando (opcional).
4. El siguiente paso es activar el control de página múltiple (no una página del control) y asignar a su propiedad **`Style`** el valor **`2 – fbTabStyleNone`** (esto ocultará las etiquetas). El modo más fácil de seleccionar el control de página múltiple es utilizar la lista desplegable de la ventana **Propiedades**. Probablemente tendrás que ajustar el tamaño del control de página múltiple para tener en cuenta que las etiquetas no están visibles.

Insertar el procedimiento UpdateProgress para un indicador de progreso con un control de página múltiple

Inserta el siguiente procedimiento en el módulo de código del UserForm:

```
Sub UpdateProgress(Pct)
    With UserForm1
        .FrameProgress.Caption = Format(Pct, "0%")
        .LabelProgress.Width = Pct * (.FrameProgress. _
          Width - 10)
    End With
    DoEvents
End Sub
```

Este procedimiento se invoca desde la macro que se ejecuta cuando el usuario hace clic en el botón **Aceptar**, y realiza la actualización del indicador de progreso.

Modificar el procedimiento para un indicador de progreso con un control de página múltiple

Tendrás que modificar el procedimiento que se ejecuta cuando el usuario hace clic en el botón Aceptar (el procedimiento de control del evento `Click` para el botón, llamado `OK_Click`). En primer lugar, inserta la siguiente instrucción en la parte superior del procedimiento:

```
MultiPage1.Value = 1
```

Esta instrucción activa la página 2 del control de página múltiple (la página que muestra el indicador de progreso).

En el siguiente paso, todo dependerá mucho más del programador. Quizás haya que escribir código que calcule el porcentaje completado y asigne este valor a una variable llamada `PctDone`. Este cálculo se realizará dentro de un bucle. Después se insertará la siguiente instrucción, actualizando el indicador de progreso:

```
Call UpdateProgress(PctDone)
```

Cómo funciona un indicador de progreso con un control de página múltiple

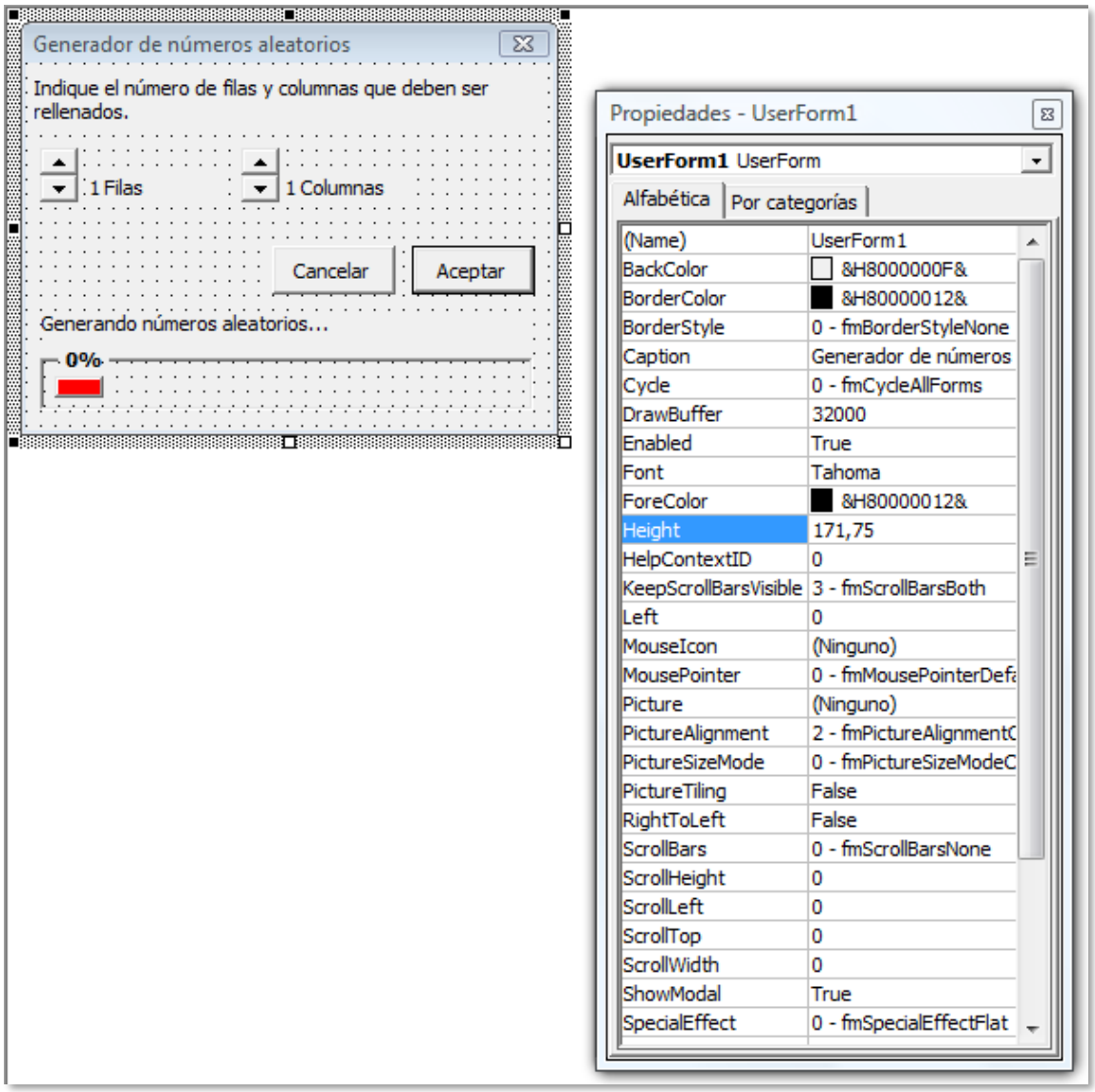
Esta técnica es muy sencilla y, como has visto, sólo usa un UserForm. El código intercambia páginas del control de página múltiple y convierte un cuadro de diálogo normal en un indicador de progreso. Como en las fichas de la página múltiple están ocultas, ni siquiera se parece a un control de página múltiple.

Mostrar un indicador de progreso sin utilizar un control de página múltiple

El ejemplo de esta sección es similar al ejemplo de la anterior. Sin embargo, esta técnica es más simple porque no utiliza un control de página múltiple. En lugar de eso, el indicador de progreso se almacena en la parte inferior del UserForm (pero se reduce la altura del UserForm de forma que los controles del indicador de progreso no sean visibles). Cuando hay que mostrar el indicador de progreso, se incrementa la altura del UserForm, lo que hace que se vea el indicador de progreso.

La siguiente imagen muestra el UserForm en el editor de Visual Basic. La propiedad **Height** del UserForm es **172**. Sin embargo, antes de que aparezca, el valor de **Height** cambia a **124** (lo que significa que los controles del indicador de progreso no están visibles para el usuario). Cuando el usuario hace clic en Aceptar, el código VBA cambia la propiedad **Height** a **172** utilizando la siguiente instrucción:

```
Me.Height = 172
```



Crear asistentes

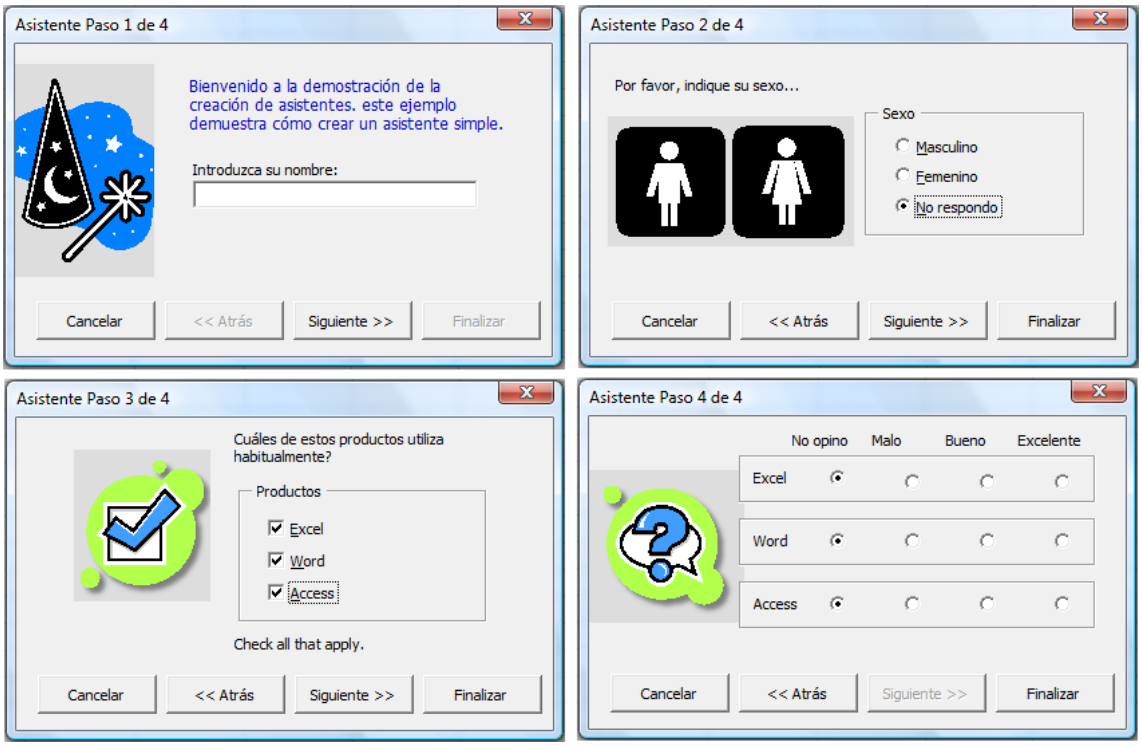
Muchas aplicaciones incorporan asistentes para ayudar a los usuarios en alguna operación. El asistente para importar texto de Excel es un buen ejemplo. Un asistente es básicamente una serie de cuadros de diálogo que solicitan información del usuario. A menudo, las elecciones del usuario en los primeros cuadros de diálogo influyen en los contenidos de los siguientes

cuadros de diálogo. En la mayoría de los asistentes, el usuario puede avanzar o retroceder en la secuencia de cuadros de diálogo o hacer clic en el botón Finalizar para aceptar todos los valores predeterminados.

Por supuesto, es posible crear asistentes utilizando VBA y series de UserForms. Sin embargo, el modo más eficiente de crear un asistente es utilizar un único UserForm y un control de página múltiple con las fichas ocultas. La siguiente imagen muestra un ejemplo de un simple asistente de cuatro pasos que consta de un único UserForm que contiene un control de página múltiple. Cada paso del asistente muestra una página diferente del control de página múltiple.

Configurar el control de página múltiple para el asistente

Comienza creando un nuevo UserForm y añade un control de página múltiple. Por defecto, este control contiene dos páginas. Haz clic con el botón derecho del ratón en la ficha de la página múltiple e inserta suficientes páginas nuevas para controlar el asistente (una página por cada paso). El ejemplo mostrado es un asistente de cuatro pasos, de modo que el control de página múltiple tendrá cuatro páginas. Los nombres de las fichas de la página múltiple no son importantes. La propiedad `Style` del control tendrá un valor final de `2 – fmTabStyleNone`. Mientras trabajas con el UserForm querrás mantener las fichas visibles para hacer más fácil acceder a las diferentes páginas.



A continuación, añade los controles que necesites en cada página del control de página múltiple. Por supuesto, esto variará dependiendo de la aplicación. Puede que tengas que modificar el tamaño del control de página múltiple mientras trabajas para tener espacio para los controles.

Añadir los botones al UserForm del asistente

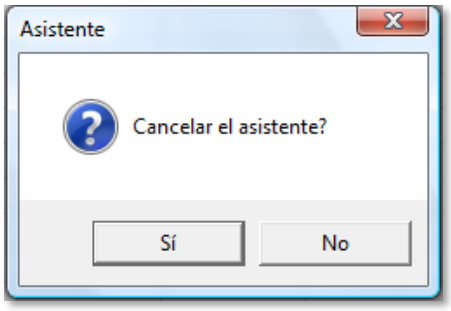
A continuación se añadirán los botones que controlan el progreso del asistente. Estos botones se colocan fuera del control de página múltiple porque se usan mientas se muestra cualquiera de las páginas. La mayoría de los asistentes tienen cuatro botones:

- **Cancelar**: Cancela el asistente.
- **Anterior**: Vuelve al paso previo. Durante el Paso 1 del asistente, este botón debe estar deshabilitado.
- **Siguiente**: Avanza al siguiente paso. Durante el último paso, este botón debe estar deshabilitado.
- **Finalizar**: Finaliza el asistente.

En el ejemplo, estos botones de comando se llaman `CancelButton`, `BackButton`, `NextButton` y `FinishButton`.

En algunos casos, el usuario puede hacer clic en el botón **Finalizar** en cualquier momento y aceptar los valores predeterminados para aquellos elementos que no ha definido. En otros casos, el asistente necesita una respuesta del usuario para algunos elementos. En ese caso, el botón **Finalizar** se deshabilita hasta que se realicen todas las entradas necesarias.

Programar los botones del asistente



Cada uno de los cuatro botones del asistente necesita un procedimiento para controlar su evento `Click`. A continuación se muestra el control de eventos para `CancelButton`. Este procedimiento utiliza una función `MsgBox` para verificar que el usuario quiere salir realmente. Si el usuario hace clic en el botón Sí, el UserForm se descarga sin realizar ninguna acción. Por supuesto, este tipo de comprobación es opcional.

```
Private Sub CancelButton_Click()
    Dim Msg As String
    Dim Ans As Integer
    Msg = "Cancelar el asistente?"
    Ans = MsgBox(Msg, vbQuestion + vbYesNo, APPNAME)
    If Ans = vbYes Then Unload Me
End Sub
```

Los procedimientos de control de eventos para los botones **Atrás** y **Siguiente** son los siguientes:

```
Private Sub BackButton_Click()
    MultiPage1.Value = MultiPage1.Value - 1
    UpdateControls
End Sub
Private Sub NextButton_Click()
    MultiPage1.Value = MultiPage1.Value + 1
    UpdateControls
End Sub
```

Estos dos procedimientos son muy simples. Cambian la propiedad **Value** del control de página múltiple y después invocan a otro procedimiento llamado **UpdateControls** (que se muestra a continuación). El procedimiento **UpdateControls** se encarga de habilitar y deshabilitar los controles **BackButton** y **NextButton**.

```
Sub UpdateControls()
    Select Case MultiPage1.Value
        Case 0
            BackButton.Enabled = False
            NextButton.Enabled = True
        Case MultiPage1.Pages.Count - 1
            BackButton.Enabled = True
            NextButton.Enabled = False
        Case Else
            BackButton.Enabled = True
            NextButton.Enabled = True
    End Select

'   Actualiza el título
    Me.Caption = APPNAME & " Paso " _
      & MultiPage1.Value + 1 & " de " _
      & MultiPage1.Pages.Count

'   El campo Nombre es obligatorio
    If tbName.Text = "" Then
        FinishButton.Enabled = False
    Else
        FinishButton.Enabled = True
    End If
End Sub
```

El procedimiento cambia el título del UserForm para mostrar el paso en el que se encuentra y el número total de pasos. **APPNAME** es una constante pública definida en el Módulo1. A continuación, examina el campo del nombre de la primera página (un cuadro de texto llamado **tbName**). Este es un campo necesario, de modo que no se puede hacer clic en el botón Finalizar. Si el cuadro de texto está vacío, se deshabilita **FinishButton**; en caso contrario, se habilita.

Programar dependencias en un asistente

En la mayoría de los asistentes, una respuesta del usuario en un determinado paso puede afectar a lo que se muestra en el siguiente paso. Los botones de opción para la valoración de un producto sólo estarán visibles si el usuario indica un determinado producto.

A efectos de programación, esto se consigue supervisando el evento **Change** del control de página múltiple. Cada vez que se cambia el valor de la página múltiple (haciendo clic en los botones **Atrás** o **Siguiente**), se ejecuta el procedimiento **MultiPage1_Change**. Si el control de página múltiple está en la última ficha (Paso 4), el procedimiento examina los valores de los controles de casilla de verificación del paso 3 y realiza los ajustes apropiados en el paso 4.

En este ejemplo, el código utiliza dos matrices de controles, uno para los controles de casilla de verificación (paso 3) y otro para los controles de marco (paso 4). El código utiliza un bucle **For-Next** para ocultar los controles de los productos que no se utilizan y ajusta su posición

vertical. Si ninguna de las casillas de verificación del paso 3 está marcada, se ocultan todos los elementos del paso 4 excepto un cuadro de texto que muestra **Haz clic en Finalizar para salir** (si no se introdujo un nombre en el paso1) o **Se requiere un nombre para el Paso 1** (si no se introdujo un nombre en el paso 1). El procedimiento `MultiPage1_Change` aparece a continuación:

```
Private Sub MultiPage1_Change()
    Dim TopPos As Integer
    Dim FSpace As Integer
    Dim AtLeastOne As Boolean
    Dim i As Integer

'   Actualizar página de puntuaciones
    If MultiPage1.Value = 3 Then
'       Crea un array con casillas de verificación
        Dim ProdCB(1 To 3) As MSForms.CheckBox
        Set ProdCB(1) = cbExcel
        Set ProdCB(2) = cbWord
        Set ProdCB(3) = cbAccess

'       Crea un array con controles de marco
        Dim ProdFrame(1 To 3) As MSForms.Frame
        Set ProdFrame(1) = FrameExcel
        Set ProdFrame(2) = FrameWord
        Set ProdFrame(3) = FrameAccess

        TopPos = 22
        FSpace = 8
        AtLeastOne = False

'       Bucle por todos los productos
        For i = 1 To 3
            If ProdCB(i) Then
                ProdFrame(i).Visible = True
                ProdFrame(i).Top = TopPos
                TopPos = TopPos + ProdFrame(i).Height + _
                FSpace
                AtLeastOne = True
            Else
                ProdFrame(i).Visible = False
            End If
        Next i

'       No usa productos?
        If AtLeastOne Then
            lblHeadings.Visible = True
            Image4.Visible = True
            lblFinishMsg.Visible = False
        Else
            lblHeadings.Visible = False
            Image4.Visible = False
            lblFinishMsg.Visible = True
            If tbName = "" Then
                lblFinishMsg.Caption = _
                  "Se requiere un nombre para el paso 1."
```

```
            Else
                lblFinishMsg.Caption = _
                  "Haz clic en Finalizar para salir."
             End If
        End If
    End If
End Sub
```

Realizar la tarea con el asistente

Cuando el usuario hace clic en el botón Finalizar, el asistente realiza su tarea: transferir la información desde el UserForm a la siguiente fila vacía de la hoja. Este procedimiento, llamado **FinishButton_Click**, es muy claro. Comienza determinando la siguiente fila vacía de la hoja y asignando a este valor una variable (r). El resto del procedimiento simplemente traslada los valores de los controles e introduce datos en la hoja.

```
Private Sub FinishButton_Click()
    Dim r As Long

    r = Application.WorksheetFunction. _
      CountA(Range("A:A")) + 1

'   Insertar el nombre
    Cells(r, 1) = tbName.Text

'   Insertar el sexo
    Select Case True
        Case obMale: Cells(r, 2) = "Masculino"
        Case obFemale: Cells(r, 2) = "Femenino"
        Case obNoAnswer: Cells(r, 2) = "Desconocido"
    End Select

'   Insertar productos
    Cells(r, 3) = cbExcel
    Cells(r, 4) = cbWord
    Cells(r, 5) = cbAccess

'   Insertar puntuaciones
    If obExcel1 Then Cells(r, 6) = ""
    If obExcel2 Then Cells(r, 6) = 0
    If obExcel3 Then Cells(r, 6) = 1
    If obExcel4 Then Cells(r, 6) = 2
    If obWord1 Then Cells(r, 7) = ""
    If obWord2 Then Cells(r, 7) = 0
    If obWord3 Then Cells(r, 7) = 1
    If obWord4 Then Cells(r, 7) = 2
    If obAccess1 Then Cells(r, 8) = ""
    If obAccess2 Then Cells(r, 8) = 0
    If obAccess3 Then Cells(r, 8) = 1
    If obAccess4 Then Cells(r, 8) = 2

    Unload Me

End Sub
```

Tras probar el asistente y comprobar que todo parece funcionar correctamente, podemos establecer el valor para la propiedad Style del control de página múltiple a **2 – fmTabStyleNone**.

Emular la función MsgBox

La función **MsgBox** de VBA es un poco diferente porque, a diferencia de la mayoría de las funciones, muestra un cuadro de diálogo. Pero, al igual que las demás funciones, también devuelve un valor: un número entero que representa en qué botón hizo clic el usuario.

Este ejemplo estudia una función personalizada que emula la función **MsgBox** de VBA. Al principio, crear tal función puede parecer bastante fácil, pero no es del todo cierto. La función **MsgBox** es extraordinariamente versátil gracias a los argumentos que acepta. Por tanto, crear una función que emule **MsgBox** no es una tarea fácil.

La clave de este ejercicio no es crear una función alternativa que muestre mensajes. El objetivo es mostrar cómo desarrollar una función relativamente compleja que también incorpora un UserForm. Sin embargo, a algunas personas les puede gustar la idea de ser capaces de personalizar sus mensajes. En ese caso, descubrirán que esta función es muy fácil de personalizar. Por ejemplo, pueden cambiar las fuentes, los colores, el botón de texto y elementos similares.

Hemos llamado a esta función que emula a **MsgBox** **MiMsgbox**. La emulación no es perfecta. **MiMsgBox** tiene las siguientes limitaciones:

- No admite el argumento **Helpfile** (que añade un botón Ayuda, el cual, cuando se hace clic sobre él, abre un archivo de ayuda).
- No admite el argumento **Context** (que especifica la identidad del contexto para el archivo de ayuda).
- No admite la opción de sistema modal, que detiene todos los procesos de Windows hasta que se responde al cuadro de diálogo.

La sintaxis para **MiMsgBox** es:

```
MiMsgBox (mensaje [, botones] [, título])
```

Esta sintaxis es exactamente igual que la sintaxis de **MsgBox**, excepto que no utiliza los dos últimos argumentos opcionales (**Helpfile** y **Context**). **MsgBox** también utiliza las mismas constantes predefinidas que **MsgBox**: **vbOkOnly**, **vbQuestion**, **vbDefaultButton1** y similares.

Emulación de MsgBox: el código MiMsgBox

La función **MiMsgBox** usa un UserForm llamado **MyMsgBoxForm**. La función en sí misma, que se muestra a continuación, es muy corta. La mayor parte del trabajo se hace en el procedimiento **UserForm_Initialize**.

```
Function MiMsgBox(ByVal prompt As String, _
    Optional ByVal buttons As Integer, _
    Optional ByVal title As String) As Integer
        Prompt1 = prompt
        Buttons1 = buttons
        Title1 = title
        MiMsgBoxForm.Show
        MiMsgBox = UserClick
End Function
```

A continuación, se muestra el código utilizado para ejecutar la función:

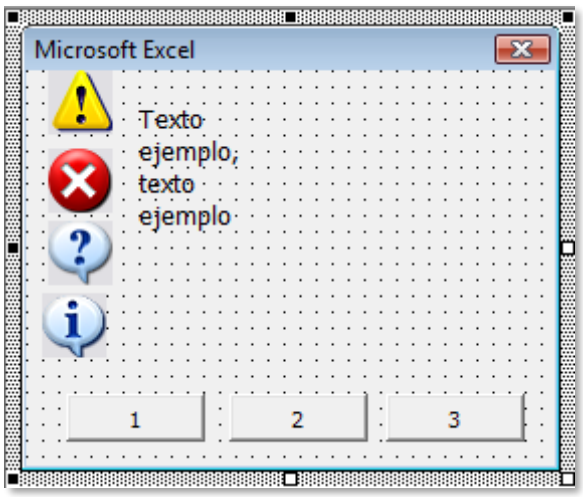
```
prompt = "Está a punto de formatear su disco duro."
prompt = prompt & vbCrLf & vbCrLf & "¿Desea continuar?"
buttons = vbQuestion + vbYesNo
title = "Existe un problema"
Ans = MiMsgBox(prompt, buttons, title)
```

Por supuesto, este ejemplo no borrará ningún disco duro.

Cómo funciona la emulación de MsgBox

Hay que destacar el uso de cuatro variables **Public**. Las tres primeras (**Prompt1**, **Buttons1** y **Title1**) representan los argumentos que se pasan a la función. La otra variable (**UserClick**) representa los valores devueltos por la función. El procedimiento **UserForm_Initialize** necesita un modo de obtener esta información y enviarla de nuevo a la función, y la única forma de hacer esto es utilizar la variable **Public**.

El UserForm que aparece a continuación contiene cuatro controles de imagen (uno para cada uno de los posibles iconos), tres controles de botón de comando y un control de cuadro de texto.



El código del procedimiento **UserForm_Initialize** examina los argumentos y hace lo siguiente:

- Determina qué imagen (si la hay) se mostrará (y oculta las otras).
- Determina qué botón (o botones) se muestra (y oculta los otros).
- Determina qué botón es el botón predeterminado.
- Centra los botones en el cuadro de diálogo.
- Determina los títulos de los botones de comando.
- Determina la posición del texto dentro del cuadro de diálogo.
- Determina el ancho del cuadro de diálogo (utiliza una llamada API para obtener la resolución del vídeo).
- Determina la altura que debe tener el cuadro de diálogo.
- Muestra el UserForm.

Se incluyen otros tres procedimientos de control de eventos (uno para cada botón de comando). Estas rutinas determinan en qué botón se hizo clic y devuelven un valor para la función asignando un valor a la variable **UserClick**.

Interpretar el segundo argumento (**buttons**) es un poco más complicado. Este argumento puede estar formado por varias constantes añadidas juntas. Por ejemplo, el segundo argumento puede ser algo así:

```
vbYesNoCancel + vbQuestion + vbDefaultButton3
```

Este argumento crea un **MsgBox** con tres botones (**Sí**, **No** y **Cancelar**), muestra el icono de pregunta y hace que el tercer botón sea el botón predeterminado. El argumento real es 547 (3 + 32 + 512).

El objetivo era obtener tres fragmentos de información a partir de un único número. La solución implica convertir el argumento en un número binario y después examinar los bits específicos. Por ejemplo, el equivalente binario de 547 es 1000100011. Los dígitos binarios de 4 a 6 determinan la imagen que se muestra, los dígitos de 8 a 10 determinan qué botones mostrar y los dígitos de 1 y 2 determinan qué botón es el botón predeterminado.

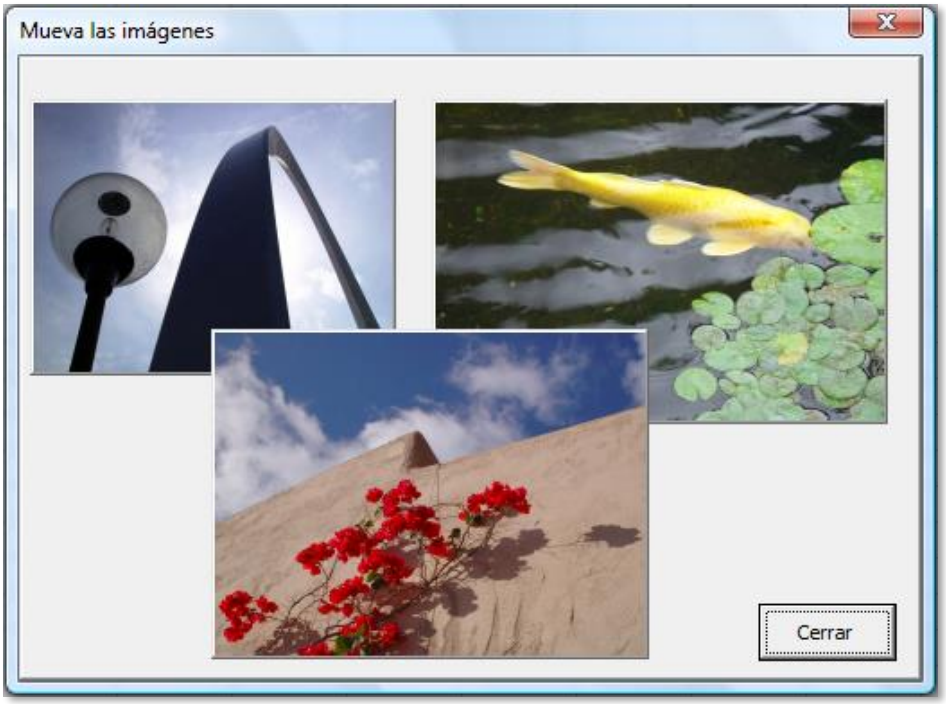
Utilizar la función MiMsgBox en la emulación de MsgBox

Para utilizar esta función en tus propios proyectos, exporta el módulo **MiMsgBoxMod** y el UserForm **MyMsgBoxForm**. A continuación, importa estos dos archivos al proyecto. Después podrás utilizar la función **MiMsgBox** en tu código como lo harías con la función **MsgBox**.

Un UserForm con controles deslizantes

A pesar de las reducidas aplicaciones prácticas de esta técnica, el ejemplo de esta sección te ayudará a comprender los eventos relacionados con el ratón. El UserForm que se muestra a

continuación contiene tres controles de imagen. El usuario puede utilizar el ratón para arrastrar las imágenes por el cuadro de diálogo.



Cada uno de los tres controles de imagen tiene dos procedimientos de eventos asociados: **MouseDown** y **MouseMove**. El procedimiento de evento para el control **Image1** se muestra a continuación (los demás controles de imagen son idénticos a este, excepto por el nombre de cada control):

```
Private Sub Image1_MouseDown(ByVal Button As Integer, _
    ByVal Shift As Integer, ByVal X As Single, ByVal _
     Y As Single)
'   Posición de inicio cuando se presiona el botón
    OldX = X
    OldY = Y
    Image1.ZOrder 0
End Sub

Private Sub Image1_MouseMove(ByVal Button As Integer, _
    ByVal Shift As Integer, ByVal X As Single, ByVal _
     Y As Single)
'   Mueve la imagen
    If Button = 1 Then
        Image1.Left = Image1.Left + (X - OldX)
        Image1.Top = Image1.Top + (Y - OldY)
    End If
End Sub
```

Cuando se hace clic con el botón del ratón, tiene lugar el evento **MouseDown**, y se guardan las posiciones X e Y del cursor. Se utilizan dos variables públicas para hacer un seguimiento de la

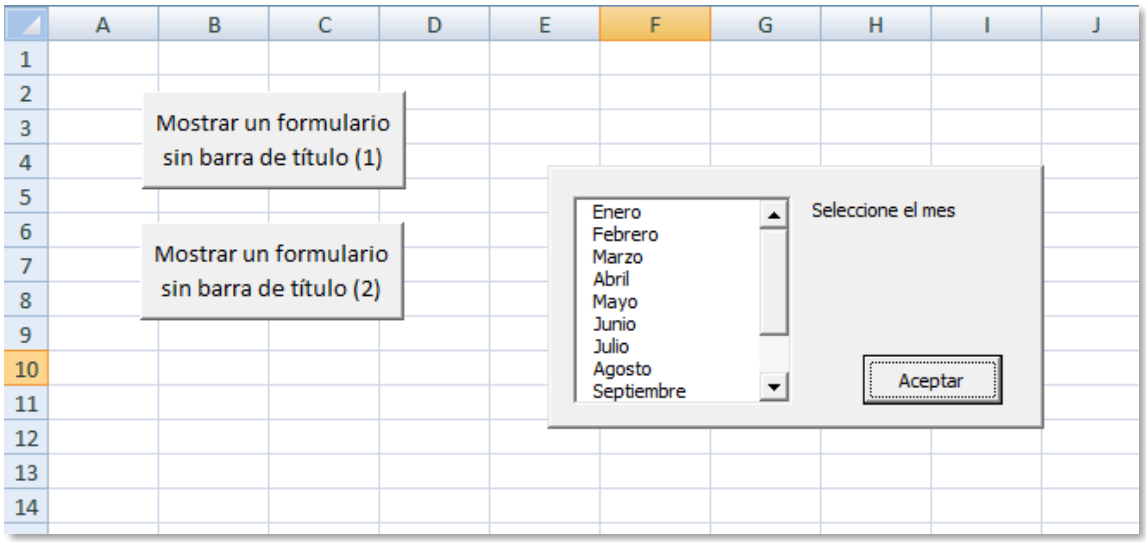
posición original de los controles: **OldX** y **OldY**. Este procedimiento también modifica la propiedad **ZOrder**, que coloca la imagen sobre las otras.

Cuando se mueve el ratón, el evento **MouseMove** ocurre repetidamente. El procedimiento del evento comprueba el botón del ratón. Si el argumento **Button** es **1**, significa que el botón izquierdo del ratón está presionado. En este caso, el control de imagen se mueve con relación a su antigua posición.

También, observa que el cursor cambia cuando está sobre la imagen. Esto se debe a que la propiedad **MousePointer** es **15 – fmMousePointSizeAll**. Este estilo de cursor se utiliza comúnmente para indicar que algo se puede mover.

Un UserForm sin barra de título

Excel no ofrece ninguna manera de mostrar directamente un UserForm sin barra de título. Pero este reto es posible con la llamada de algunas funciones API. La siguiente imagen muestra un UserForm sin barra de título.

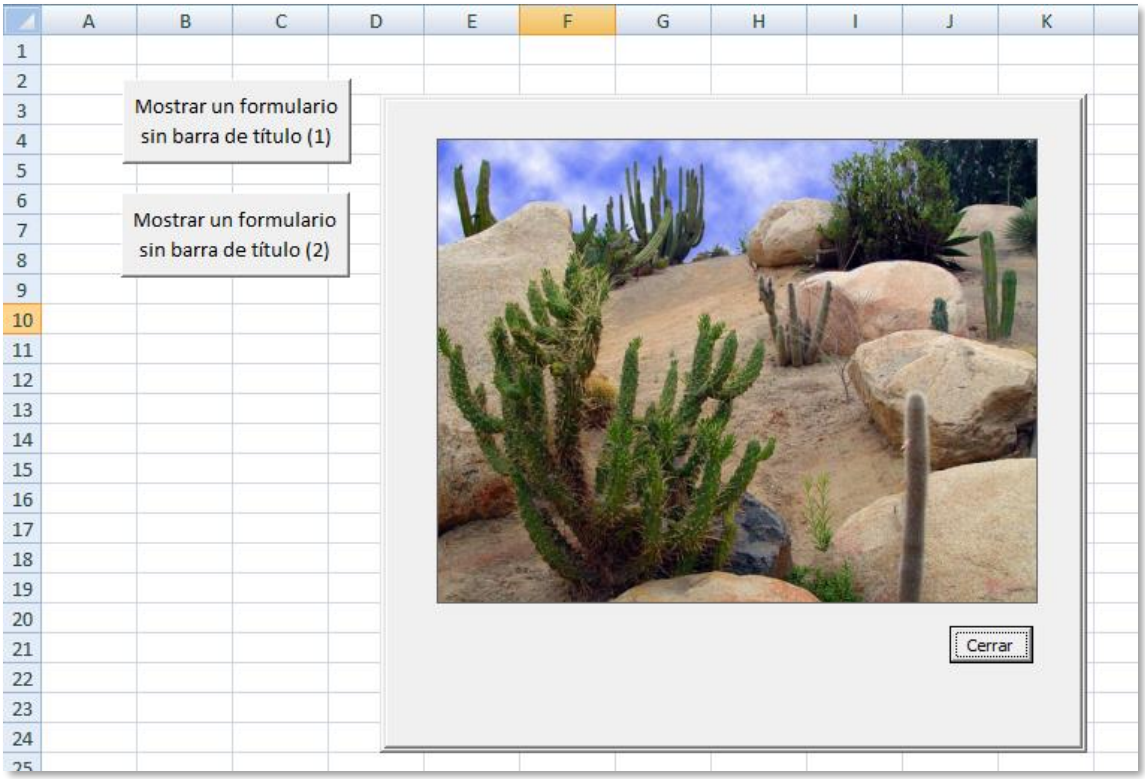


Otro ejemplo de UserForm sin barra de título se ofrece en la siguiente imagen. Este cuadro de diálogo contiene un control de imagen y un control de botón de comando.

Para mostrar un UserForm sin barra de título necesitas cuadro funciones API: **GetWindowLong**, **SetWindowLong**, **DrawMenuBar**, y **FindWindowA**. El procedimiento **UserForm_Initialize** invoca estas funciones:

```
Private Sub UserForm_Initialize()
    Dim lngWindow As Long, lFrmHdl As Long
    lFrmHdl = FindWindowA(vbNullString, Me.Caption)
    lngWindow = GetWindowLong(lFrmHdl, GWL_STYLE)
    lngWindow = lngWindow And (Not WS_CAPTION)
    Call SetWindowLong(lFrmHdl, GWL_STYLE, lngWindow)
```

```
        Call DrawMenuBar(lFrmHdl)
    End Sub
```



Un problema de no mostrar la barra de título, es que el usuario no tiene forma de cambiar la posición del cuadro de diálogo. La solución es utilizar los eventos `MouseDown` y `MouseMove`, descritos en la sección anterior.

Simular una barra de herramientas con un UserForm

La creación de barras de herramientas en las versiones anteriores a Excel 2007 era relativamente fácil. A partir de Excel 2007 es imposible. Para ser exactos, es posible crear una barra de herramientas personalizada con VBA, pero Excel 2007 y versiones posteriores ignoran muchas de las instrucciones VBA. A partir de Excel 2007 , todas las barras de herramientas personalizadas se muestran en el grupo **Barras personalizadas**, de la ficha **Complementos**. Estas barras de herramientas no se pueden mover, cambiar de tamaño, ni acoplar.

Esta sección describe cómo crear una barra de herramientas alternativa: un UserForm no modal que simula una barra de herramientas flotante. El siguiente dibujo muestra un UserForm que puede sustituir a una barra de herramientas.

El UserForm contiene ocho controles de imagen, y cada uno de ellos ejecuta una macro.

Al ver, en el editor de Visual Basic, el UserForm en modo Diseño, podrás observar que:

- Los controles no están alineados
- El UserForm no es el tamaño final.
- El tamaño de la barra de título es estándar.

El código VBA se encarga de la apariencia. Alinea los controles y ajusta la imagen del UserForm para que no se malgaste ningún espacio, además, el código emplea funciones API de Windows para reducir el tamaño de la barra de título, justo como una barra de herramientas de verdad. Para hacer que el UserForm se parezca todavía más a una barra de herramientas, se establecerá la propiedad **ControlTipText** de cada control de imagen. Esto hará que se muestre información sobre la herramienta similar a la de una barra de herramientas cada vez que pases el cursor sobre ese control.

Si abres el archivo correspondiente a este ejemplo, de los adjuntos al manual, también observarás que las imágenes cambian ligeramente cuando pasas el cursor sobre ellas. Esto es porque cada control de imagen tiene asociado un controlador de eventos **MouseMove** que cambia la propiedad **SpecialEffect**. Este es el controlador de eventos **MouseMove** de **Image1** (los demás son idénticos).

```
Private Sub Image1_MouseMove(ByVal Button As Integer, _
  ByVal Shift As Integer, ByVal X As Single, ByVal Y As
Single)
    Call NoRaise
    Image1.SpecialEffect = fmSpecialEffectRaised
End Sub
```

Este procedimiento invoca al procedimiento **NoRaise**, que desactiva el efecto especial de relieve de cada control.

```
Private Sub NoRaise()
    Dim ctl As Control
    For Each ctl In Controls
        ctl.SpecialEffect = fmSpecialEffectFlat
    Next ctl
End Sub
```

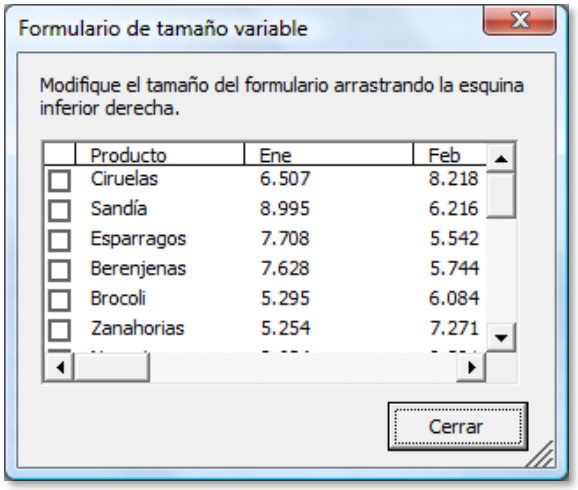
El efecto final es que el usuario consigue cierta información visual cuando el cursor pasa sobre el control, como sucedería con una barra de herramientas de verdad. Sin embargo, esto es todo lo que da de sí la simulación de la barra de herramientas. No es posible cambiar el tamaño del UserForm (por ejemplo, hacer que las imágenes se muestren verticalmente en lugar de horizontalmente). Y, por supuesto, tampoco es posible acoplar la barra de herramientas a uno de los bordes de la ventana de Excel.

Un UserForm de tamaño ajustable

Excel utiliza varios cuadros de diálogo de tamaño ajustable. Por ejemplo, puede cambiarse el tamaño del cuadro de diálogo **Administrador de nombres** si haces clic en la esquina inferior derecha y arrastras.

Si quieres crear un UserForm de tamaño ajustable, pronto descubrirás que no hay una forma de hacerlo directamente. Una solución es recurrir a las llamadas de la API de Windows. El método funciona, pero es complicado de configurar. En esta sección verás una técnica más sencilla para crear un UserForm de tamaño ajustable.

La siguiente imagen muestra el UserForm descrito en esta sección. Contiene un control de cuadro de lista que muestra los datos de una hoja. Observa las barras de desplazamiento del cuadro de lista. Esto quiere decir que hay más información de la que se puede ver. También fíjate en la esquina inferior derecha del cuadro de diálogo. Muestra un control de tamaño que probablemente te resulte familiar.



Las imágenes que se muestran son caracteres de la fuente Wingdings. Para introducir el carácter en una celda selecciona en Excel **Insertar>Texto>Símbolo**. Después cópialo al portapapeles y pégalo en la propiedad `Picture` en la ventana **Propiedades**. Esta es una forma rápida y sencilla de insertar imágenes en los controles de los UserForm.

Al arrastrar la esquina inferior derecha observa que el cuadro de lista también aumenta o disminuye su tamaño junto con el del UserForm y que el botón Cerrar se mantiene en la misma posición relativa. Puedes aumentar el tamaño del UserForm hasta los límites del monitor. Un UserForm puede tener un ancho y alto máximo de 12.287,25 unidades. La altura mínima es de 25.75 unidades (igual que la altura de la barra de título), y el ancho mínimo es de 105 unidades.

El truco está en el control de etiqueta, que se agrega al UserForm en tiempo de ejecución. El control de tamaño en la esquina inferior derecha es en realidad un control de etiqueta que muestra la letra “O” (carácter 111) de la fuente Marlett (grupo de caracteres 2). Este control, llamado `objResizer`, se agrega al UserForm en el procedimiento `UserForm_initialize`:

```
Private Sub UserForm_Initialize()
'   Agrega el control de tamaño al formulario
    Set objResizer = Me.Controls.Add("Forms.label.1",
MResizer, True)
    With objResizer
        With .Font
```

```
            .Name = "Marlett"
            .Charset = 2
            .Size = 16
            .Bold = True
        End With
        .BackStyle = fmBackStyleTransparent
        .AutoSize = True
        .BorderStyle = fmBorderStyleNone
        .Caption = "o"
        .MousePointer = fmMousePointerSizeNWSE
        .ForeColor = RGB(100, 100, 100)
        .ZOrder
        .Top = Me.InsideHeight - .Height
        .Left = Me.InsideWidth - .Width
    End With
End Sub
```

Esta técnica se basa en los siguientes hechos:

- El usuario puede mover un control de un UserForm, como viste anteriormente.
- Existen eventos que pueden identificar los movimientos del ratón y las coordenadas del cursor. Estoy hablando de **MouseDown** y **MouseMove**.
- El código VBA puede cambiar el tamaño de un UserForm en tiempo de ejecución, pero un usuario no puede.

Si estudias estos hechos de manera creativa, verás que es posible traducir los movimientos del usuario de un control de etiqueta, en información que se puede utilizar para ajustar el tamaño de un UserForm.

Cuando un usuario hace clic en el objeto de etiqueta **objResizer**, se ejecuta el procedimiento que controla los eventos **objResizer_MouseDown**:

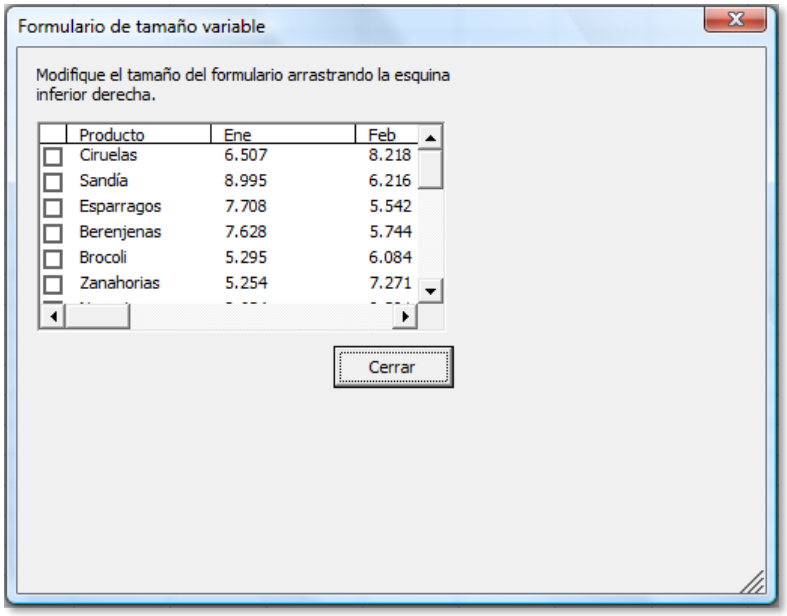
```
Private Sub objResizer_MouseDown(ByVal Button As Integer, _
    ByVal Shift As Integer, ByVal X As Single, ByVal Y As
Single)
    If Button = 1 Then
        LeftResizePos = X
        TopResizePos = Y
    End If
End Sub
```

Este procedimiento se ejecuta únicamente si el botón izquierdo del ratón está presionado (es decir, el argumento **Button** es 1) y el cursor está en la etiqueta **objResizer**. Las coordenadas X e Y del ratón se almacenan en las variables de módulo **LeftResizePos** y **TopResizePos**.

Los movimientos posteriores del ratón inician el evento **MouseMove**, y el controlador de eventos **objResizer_MouseMove** se pone en acción. Esta es una versión inicial de este procedimiento:

```
Private Sub objResizer_MouseMove(ByVal Button As _
Integer, ByVal Shift As Integer, ByVal X As Single, ByVal Y
As Single)
    If Button = 1 Then
        With objResizer
            .Move .Left + X - LeftResizePos, .Top + Y -
TopResizePos
            Me.Width = Me.Width + X - LeftResizePos
            Me.Height = Me.Height + Y - TopResizePos
            .Left = Me.InsideWidth - .Width
            .Top = Me.InsideHeight - .Height
        End With
    End If
End Sub
```

Si te fijas en el código, verás que las propiedades **Width** y **Height** se ajustan basándose en el movimiento del control de etiqueta **objResizer**.



El problema de la imagen anterior, desde luego, es que el otro control del UserForm no responde al nuevo tamaño del UserForm. Debería expandirse el cuadro de lista, y debería cambiar la posición del botón de comando para que permanezca en la esquina inferior derecha.

Es necesario más código VBA para ajustar los controles del UserForm cuando se modifica su tamaño. Este código nuevo se escribirá en el procedimiento controlador de eventos **objResizerMouseMove**. La siguiente declaración se encarga de ello:

```
' Ajusta el cuadro de lista
On Error Resume Next
With ListBox1
    .Width = Me.Width - 22
    .Height = Me.Height - 100
End With
```

```
On Error GoTo 0

'Ajusta el botón Cerrar
With CloseButton
    .Left = Me.Width - 70
    .Top = Me.Height - 54
End With
```

Estos dos controles se ajustan en relación al tamaño del UserForm (es decir, **Me**). Después de insertar este código nuevo, el cuadro de diálogo funciona sin problemas. El usuario lo puede aumentar como necesite, y los controles se ajustan al tamaño.

La parte que presenta un mayor desafío al crear un cuadro de diálogo de tamaño ajustable es lograr que los controles se ajusten. Cuando tienes más de uno o dos controles las cosas se pueden complicar bastante.

Controlar varios botones de UserForm con un controlador de eventos

Cada botón de comando de un UserForm debe tener su propio procedimiento para controlar sus eventos. Por ejemplo, si tenemos dos botones de comando, necesitaremos al menos dos procedimientos de control de eventos.

```
Private Sub CommandButton1_Click()
' Aquí va el código
End Sub

Private Sub CommandButton2_Click()
' Aquí va el código
End Sub
```

En otras palabras, no se puede asignar una macro para que se ejecute cuando se haga clic en cualquier botón de comando. Cada controlador de eventos **Click** está conectado a su botón de comando. Sin embargo, puedes hacer que cada controlador de eventos llame a otra macro que los incluya a todos en los procedimientos de control de eventos, pero tendrás que pasar el argumento que indique en qué botón se hizo clic. En los siguientes ejemplos, al hacer clic en CommandButton1 o CommandButton2 se ejecuta el procedimiento **ButtonClick** y el único argumento indica al procedimiento **ButtonClick** en qué botón se hizo clic.

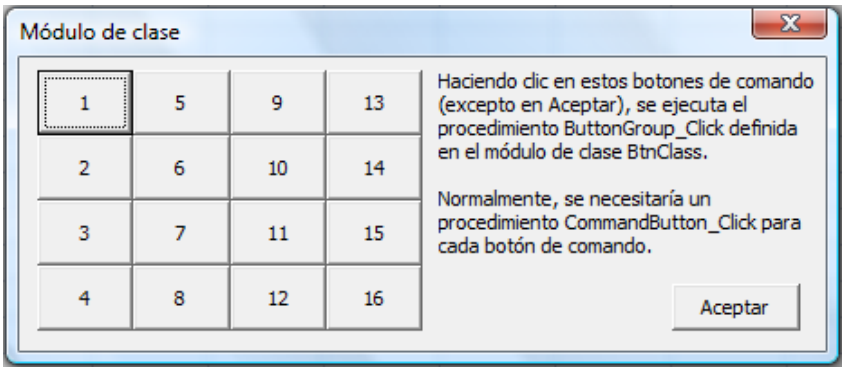
```
Private Sub CommandButton1_Click()
Call ButtonClick(1)
End Sub

Private Sub CommandButton2_Click()
Call ButtonClick(2)
```

```
End Sub
```

Si el UserForm tiene muchos botones de comando, configurar estos procedimientos de control de eventos puede ser muy laborioso. Muchas personas preferirán tener un único procedimiento que pueda determinar en qué botón se hizo clic y realizar la acción apropiada.

Esta sección describe una forma de eliminar esta limitación utilizando un módulo de clase para definir una nueva clase.

Los siguientes pasos describen cómo crear el ejemplo de UserForm que se muestra a continuación:



1. Crea el UserForm de la forma habitual y añade varios botones de comando (este ejemplo tiene 16 botones). El ejemplo asume que el formulario se llama UserForm1.
2. Inserta un módulo de clase en el proyecto (utilizamos **Insertar>Módulo de clase**), llámalo `BtnClass` e introduce el siguiente código. Tendrás que personalizar el procedimiento `ButtonGroup_Click`.

```
Private Sub ButtonGroup_Click()
    Dim Msg As String

    Msg = "Usted hizo clic en " & ButtonGroup.Name & vbCrLf
& vbCrLf
    Msg = Msg & "Nombre: " & ButtonGroup.Caption & vbCrLf
    Msg = Msg & "Posición izquierda: " & ButtonGroup.Left &
vbCrLf
    Msg = Msg & "Posición superior: " & ButtonGroup.Top
    MsgBox Msg, vbInformation, ButtonGroup.Name
End Sub
```

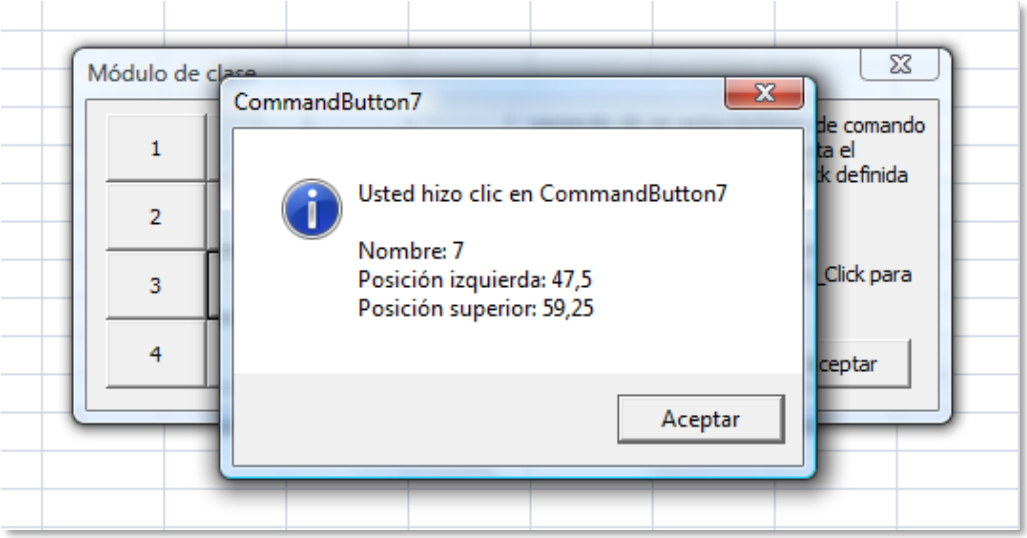
3. Inserta un módulo de VBA estándar e introduce el siguiente código. Esta rutina simplemente muestra el UserForm.

```
Sub ShowDialog()
    UserForm1.Show
End Sub
```

4. En el módulo de código para el UserForm, introduce el siguiente procedimiento **UserForm_Initialize**. Este procedimiento lo inicia el evento **Initialize** del UserForm. Observa que el código excluye el botón llamado **OKButton** del grupo de botones. Por lo tanto, hacer clic en el botón Aceptar no hará que se ejecute el procedimiento **ButtonGroup_Click**.

```
Private Sub UserForm_Initialize()
    Dim ButtonCount As Integer
    Dim ctl As Control

'   Crea los objetos Button
    ButtonCount = 0
    For Each ctl In UserForm1.Controls
        If TypeName(ctl) = "CommandButton" Then
            If ctl.Name <> "OKButton" Then 'Omite el _
                OkButton
                ButtonCount = ButtonCount + 1
                ReDim Preserve Buttons(1 To ButtonCount)
                Set Buttons(ButtonCount).ButtonGroup = ctl
            End If
        End If
    Next ctl
End Sub
```

Tras realizar estos pasos, puedes ejecutar el procedimiento **ShowDialog** para mostrar el UserForm. Haciendo clic en cualquiera de los botones de comando (excepto en Aceptar) se ejecuta el procedimiento **ButtonGroup_Click**.



Seleccionar un color en un UserForm

El ejemplo de esta sección es una función que muestra un cuadro de diálogo (la idea es similar a la función MsgBox, pero un poco más complejo). La función, llamada GetAColor, devuelve un valor de color:

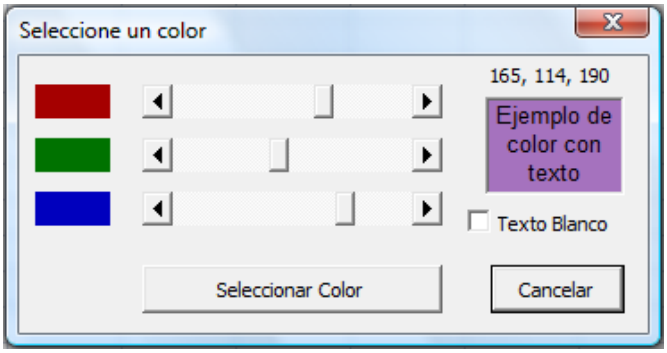
```
Function GetAColor() As Variant
    UserForm1.Show
    GetAColor = ColorValue
End Function
```

Puedes utilizar la función **GetAColor** con una instrucción como la siguiente:

```
UserColor = GetAColor ()
```

Ejecutar esta instrucción muestra el UserForm. El usuario selecciona un color y hace clic en Seleccionar color. A continuación, la función asigna el valor del color que ha seleccionado el usuario a la variable **UserColor**.

El siguiente dibujo muestra el UserForm que contiene tres controles de barra de desplazamiento, uno por cada color de la paleta de color del libro. El valor de cada barra de desplazamiento va desde 0 a 255.



El UserForm **GetAColor** tiene otro detalle: recuerda el último color seleccionado. Cuando la función finaliza, los tres valores de la barra de desplazamiento se almacenan en el registro de Windows, con el siguiente código (**APPNAME** es una cadena definida en Module1):

```
SaveSetting APPNAME, "Colors", "RedValue", _
      ScrollBarRed.Value
SaveSetting APPNAME, "Colors", "BlueValue", _
      ScrollBarBlue.Value
SaveSetting APPNAME, "Colors", "GreenValue", _
      ScrollBarGreen.Value
```

El procedimiento **UserForm_initialize** recupera estos valores y los asigna a las barras de desplazamiento.

```
ScrollBarRed.Value = GetSetting(APPNAME, "Colors", _
     "RedValue", 128)
ScrollBarGreen.Value = GetSetting(APPNAME, "Colors", _
     "GreenValue", 128)
ScrollBarBlue.Value = GetSetting(APPNAME, "Colors", _
     "BlueValue", 128)
```

El último argumento de la función **GetSetting** es el valor predeterminado, que se utilizará en caso de no encontrar la clave del registro. En este caso, cada color predeterminado será 128, lo que genera una tonalidad de gris.

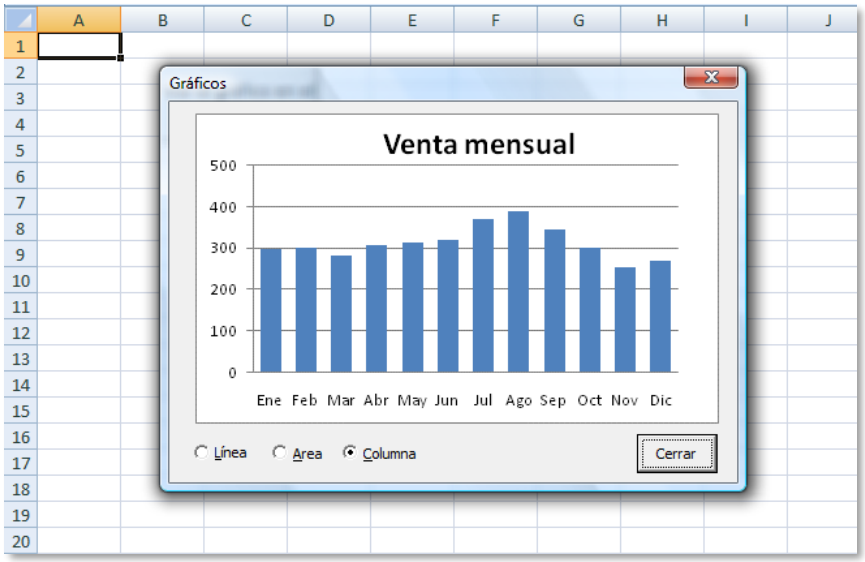
Las funciones **SaveSetting** y **GetSetting** siempre utilizan esta clave del registro:

HKEY_CURRENT_USER\Software|VB and VBA Program Settings\

Mostrar un gráfico en un UserForm

Curiosamente, no hay forma directa de mostrar un gráfico en un UserForm. Por supuesto, puedes copiar el gráfico y pegarlo en la propiedad **Picture** de un control de imagen, pero esto crea una imagen estática del gráfico y no mostrará ningún cambio que realicemos en él.

Esta sección describe cómo mostrar un gráfico en un UserForm. El siguiente dibujo muestra un UserForm con un gráfico mostrado como un objeto de imagen. El gráfico en realidad se encuentra en una hoja y el UserForm siempre muestra el gráfico actual. Esta técnica funciona copiando el gráfico a un archivo de gráficos temporal y después, con la función **LoadPicture**, se especifica el archivo temporal de la propiedad **Picture** del control de imagen.



Pasos generales para mostrar un gráfico en un UserForm

Para mostrar un gráfico en un UserForm, sigue estos pasos generales:

1. Crea un gráfico de la forma habitual.
2. Inserta un UserForm y añadimos un control de imagen.
3. Escribe código de VBA para guardar el gráfico como archivo gif y luego asigna el archivo gif a la propiedad **`Picture`** del control de imagen. Para ello tienes que utilizar la función de VBA **`LoadPicture`**.
4. Añade otros elementos embellecedores a tu gusto. Por ejemplo, el UserForm del archivo del ejemplo contiene controles que permiten cambiar el tipo de gráfico. Alternativamente, podemos escribir código para mostrar varios gráficos.

Guardar un gráfico como archivo gif

El siguiente código muestra cómo crear un archivo gif llamado temp.gif) a partir de un gráfico (en este caso, el primer objeto gráfico de una hoja llamada Datos).

```
Set currentChart = Sheets("Datos").ChartObjects(1).Chart
Fname = Application.DefaultFilePath &
Application.PathSeparator & "temp.gif"
CurrentChart.Export Filename:=Fname, FilterName:="GIF"
```

Cambiar la propiedad Picture del control de imagen

Si el control de imagen del UserForm se llama Image1, la siguiente instrucción carga la imagen (representada por la variable **`Fname`**) en el control de imagen:

```
Image1.Picture = LoadPicture(Fname)
```

Si deseas obtener más ejemplos de UserForms o aprender otras técnicas con VBA te invito a visitar https://www.ayudaexcel.com, donde encontrarás la colección más completa de ejemplos con VBA, plantillas, manuales y trucos de Excel.