Глава 1

Окна

- Привлечение внимания к приложению
- Окно приложения
- Полупрозрачные окна
- Окна и кнопки нестандартной формы
- Немного о перемещении окон
- Масштабирование окон
- Добавление команды в системное меню окна
- Отображение формы поверх других окон

Почему было решено начать книгу именно с необычных приемов использования оконного интерфейса? Причиной стало то, что при работе с операционной системой Windows мы видим окна постоянно и повсюду (отсюда, собственно, и название этой операционной системы). Речь идет не только об окнах приложений, сообщений, свойств: понятие о таких окнах есть у любого начинающего пользователя Windows.

В своих собственных окнах рисуются и элементы управления (текстовые поля, панели инструментов, таблицы, полосы прокрутки, раскрывающиеся списки и т. д.). Взгляните на интерфейс, например, Microsoft Word. Здесь вы увидите, что даже содержимое документа находится в своем собственном окне с полосами прокрутки (правда, это необязательно элемент управления). Окна элементов управления отличаются от «самостоятельных» окон (упрощенно) отсутствием стиля, позволяющего им иметь заголовок, а также тем, что они являются дочерними по отношению к другим окнам. Понимание этого является важным, так как на нем основана часть примеров данной главы.

Рассматриваемые примеры частично используют средства, предусмотренные в Borland Delphi, а частично — возможности «чистого» API (см. гл. 2). Практически все API-функции работы с окнами требуют задания параметра с типом значения `HWND` — дескриптора окна. Это уникальное значение, идентифицирующее каждое существующее в текущем сеансе Windows окно. В Delphi дескриптор окна формы и элемента управления хранится в параметре `Handle` соответствующего объекта.

Нужно также уточнить, что в этой главе термины «окно» и «форма» употребляются как синонимы, когда речь идет о форме. Когда же речь идет об элементах управления, то так и говорится: «окно элемента управления».

Привлечение внимания к приложению

Начнем с простых примеров, позволяющих привлечь внимание пользователя к определенному окну приложения. Это может пригодиться в различных ситуациях: начиная от необходимости уведомления пользователя об ошибке программы и заканчивая простой сигнализацией ему, какое окно в данный момент времени ожидает пользовательского ввода.

Инверсия заголовка окна

Вероятно, вы не раз могли наблюдать, как некоторые приложения после выполнения длительной операции или при возникновении ошибки как бы подмигивают. При этом меняется цвет кнопки приложения на Панели задач, а состояние открытого окна меняется с активного на неактивное. Такой эффект легко достижим при использовании API-функции `FlashWindow` или ее усовершенствованного, но более сложного варианта — функции `FlashWindowEx`.

ПРИМЕЧАНИЕ

Здесь сказано, что функции изменяют цвет кнопки приложения на Панели задач. Однако этого не происходит при выполнении приведенных ниже примеров. Почему так получается и как это изменить, рассказано в следующем разделе этой главы (стр. 15).

Первая из этих функций позволяет один раз изменить состояние заголовка окна и кнопки на Панели задач (листинг 1.1).

Листинг 1.1. Простая инверсия заголовка окна

```
procedure TForm1.cmbFlashOnceClick(Sender: TObject);
begin
  FlashWindow(Handle, True);
end;
```

Как видите, функция принимает дескриптор нужного окна и параметр (тип BOOL) инверсии. Если значение флага равно True, то состояние заголовка окна изменяется на противоположное (из активного становится неактивным и наоборот). Если значение флага равно False, то состояние заголовка окна дважды меняет свое состояние, то есть восстанавливает свое первоначальное значение (активно или неактивно).

Более сложная функция FlashWindowEx в качестве дополнительного параметра (кроме дескриптора окна) принимает структуру FLASHWINFO, заполняя поля которой можно настроить параметры мигания кнопки приложения и/или заголовка окна.

В табл. 1.1 приведено описание полей структуры FLASHWINFO.

Таблица 1.1. Поля структуры FLASHWINFO

Поле	Тип	Назначение
cbSize	UINT	Размер структуры FLASHWINFO (для отслеживания версий)
hwnd	HWND	Дескриптор окна
dwFlags	DWORD	Набор флагов, задающий режим использования функции FlashWindowEx. Значения этих флагов и их описания приведены после таблицы
uCount	UINT	Количество инверсий заголовка окна и/или кнопки на Панели задач
dwTimeout	DWORD	Время между изменениями состояния заголовка окна и/или кнопки на Панели задач. Если значение равно нулю, используется системное значение таймаута

Значение параметра dwFlags формируется из приведенных ниже флагов с использованием операции побитового ИЛИ:

- FLASHW_CAPTION — инвертирует состояние заголовка окна;
- FLASHW_TRAY — заставляет мигать кнопку на Панели задач;
- FLASHW_ALL — сочетание FLASHW_CAPTION и FLASHW_TRAY;
- FLASHW_TIMER — периодически измененяет состояния заголовка окна и/или кнопки на Панели задач до того момента, пока функция FlashWindowEx не будет вызвана с флагом FLASHW_STOP;

- FLASHW_TIMERNOFG — периодически измененяет состояния заголовка окна и/или кнопки на Панели задач до тех пор, пока окно не станет активным;
- FLASHW_STOP — восстанавливает исходное состояние окна и кнопки на Панели задач.

Далее приведены два примера использования функции FlashWindowEx.

В первом примере состояние заголовка окна и кнопки на Панели задач изменяется десять раз в течение двух секунд (листинг 1.2).

Листинг 1.2. Десятикратная инверсия заголовка окна

```
procedure TForm1.cmbInverse10TimesClick(Sender: TObject);
var
  fl: FLASHWINFO;
begin
  fl.cbSize := SizeOf(fl);
  fl.hwnd := Handle;
  fl.dwFlags := FLASHW_CAPTION or FLASHW_TRAY; // аналогично FLASHW_ALL
  fl.uCount := 10;
  fl.dwTimeout := 200;
  FlashWindowEx(fl);
end;
```

Второй пример демонстрирует использование флагов FLASHW_TIMER и FLASHW_STOP для инверсии заголовка окна в течение заданного промежутка времени (листинг 1.3).

Листинг 1.3. Инверсия заголовка окна в течение определенного промежутка времени

```
//Запуск процесса периодической инверсии заголовка
procedure TForm1.cmbFlashFor4SecClick(Sender: TObject);
var
  fl: FLASHWINFO;
begin
  fl.cbSize := SizeOf(fl);
  fl.hwnd := Handle;
  fl.dwTimeout := 200;
  fl.dwFlags := FLASHW_ALL or FLASHW_TIMER;
  fl.uCount := 0;
  FlashWindowEx(fl);
```

```
  Timer1.Enabled := True;
end;
//Остановка инверсии заголовка
procedure TForm1.Timer1Timer(Sender: TObject);
var
  fl: FLASHWINFO;
begin
  fl.cbSize := SizeOf(fl);
  fl.hwnd := Handle;
  fl.dwFlags := FLASHW_STOP;
  FlashWindowEx(fl);
  Timer1.Enabled := False;
end;
```

В данном примере используется таймер, срабатывающий каждые четыре секунды. Таймер первоначально неактивен. Конечно, можно было бы не использовать его, а просто посчитать количество инверсий, совершаемых в течение требуемого интервала времени (в данном случае четырех секунд) и задать его в поле `uCount`. Но приведенный пример предназначен именно для демонстрации использования флагов `FLASHW_TIMER` и `FLASHW_STOP`.

Активизация окна

Теперь рассмотрим другой, гораздо более гибкий способ привлечения внимания к окну приложения. Он базируется на использовании API-функции `SetForegroundWindow`. Данная функция принимает один единственный параметр — дескриптор окна. Если выполняется ряд условий, то окно в заданным дескриптором будет выведено на передний план, и пользовательский ввод будет направлен в это окно. Функция возвращает нулевое значение, если не удалось сделать окно активным.

В приведенном ниже примере окно активизируется при каждом срабатывании таймера (листинг 1.4).

Листинг 1.4. Активизация окна

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  SetForegroundWindow(Handle);
end;
```

В операционных системах старше Windows 95 и Windows NT 4.0 введен ряд ограничений на действие функции `SetForegroundWindow`. Приведенный выше пример

как раз и является одним из случаев недружественного использования активизации окна — но это всего лишь пример.

Чтобы активизировать окно, процесс не должен быть фоновым либо должен иметь право устанавливать активное окно, назначенное ему другим процессом с таким правом, и т. д. Все возможные нюансы в пределах одного трюка рассматривать не имеет смысла. Стоит отметить, что в случае, когда окно не может быть активизировано, автоматически вызывается функция `FlashWindow` для окна приложения (эта функция заставляет мигать кнопку приложения на Панели задач). Поэтому даже при возникновении ошибки при вызове функции `SetForegroundWindow` приложение, нуждающееся во внимании, не останется незамеченным.

Окно приложения

Обратите внимание на то, что название приложения, указанное на кнопке, расположенной на Панели задач, совпадает в названием проекта (можно установить на вкладке Application окна Project options, вызываемого командой меню Project ▸ Options). Но это название не совпадает с заголовком главной формы приложения. Взгляните на приведенный ниже код, который можно найти в DPR-файле (несущественная часть опущена).

```
program ...
begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end;
```

В конструкторе класса `TApplication`, экземпляром которого является глобальная переменная `Application` (ее объявление находится в модуле `Forms`), происходит неявное создание главного окна приложения. Заголовок именно этого окна отображается на Панели задач (кстати, этот заголовок можно также изменить с помощью свойства `Title` объекта `Application`). Дескриптор главного окна приложения можно получить с помощью свойства `Handle` объекта `Application`.

Главное окно приложения делается невидимым (оно имеет нулевую высоту и ширину), чтобы создавалась иллюзия его отсутствия и можно было считать, что главной является именно форма, создаваемая первой.

Для подтверждения вышесказанного можно отобразить главное окно приложения, используя следующий код (листинг 1.5).

Листинг 1.5. Отображение окна приложения

```
procedure TForm1.Button1Click(Sender: TObject);
begin
```

```
  SetWindowPos(Application.Handle, 0, 0, 0, 200, 100,
               SWP_NOZORDER or SWP_NOMOVE);
end;
```

В результате использования этого кода ширина окна станет равной 200, а высота 100, и вы сможете посмотреть на главное окно. Кстати, можно заметить, что при активизации этого окна (например, щелчке кнопкой мыши на заголовке) фокус ввода немедленно передается созданной первой, то есть главной, форме.

Теперь должно стать понятно, почему не мигала кнопка приложения при применении функций `FlashWindow` или `FlashWindowEx` к главной форме приложения. Недостаток этот теперь можно легко устранить, например, следующим образом (листинг 1.6).

Листинг 1.6. Мигание кнопки приложения на Панели задач

```
procedure TForm1.Button2Click(Sender: TObject);
var
  fl: FLASHWINFO;
begin
  fl.cbSize := SizeOf(fl);
  fl.hwnd := Application.Handle;
  fl.dwFlags := FLASHW_ALL;
  fl.uCount := 10;
  fl.dwTimeout := 200;
  FlashWindowEx(fl);
end;
```

В данном случае одновременно инвертируется и заголовок окна приложения. Убедиться в этом можно, предварительно выполнив код листинга 1.5. Наконец, чтобы добиться одновременного мигания кнопки приложения на Панели задач и заголовка формы (произвольной, а не только главной), можно выполнить следующий код (листинг 1.7).

Листинг 1.7. Мигание кнопки приложения и инверсия заголовка формы

```
procedure TForm1.Button3Click(Sender: TObject);
var
  fl: FLASHWINFO;
begin
  //Мигание кнопки
  fl.cbSize := SizeOf(fl);
  fl.hwnd := Application.Handle;
```

```
  fl.dwFlags := FLASHW_TRAY;
  fl.uCount := 10;
  fl.dwTimeout := 200;
  FlashWindowEx(fl);
  //Инверсия заголовка
  fl.cbSize := SizeOf(fl);
  fl.hwnd := Handle;
  fl.dwFlags := FLASHW_CAPTION;
  fl.uCount := 10;
  fl.dwTimeout := 200;
  FlashWindowEx(fl);
end;
```

В данном случае инвертируется заголовок формы Form1. Кнопка на Панели задач может не только мигать, но и, например, быть скрыта или показана, когда в этом есть необходимость. Так, для скрытия кнопки приложения можно применить API-функцию ShowWindow:

```
ShowWindow(Application.Handle, SW_HIDE);
```

Чтобы показать кнопку приложения, можно функцию ShowWindow вызвать с равным SW_NORMAL вторым параметром,.

Полупрозрачные окна

В Windows 2000 впервые появилась возможность использовать прозрачность окон (в англоязычной документации такие полупрозрачные окна называются Layered windows). Сделать это можно, задав дополнительный стиль окна (о назначении и использовании оконных стилей вы можете узнать из материалов, представленных в гл. 2). Здесь не будет рассматриваться использование API-функций для работы с полупрозрачными окнами, так как их поддержка реализована для форм Delphi. Соответствующие свойства включены в состав класса TForm.

- AlphaBlend — включение или выключение прозрачности. Если параметр имеет значение True, то прозрачность включена, если False — выключена.
- AlphaBlendValue — значение, обратное прозрачности окна (от 0 до 255). Если параметр имеет значение 0, то окно полностью прозрачно, если 255 — непрозрачно.

Рис. 1.1. Свойства для настройки прозрачности в окне Object Inspector

Значения перечисленных свойств можно изменять как с помощью окна Object Inspector, так и во время выполнения программы (рис. 1.1).