

## IDŹ DO

PRZYKŁADOWY ROZDZIAŁ



SPIS TREŚCI

## KATALOG KSIĄŻEK

KATALOG ONLINE

ZAMÓW DRUKOWANY KATALOG

## TWÓJ KOSZYK

DODAJ DO KOSZYKA

## CENNIK I INFORMACJE

ZAMÓW INFORMACJE  
O NOWOŚCIACH

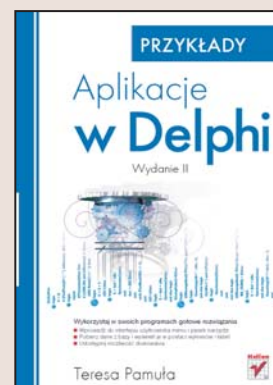
ZAMÓW CENNIK

## CZYTELNIA

FRAGMENTY KSIĄŻEK ONLINE

# Aplikacje w Delphi. Przykłady. Wydanie II

Autor: Teresa Pamuła  
ISBN: 83-246-0576-2  
Format: B5, stron: 360



Delphi to jedno z najpopularniejszych środowisk programistycznych. Koncepcja połączenia znanego i łatwego do opanowania języka Pascal z możliwościami projektowania obiektowego oraz techniką tworzenia aplikacji z komponentów, dzięki której można błyskawicznie zbudować szkielet programu, zyskała ogromne uznanie wśród programistów. Delphi ma ogromną liczbę użytkowników, a możliwości najnowszych wersji sprawiają, że narzędzie to wykorzystywane jest coraz powszechniej. Ostatnia edycja Delphi umożliwia także tworzenie aplikacji dla platformy .NET oraz aplikacji internetowych.

„Aplikacje w Delphi. Przykłady. Wydanie II” to książka przedstawiająca wyłącznie praktyczne aspekty wykorzystania tego środowiska programistycznego. Dzięki zaprezentowanym w niej przykładom nauczysz się stosować komponenty, za pomocą których można utworzyć elementy interfejsu użytkownika (menu rozwijane, paski narzędzi i listy wyboru), a także dowiesz się, jak pobierać dane z plików zewnętrznych i bazy danych oraz tworzyć nowe komponenty.

- Elementy projektu w Delphi 2006
- Tworzenie menu rozwijanego
- Paski narzędzi
- Formatowanie i wyświetlanie danych na ekranie
- Okna dialogowe i okna komunikatów
- Edytor tekstu zbudowany na podstawie komponentów
- Wyświetlanie tabel i wykresów
- Komunikacja z bazami danych
- Praca z systemem plików

**Sprawdź, jak inni rozwiązali problemy, które napotkałeś, programując w Delphi**

Wydawnictwo Helion  
ul. Kościuszki 1c  
44-100 Gliwice  
tel. 032 230 98 63  
e-mail: [helion@helion.pl](mailto:helion@helion.pl)



# Spis treści

|                                                                                      |           |
|--------------------------------------------------------------------------------------|-----------|
| <b>Wprowadzenie .....</b>                                                            | <b>7</b>  |
| <b>Rozdział 1. Projektowanie aplikacji w Delphi 2006 .....</b>                       | <b>9</b>  |
| Środowisko zintegrowane — Delphi IDE .....                                           | 10        |
| Elementy projektu aplikacji .....                                                    | 12        |
| Standardowe właściwości komponentów .....                                            | 14        |
| Standardowe zdarzenia .....                                                          | 14        |
| <b>Rozdział 2. Podstawowe składniki aplikacji .....</b>                              | <b>19</b> |
| Okno aplikacji .....                                                                 | 19        |
| Ikona aplikacji .....                                                                | 24        |
| Wyświetlanie napisów .....                                                           | 24        |
| Rodzaje przycisków, podobieństwa i różnice .....                                     | 29        |
| Etykiety i przyciski .....                                                           | 33        |
| <b>Rozdział 3. Menu główne i podręczne, pasek narzędzi .....</b>                     | <b>37</b> |
| Wielopoziomowe menu główne .....                                                     | 37        |
| Przyporządkowanie poleceń opcjom menu .....                                          | 39        |
| Menu podręczne .....                                                                 | 44        |
| „Polskie litery” w nazwach poleceń menu .....                                        | 45        |
| Pasek narzędzi TToolBar .....                                                        | 46        |
| <b>Rozdział 4. Wprowadzanie danych, formatowanie i wyświetlanie na ekranie .....</b> | <b>49</b> |
| Liczby — funkcje konwersji i formatowanie. Przecinek czy kropka? .....               | 50        |
| Daty — funkcje konwersji i formatowanie daty i czasu .....                           | 52        |
| Systemowe separatory liczb i daty .....                                              | 54        |
| Wprowadzanie danych za pomocą okienek edycyjnych TEdit .....                         | 55        |
| Wprowadzanie danych za pomocą okienek InputBox i InputQuery .....                    | 62        |
| Sposoby zabezpieczania programu przed błędami przy wprowadzaniu danych .....         | 63        |
| Maskowanie danych wejściowych .....                                                  | 63        |
| Blokowanie możliwości wprowadzania niektórych znaków, np. liter lub cyfr .....       | 65        |
| Korzystanie z funkcji konwersji StrToIntDef (z wartością domyślną) .....             | 66        |
| Zmiana zawartości okienka TEdit za pomocą suwaka TScrollBar .....                    | 66        |
| Zmiana zawartości okienka TEdit za pomocą komponentu TUpDown .....                   | 67        |
| Stosowanie instrukcji obsługi wyjątków .....                                         | 68        |
| Obliczenia. Wybrane funkcje modułu Math .....                                        | 70        |

|                                                                                                      |            |
|------------------------------------------------------------------------------------------------------|------------|
| <b>Rozdział 5. Okienka komunikatów .....</b>                                                         | <b>73</b>  |
| Wyświetlanie komunikatów z napisami stałymi w języku systemowym                                      |            |
| — MessageBox .....                                                                                   | 74         |
| Wyświetlanie komunikatów za pomocą funkcji ShowMessage, MessageDlg, MessageDlgPos .....              | 75         |
| <b>Rozdział 6. Okienka dialogowe z karty Dialogs .....</b>                                           | <b>81</b>  |
| <b>Rozdział 7. Listy wyboru — TListBox i TComboBox .....</b>                                         | <b>87</b>  |
| Dodawanie elementów do listy .....                                                                   | 89         |
| Wybieranie elementów z listy .....                                                                   | 90         |
| Sposoby wyświetlania elementów listy .....                                                           | 93         |
| Lista z nazwami czcionek .....                                                                       | 95         |
| Blokowanie edycji dla listy TComboBox .....                                                          | 95         |
| Czytanie i zapisywanie zawartości listy do pliku dyskowego .....                                     | 96         |
| <b>Rozdział 8. Prosty edytor — komponent TMemo .....</b>                                             | <b>99</b>  |
| Kopiowanie, wycinanie i wklejanie tekstu .....                                                       | 101        |
| Czytanie i zapisywanie tekstu do pliku .....                                                         | 102        |
| Wyświetlanie informacji o położeniu kursora .....                                                    | 102        |
| Automatyczne kasowanie linii niezawierających liczb lub wybranych znaków .....                       | 103        |
| Wyświetlanie współrzędnych kursora w polu TMemo i zegara na pasku TStatusBar ..                      | 104        |
| <b>Rozdział 9. Grupowanie komponentów .....</b>                                                      | <b>109</b> |
| Pola opcji i pola wyboru .....                                                                       | 109        |
| Komponenty grupujące .....                                                                           | 110        |
| Ramka TBevel .....                                                                                   | 115        |
| <b>Rozdział 10. Komponenty do wyboru daty i czasu TDateTimePicker i TMonthCalendar .....</b>         | <b>117</b> |
| <b>Rozdział 11. Zakładki TTabControl i TPageControl .....</b>                                        | <b>121</b> |
| <b>Rozdział 12. Odmierzanie czasu — komponent TTimer .....</b>                                       | <b>127</b> |
| <b>Rozdział 13. Grafika w Delphi — korzystanie z metod obiektu TCanvas .....</b>                     | <b>131</b> |
| Wyświetlanie prostych figur geometrycznych i tekstu .....                                            | 132        |
| Rysowanie „trwałe” — zdarzenie OnPaint .....                                                         | 139        |
| Rysowanie po formularzu i bitmapie .....                                                             | 141        |
| Przykłady animacji w Delphi .....                                                                    | 144        |
| <b>Rozdział 14. Wyświetlanie obrazów — komponent TImage .....</b>                                    | <b>151</b> |
| Rysowanie po obrazie .....                                                                           | 153        |
| Binaryzacja obrazu .....                                                                             | 154        |
| Skalowanie obrazów .....                                                                             | 157        |
| Przeglądanie wczytanych obrazów .....                                                                | 160        |
| Wyświetlanie zawartości listy obrazków TImageList .....                                              | 161        |
| <b>Rozdział 15. Tabelaryzacja danych — komponent TStringGrid, TDrawGrid i TValueListEditor .....</b> | <b>163</b> |
| Ustalanie podstawowych parametrów tabeli .....                                                       | 166        |
| Wypełnianie tabeli danymi .....                                                                      | 168        |
| Wybieranie komórek tabeli .....                                                                      | 170        |
| Filtrowanie wprowadzanych danych .....                                                               | 173        |
| Niestandardowe przejście do kolejnej komórki — klawisz Enter .....                                   | 175        |
| Zmiana koloru i wyrównania tekstu w wybranych komórkach .....                                        | 176        |
| Wyświetlanie tekstu w komórce w dwóch wierszach .....                                                | 180        |

|                                                                                                             |            |
|-------------------------------------------------------------------------------------------------------------|------------|
| Totolotek .....                                                                                             | 182        |
| Tabela i lista .....                                                                                        | 184        |
| Tabela TDrawgrid .....                                                                                      | 185        |
| Tabela TValueListEditor .....                                                                               | 192        |
| <b>Rozdział 16. Graficzna prezentacja danych — komponent TChart .....</b>                                   | <b>197</b> |
| Rysowanie wykresów z wykorzystaniem komponentu TChart .....                                                 | 197        |
| Opis wybranych właściwości, metod i zdarzeń komponentów TChart<br>i TChartSeries .....                      | 199        |
| Wykresy kołowe .....                                                                                        | 202        |
| Wykresy kolumnowe .....                                                                                     | 205        |
| Wykresy funkcji matematycznych .....                                                                        | 208        |
| Formatowanie i skalowanie wykresów .....                                                                    | 212        |
| Posługiwanie się wieloma wykresami .....                                                                    | 218        |
| <b>Rozdział 17. Współpraca programu z plikami dyskowymi .....</b>                                           | <b>223</b> |
| Wybór foldera plików .....                                                                                  | 223        |
| Wyszukiwanie plików .....                                                                                   | 225        |
| Zapisywanie danych z okienek TEdit i tabeli do pliku tekstowego .....                                       | 228        |
| Czytanie danych z pliku tekstowego .....                                                                    | 230        |
| Zapisywanie i odczytywanie danych z tabeli do pliku *.csv .....                                             | 231        |
| Zmiana nazw grupy plików .....                                                                              | 233        |
| Korzystanie ze strumieni .....                                                                              | 234        |
| <b>Rozdział 18. Drukowanie w Delphi .....</b>                                                               | <b>241</b> |
| Drukowanie napisów i tekstu z okienek edycyjnych .....                                                      | 242        |
| Drukowanie tabeli i wykresu .....                                                                           | 243        |
| Drukowanie obrazu .....                                                                                     | 246        |
| Drukowanie tekstu sformatowanego za pomocą komponentu TRichEdit .....                                       | 247        |
| Drukowanie za pomocą Rave Reports .....                                                                     | 253        |
| <b>Rozdział 19. Programy z wieloma oknami .....</b>                                                         | <b>265</b> |
| Wymiana danych i metod między modułami .....                                                                | 265        |
| Program z hasłem .....                                                                                      | 268        |
| Wyświetlanie tytułu programu .....                                                                          | 271        |
| Aplikacje typu MDI .....                                                                                    | 273        |
| <b>Rozdział 20. Posługiwanie się wieloma komponentami tego samego typu.<br/>    Operatory Is i As .....</b> | <b>277</b> |
| Wprowadzanie i kasowanie danych dla kilku okienek edycyjnych .....                                          | 278        |
| Przypisywanie grupie komponentów tej samej procedury obsługi zdarzenia .....                                | 279        |
| Wyświetlanie informacji o numerach kontroltek, ich nazwach i klasach .....                                  | 283        |
| <b>Rozdział 21. Tablice dynamiczne .....</b>                                                                | <b>285</b> |
| <b>Rozdział 22. Dynamiczne tworzenie komponentów .....</b>                                                  | <b>289</b> |
| Wyświetlanie kontroltek i przypisywanie zdarzeniom procedur obsługi .....                                   | 290        |
| Przykłady dynamicznego tworzenia wykresów .....                                                             | 295        |
| Tworzenie menu w czasie działania programu .....                                                            | 298        |
| <b>Rozdział 23. Definiowanie nowych klas komponentów .....</b>                                              | <b>301</b> |
| Klasa tabel z wyrównaniem zawartości komórek do prawej strony .....                                         | 301        |
| Klasa okienek z właściwością Alignment .....                                                                | 303        |
| Instalowanie nowych komponentów na palecie komponentów .....                                                | 305        |
| Instalacja nowego komponentu w Borland Delphi 2006 .....                                                    | 309        |
| Nowy komponent do ankiety .....                                                                             | 311        |
| Nowy komponent złożony z komponentów standardowych .....                                                    | 313        |

|                                                                |            |
|----------------------------------------------------------------|------------|
| <b>Rozdział 24. Podstawowe operacje na bazach danych .....</b> | <b>317</b> |
| Przeglądanie istniejących baz danych w formacie .dbf .....     | 319        |
| Tworzenie własnej bazy danych .....                            | 321        |
| Modyfikowanie bazy .....                                       | 323        |
| Filtrowanie rekordów bazy danych .....                         | 324        |
| Wyszukiwanie rekordów .....                                    | 326        |
| Sortowanie .....                                               | 327        |
| Rysowanie wykresów na podstawie danych z bazy .....            | 327        |
| Obliczanie średniej ze wszystkich wartości danego pola .....   | 329        |
| Biblioteka — przykład relacyjnej bazy danych .....             | 329        |
| Logiczne połączenie tabel .....                                | 332        |
| Drukowanie danych za pomocą programu Rave Reports .....        | 333        |
| <b>Skorowidz .....</b>                                         | <b>341</b> |

## Rozdział 13.

# Grafika w Delphi — korzystanie z metod obiektu TCanvas

Niektóre komponenty posiadają właściwość typu obiektowego TCanvas (tzw. płótno). Są to m.in.: TForm, TImage, TPaintBox, TBitmap, TComboBox, TStringGrid, TListBox, TPrinter.

Właściwość Canvas zawiera metody, które pozwalają na rysowanie na tych komponentach za pomocą linii różnych figur, kolorowanie powierzchni oraz wyświetlanie tekstu. Możliwa jest również zmiana koloru i grubości linii, koloru i wzoru wypełnienia, atrybutów czcionki itd.

Rysowanie za pomocą metod obiektu Canvas różnych obiektów może być przydatne do zmiany cech niektórych komponentów, np. TStringGrid czy TChart, a także przy drukowaniu formularza i tekstu.

Wybrane właściwości obiektu TCanvas:

Brush — określa wzór lub kolor wypełnienia figur (tzw. pędzel);

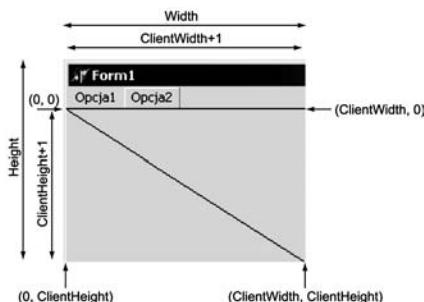
Font — krój czcionki dla wyświetlanych napisów;

Pen — określa cechy kreślonych linii: grubość, styl, kolor (tzw. pióro);

PenPos — określa współrzędne kursora graficznego.

Podstawowymi parametrami większości procedur i funkcji graficznych są współrzędne punktu na komponencie, po którym rysujemy. Lewy górny róg ma współrzędne (0, 0), a prawy dolny najczęściej (Width, Height). Na rysunku 13.1 przedstawiono współrzędne okna formularza, które wykorzystano w zadaniach tego rozdziału.

**Rysunek 13.1.**  
Formularz  
z zaznaczonymi  
wartościami  
współrzędnych  
wierzchołków  $(x, y)$



## Wyświetlanie prostych figur geometrycznych i tekstu

Proste figury i tekst możemy wyświetlić na formularzu, korzystając z procedur i funkcji obiektu typu `TCanvas` — tabela 13.1. Właściwości takiego obiektu umożliwiają m.in. zmianę grubości i stylu rysowanych linii, zmianę koloru i wzoru wypełnienia figur oraz wybór kroju i stylu czcionki dla tekstu.

**Tabela 13.1.** Wybrane metody obiektu `TCanvas`

| Metoda                                          | Znaczenie                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>Kolor:=Canvas.Pixels[x,y]</code>          | Za pomocą funkcji <code>Pixels</code> można odczytać kolor piksela w miejscu o współrzędnych $(x, y)$ — zmienna <code>Kolor</code> jest typu <code>TColor</code> .                                                                                                                                                                                                                                                                                    |
| <code>Canvas.Pixels[10,20]:=clRed</code>        | Ta sama funkcja wywołana w ten sposób powoduje wyświetlenie na formularzu czerwonego punktu w miejscu o współrzędnych $[10, 20]$ — współrzędną poziomą ( $x$ ) liczymy od lewej do prawej, a współrzędną pionową od góry w dół. Współrzędne lewego górnego wierzchołka to $(0, 0)$ .                                                                                                                                                                  |
| <code>MoveTo(x,y: integer)</code>               | Przenosi kursor graficzny do punktu o współrzędnych $x, y$ .                                                                                                                                                                                                                                                                                                                                                                                          |
| <code>LineTo(x,y:integer)</code>                | Rysuje linię od bieżącej pozycji kursora graficznego do punktu o współrzędnych $x, y$ .                                                                                                                                                                                                                                                                                                                                                               |
| <code>Rectangle(x1, y1, x2, y2: Integer)</code> | Procedura rysuje prostokąt wypełniony standardowym kolorem pędzla ( <code>Canvas.Brush.Color</code> ).                                                                                                                                                                                                                                                                                                                                                |
| <code>Ellipse(x1, y1, x2, y2: Integer)</code>   | Procedura rysuje elipsę (lub koło) — parametrami są współrzędne dwóch przeciwległych wierzchołków prostokąta (kwadratu), w który elipsa jest wpisana.                                                                                                                                                                                                                                                                                                 |
| <code>Polyline(Points: array of TPoint)</code>  | Procedura rysuje linię łamaną lub wielokąt. Parametrami są współrzędne punktów, które zostaną połączone linią. Jeśli współrzędne punktu pierwszego i ostatniego są takie same, to rysowany jest wielokąt; w przeciwnym razie linia łamana, np. procedura:<br><code>Polyline([Point(40, 10), Point(20, 60), Point(70, 30), Point(10, 30), Point(60, 60), Point(40, 10)])</code><br>narysuje gwiazdę pięcioramienną (patrz pomoc dla <i>polyline</i> ). |

**Tabela 13.1.** Wybrane metody obiektu TCanvas — ciąg dalszy

| Metoda                                                                  | Znaczenie                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Polygon(Points: array of TPoint)                                        | Procedura umożliwia narysowanie wielokąta wypełnionego bieżącym kolorem i stylem pędzla. Przykładowo instrukcje:<br><pre>Canvas.Brush.Color = clRed; Canvas.Polygon([Point(10, 10), Point(30, 10),   ─Point(130, 30), Point(240, 120)]);</pre> spowodują narysowanie czworokąta wypełnionego kolorem czerwonym. Współrzędne punktu pierwszego i ostatniego nie muszą się pokrywać, ponieważ procedura i tak łączy na końcu punkt ostatni z punktem pierwszym. |
| Refresh                                                                 | Odświeżanie formularza — procedura kasuje wszystkie obiekty rysowane za pomocą metod obiektu Canvas i nieumieszczone w procedurze obsługi zdarzenia OnPaint.                                                                                                                                                                                                                                                                                                  |
| Draw(x, y: integer;<br>Graphic: TGraphic)                               | Rysuje obraz określony parametrem Graphic w miejscu o współrzędnych x i y (przykład 13.14).                                                                                                                                                                                                                                                                                                                                                                   |
| Arc(x1, y1, x2, y2, x3, y3,<br>x4, y4: integer)                         | Rysuje krzywą eliptyczną w prostokącie o współrzędnych (x1, y1; x2, y2), od punktu o współrzędnych (x3, y3) do punktu (x4, y4).                                                                                                                                                                                                                                                                                                                               |
| TextOut(x, y: integer; const<br>Text: string)                           | Wyświetla tekst od punktu o współrzędnych x, y — lewy górny róg prostokąta zawierającego tekst; Text to parametr w postaci tekstu stałego w apostrofach, np. 'Ala ma kota', lub zmienna zawierająca łańcuch znaków, np. a:='Ala ma kota' (const w nagłówku procedury oznacza podobne wywołanie jak w przypadku wartości, lecz umożliwia bardziej efektywne wykorzystanie pamięci).                                                                            |
| CopyRect(const Dest: TRect;<br>Canvas: TCanvas; const Source:<br>TRect) | Kopiuje część obrazu z jednego płótna na inne płótno.                                                                                                                                                                                                                                                                                                                                                                                                         |
| FillRect(const Rect: TRect)                                             | Rysowanie prostokąta wypełnionego bieżącym kolorem i wzorem.                                                                                                                                                                                                                                                                                                                                                                                                  |
| FloodFill(X, Y: Integer; Color:<br>TColor; FillStyle: TFillStyle)       | Wypełnianie tzw. powodziowe obiektów.                                                                                                                                                                                                                                                                                                                                                                                                                         |
| FrameRect(const Rect: TRect)                                            | Rysowanie obwodu prostokąta.                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Pie(X1, Y1, X2, Y2, X3, Y3,<br>X4, Y4: Integer);                        | Rysowanie wycinka koła.                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| RoundRect(X1, Y1, X2, Y2,<br>X3, Y3: Integer)                           | Rysowanie prostokąta z zaokrąglonymi narożnikami.                                                                                                                                                                                                                                                                                                                                                                                                             |
| StretchDraw(const Rect:<br>TRect; Graphic: TGraphic)                    | Dopasowanie rysunku do obszaru danego prostokąta.                                                                                                                                                                                                                                                                                                                                                                                                             |
| TextHeight(const Text: string):<br>Integer                              | Funkcja zwraca wysokość tekstu w pikselach.                                                                                                                                                                                                                                                                                                                                                                                                                   |
| TextOut(X, Y: Integer;<br>const Text: string)                           | Procedura wyświetla napis na komponencie posiadającym właściwość TCanvas.                                                                                                                                                                                                                                                                                                                                                                                     |
| TextRect(Rect: TRect; X, Y:<br>Integer; const Text: string)             | Procedura wyświetla napis w prostokącie, którego współrzędne są podane w postaci typu TRect (pierwszy parametr). Procedura była wykorzystywana przy formatowaniu komórek tabeli.                                                                                                                                                                                                                                                                              |
| TextWidth(const Text: string):<br>Integer                               | Funkcja zwraca szerokość tekstu w pikselach.                                                                                                                                                                                                                                                                                                                                                                                                                  |

Oprócz wymienionych metod zdefiniowane są metody, które korzystają z tzw. mechanizmów niskopoziomowych i właściwości `Handle` komponentu, np. instrukcja:

```
kol:=GetNearestColor( Form1.Canvas.Handle, RGB(125,67,22));
```

spowoduje przypisanie zmiennej `kol` koloru najbardziej zbliżonego do podanego — w przypadku gdy bieżący tryb graficzny nie posiada koloru typu RGB.

### Przykład 13.1.

Wyświetl na etykiecie współrzędne prawego dolnego wierzchołka formularza — lewy górny ma współrzędne (0, 0).

### Rozwiązanie

Wstaw etykietę `TLabel`. Współrzędne prawego dolnego wierzchołka formularza możemy odczytać, korzystając z właściwości `ClientWidth` i `ClientHeight` formularza. Należy wpisać np. w procedurze obsługi zdarzenia `OnClick` etykiety instrukcję:

```
Label1.Caption:=IntToStr(ClientWidth)+' '+IntToStr(ClientHeight);
```

lub użyć funkcji `GetClientRectangle`, która zwraca wartość typu `TRect` określającą współrzędne dwóch przeciwległych wierzchołków formularza:

```
R:=Form1.GetClientRectangle; //R typu TRect można zadeklarować jako zmienną  
lokalną  
Label1.Caption:=Inttostr(R.Right)+' '+ Inttostr(R.Bottom);
```

### Przykład 13.2.

Na środku formularza wyświetl punkt koloru czerwonego, przy czym nie może w tym miejscu znajdować się inny obiekt (np. przycisk), bo wyświetlony piksel zostanie przez ten obiekt przesłonięty.

### Rozwiązanie

Poniższą instrukcję wpisz np. w procedurze obsługi przycisku:

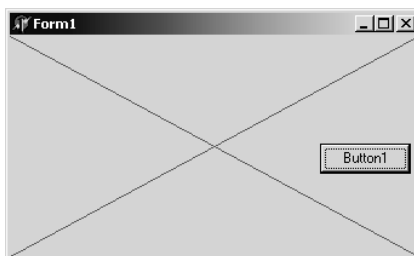
```
Canvas.Pixels[ClientWidth div 2, ClientHeight div 2]:=clRed;
```

### Przykład 13.3.

Narysuj linie koloru czerwonego będące przekątnymi formularza — rysunek 13.2.

#### Rysunek 13.2.

*Formularz  
z przekątnymi  
pozostającymi  
po zmianie  
jego rozmiaru*



## Rozwiązanie

Poniższe instrukcje wpisz np. w procedurze obsługi przycisku.

Pierwsza przekątna:

```
Canvas.Pen.Color:=clRed; //zmiana koloru pióra na czerwony
//przesunięcie kursora graficznego do punktu o współrzędnych (0,0)
Canvas.MoveTo(0,0);
//narysowanie linii od bieżącego położenia kursora graficznego do punktu z prawego
➡dolnego wierzchołka
Canvas.LineTo(ClientWidth, ClientHeight);
```

Narysuj drugą przekątną.

Aby przekątne pozostały na formularzu podczas zmiany jego rozmiaru, należy wykonać dwa zdarzenia: OnPaint i OnResize. W procedurach obsługi tych zdarzeń powinny znaleźć się instrukcje, jak w procedurach poniżej:

```
procedure TForm1.FormPaint(Sender: TObject);
begin
    Canvas.Pen.Color:=clRed;
    Canvas.MoveTo(0,0);
    Canvas.LineTo(ClientWidth, ClientHeight);
    Canvas.MoveTo(ClientWidth,0);
    Canvas.LineTo(0, ClientHeight);
end;

i

procedure TForm1.FormResize(Sender: TObject);
begin
    Refresh; // przy zmianie rozmiaru okna
            // kasowane są poprzednie przekątne
end;
```

## Przykład 13.4.

Wyświetl na formularzu punkty rozmieszczone losowo i o losowych kolorach.

## Rozwiązanie

Wstaw przycisk i w procedurze obsługi zdarzenia OnClick wpisz odpowiednie instrukcje:

```
//Losowe punkty
procedure TForm1.Button2Click(Sender: TObject);
var i:integer;
begin
    for i:=1 to 10000 do
        Canvas.Pixels[Random(ClientWidth), Random(ClientHeight)]:=
            RGB( Random(255),Random(255), Random (255 ) );
    end;
```

### Przykład 13.5.

Wyświetl na formularzu trzy różne prostokąty — ramkę, prostokąt wypełniony kolorem `Brush.Color`, prostokąt z zaokrąglonymi brzegami.

### Rozwiązanie

W procedurze obsługi przycisku wpisz instrukcje jak poniżej:

```
procedure TForm1.Button3Click(Sender: TObject);
var
  prost: TRect;
begin
  prost:= Rect(200,10,300,100);
  Canvas.Brush.Color := clBlack;
  //ramka
  Canvas.FrameRect(prost);
  Canvas.Brush.Color := clGreen;
  //prostokąt wypełniony
  Canvas.Rectangle(200,120,300,210);
  //prostokąt z zaokrąglonymi brzegami
  Canvas.RoundRect(200,230,300,320,20,20);
end;
```

### Przykład 13.6.

Wyświetl na środku formularza napis `Zadania z Delphi` w kolorze niebieskim, o rozmiarze czcionki równym 36 pkt, bez tła — rysunek 13.3.

#### Rysunek 13.3.

*Napis na środku formularza*



### Rozwiązanie

W procedurze wykorzystano funkcje zwracające szerokość i wysokość napisu oraz rozmiary formularza — i na tej podstawie obliczono współrzędne lewego górnego wierzchołka wyświetlanego napisu:

```
procedure TForm1.Button2Click(Sender: TObject);
var x,y:integer;
begin
  Canvas.Font.Name:='Arial';
  Canvas.Font.Color:=clBlue;
```

```
Canvas.Font.Size:=24;  
Canvas.Brush.Style:=bsClear;  
x:=ClientWidth-Canvas.TextWidth('Zadania  
z Delphi');  
y:=ClientHeight-Canvas.TextHeight('Z');  
Canvas.TextOut(x div 2, y div 2, 'Zadania z Delphi');  
end;
```

### Przykład 13.7.

Narysuj elipsę o maksymalnych wymiarach na formularzu.

### Rozwiązanie

W procedurze obsługi przycisku wpisz instrukcję:

```
//elipsa wpisana w prostokąt o rozmiarach formularza  
Canvas.Ellipse(0,0, ClientWidth, ClientHeight);
```

### Przykład 13.8.

Narysuj na formularzu trójkąt o zielonym obwodzie i żółtym wypełnieniu.

### Rozwiązanie

I sposób — z wykorzystaniem procedury PolyLine i FloodFill:

```
procedure TForm1.Button6Click(Sender: TObject);  
begin  
  Canvas.Brush.Color:=clYellow;  
  Canvas.Pen.Color:=clGreen;  
  //rysowanie trójkąta  
  Canvas.Polyline([Point(20,20),Point(200,20),Point(110,100),Point(20,20)]);  
  Canvas.FloodFill(100,25,clgreen,fsborder); //procedura wypełnia obiekt narysowany  
  //kolorem zielonym, wewnątrz którego znajduje się punkt o współrzędnych (100,25)  
end;
```

II sposób — z wykorzystaniem procedury Polygon, rysującej wielokąt wypełniony bieżącym kolorem pędzla (Brush). Współrzędne ostatniego punktu nie muszą pokrywać się ze współrzędnymi punktu pierwszego wielokąta, ponieważ pierwszy punkt jest automatycznie łączony z ostatnim:

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
  Canvas.Brush.Color:=clYellow;  
  Canvas.Pen.Color:=clGreen;  
  Canvas.Polygon([Point(20,20),Point(200,20),Point(110,100)]);  
end;
```

### Przykład 13.9.

Wyświetl na formularzu linie rysowane różnymi stylami.

## Rozwiązanie

Wstaw przycisk TButton. W procedurze obsługi zdarzenia OnClick przycisku wpisz instrukcje, jak w poniższej procedurze:

```
//style linii
procedure TForm1.Button1Click(Sender: TObject);
var x,y:integer;
begin
  x := 210;
  y := y+10;//y - zmienna globalna
  Canvas.MoveTo(x,y);
  x := Random(ClientWidth - 10);
  y := Random(ClientHeight - 10);
  Canvas.Pen.Color := RGB(Random(256),Random(256),Random(256));
  case Random(5) of
    0: Canvas.Pen.Style := psSolid;
    1: Canvas.Pen.Style := psDash;
    2: Canvas.Pen.Style := psDot;
    3: Canvas.Pen.Style := psDashDot;
    4: Canvas.Pen.Style := psDashDotDot;
  end;
  Canvas.LineTo(x+200, y);
end;
```

## Przykład 13.10.

Wyświetl na formularzu prostokąt malowany różnymi stylami pędzla po każdym kliknięciu przycisku.

## Rozwiązanie

Wstaw przycisk TButton. W procedurze obsługi zdarzenia OnClick przycisku wpisz instrukcje, jak w poniższej procedurze:

```
//style pędzla
procedure TForm1.Button2Click(Sender: TObject);
begin
  Refresh; //kasuje poprzedni prostokąt
  Canvas.Brush.Color :=RGB(Random(256),Random(256),Random(256)); //kolorem pędzla
                                                                    // malowane są wzory

  case Random(7) of
    0: Canvas.Brush.Style := bsClear;
    1: Canvas.Brush.Style := bsSolid;
    2: Canvas.Brush.Style := bsBDiagonal;
    3: Canvas.Brush.Style := bsFDiagonal;
    4: Canvas.Brush.Style := bsCross;
    5: Canvas.Brush.Style := bsDiagCross;
    6: Canvas.Brush.Style := bsHorizontal;
    7: Canvas.Brush.Style := bsVertical;
  end;
  Canvas.Rectangle(0,0, 200,100);
end;
```

## Rysowanie „trwałe” — zdarzenie OnPaint

Instrukcje zawierające metody obiektu `Canvas` można umieszczać w procedurach obsługi zdarzenia `OnClick` dla przycisków, dla formularza i innych komponentów. Można również korzystać z innych zdarzeń komponentów. Jednak tylko niektóre z nich umożliwiają tzw. „trwałe” rysowanie, czyli rysowanie odnawiane po każdej zmianie, np. po zmianie rozmiaru okna i przykryciu w ten sposób części obiektów graficznych. Dla okna formularza korzysta się w tym celu ze zdarzenia `OnPaint`. Dla innych komponentów podobne zdarzenia mają inne nazwy. Przedstawiono je w tabeli 13.2.

**Tabela 13.2.** Zdarzenia umożliwiające rysowanie „trwałe”

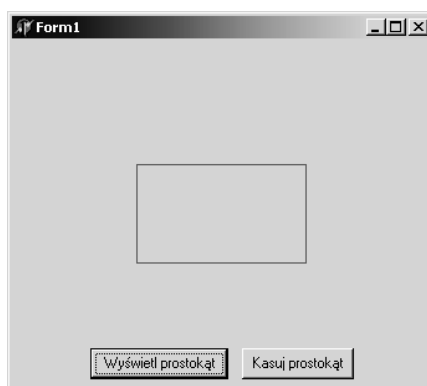
| Zdarzenie                  | Znaczenie                                                                                                                                                                                                                                                               |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>OnPaint</code>       | Zdarzenie dla formularza generowane każdorazowo, gdy zawartość okna formularza wymaga odświeżenia. Sytuacja taka ma miejsce przy tworzeniu okna formularza, a także wtedy, gdy np. jedno okno zostanie przesłonięte innym oknem lub gdy następuje zmiana jego rozmiaru. |
| <code>PaintBoxPaint</code> | Odpowiednik zdarzenia <code>OnPaint</code> dla komponentu <code>PaintBox</code> .                                                                                                                                                                                       |
| <code>OnDrawCell</code>    | Zdarzenie występujące dla komponentu typu <code>TDrawGrid</code> i <code>TStringGrid</code> — umożliwia „trwałe” rysowanie obiektów i wyświetlanie tekstu w komórkach.                                                                                                  |
| <code>OnAfterDraw</code>   | Zdarzenie dla komponentu typu <code>TChart</code> , odpowiednik zdarzenia <code>OnPaint</code> .                                                                                                                                                                        |

### Przykład 13.11.

Narysuj na formularzu prostokąt koloru czerwonego, tak aby nie kasował się po przykryciu okna formularza innym oknem. Prostokąt powinien rysować się po kliknięciu przycisku i kasować po kliknięciu drugiego przycisku — rysunek 13.4.

### Rysunek 13.4.

*Rysowanie  
i kasowanie  
prostokąta  
na formularzu*



### Rozwiązanie

Wstaw dwa przyciski `TButton`.

Gdyby instrukcję rysującą prostokąt umieścić w procedurze obsługi zdarzenia OnPaint, to prostokąt byłby na formularzu bezpośrednio po uruchomieniu programu. Dlatego procedurę obsługi tego zdarzenia z nową instrukcją należy wywołać za pomocą przycisku.

Szkielet procedury FormPaint można uzyskać klikając dwukrotnie w oknie Inspektora Obiektów z prawej strony zdarzenia OnPaint. Później trzeba jednak wykasować w Inspektorze Obiektów tę nazwę, dlatego aby instrukcje w procedurze obsługi zdarzenia OnPaint nie wykonywały się bezpośrednio po uruchomieniu programu.

W przykładzie pokazano, jak wykonać takie zadanie.

```
//obsługa zdarzenia OnPaint dla formularza rysuje prostokąt, wywoływana programowo
//może mieć inną nazwę
procedure TForm1.FormPaint(Sender: TObject);
begin
  Canvas.Rectangle(100,100,ClientWidth-100,ClientHeight-100);
end;

//procedura rysuje prostokąt koloru czerwonego i przypisuje procedurze obsługi
//zdarzenia OnPaint procedurę FormPaint
procedure TForm1.Button1Click(Sender: TObject);
begin
  Canvas.Pen.Color:=clRed;
  Canvas.Rectangle(100,100,ClientWidth-100,ClientHeight-100);
  OnPaint:=FormPaint;//przypisanie procedurze obsługi zdarzenia procedury rysującej
                        //prostokąt
end;

// odłączenie procedury FormPaint od zdarzenia OnPaint – wykasowanie prostokąta
procedure TForm1.Button2Click(Sender: TObject);
begin
  OnPaint:=nil; //ta instrukcja spowoduje, że rysunek prostokąta nie będzie odnawiany
  Refresh;      //procedura ta kasuje prostokąt
end;
```

### Przykład 13.12.

Wypełnij formularz bitmapą, np. *kawa.bmp*.

### Rozwiązanie

W procedurze obsługi zdarzenia OnPaint dla formularza wpisz instrukcje, jak w procedurze poniżej.

Zadeklaruj zmienną globalną lub pole klasy TForm1 (w sekcji public):

```
var Bitmap: TBitmap;

procedure TForm1.FormPaint(Sender: TObject);
var x, y: Integer;
begin
  y := 0;
```

```
while y < Height do
begin
  x := 0;
  while x < Width do
  begin
    Canvas.Draw(x, y, Bitmap);
    x := x + Bitmap.Width;
  end;
  y := y + Bitmap.Height;
end;
end;
```

W metodzie Form1.FormCreate dopisz instrukcje:

```
Bitmap:=TBitmap.Create;
Bitmap.LoadFromFile('C:\WINDOWS\kawa.bmp');
```

## Rysowanie po formularzu i bitmapie

### Przykład 13.13.

Napisz program umożliwiający rysowanie po formularzu po naciśnięciu lewego przycisku myszy.

### Rozwiązanie

Wykorzystano zdarzenie OnMouseDown występujące po naciśnięciu przycisku myszy na komponencie i zdarzenie OnMouseMove występujące przy przesuwaniu kursora myszy nad komponentem.

Wpisz instrukcje, jak w poniższych procedurach obsługi zdarzeń OnMouseDown i OnMouseMove.

```
procedure TForm1.FormMouseDown(Sender: TObject;Button: TMouseButton; Shift:
TShiftState; X, Y: Integer);
begin
  Canvas.MoveTo(x,y);
end;

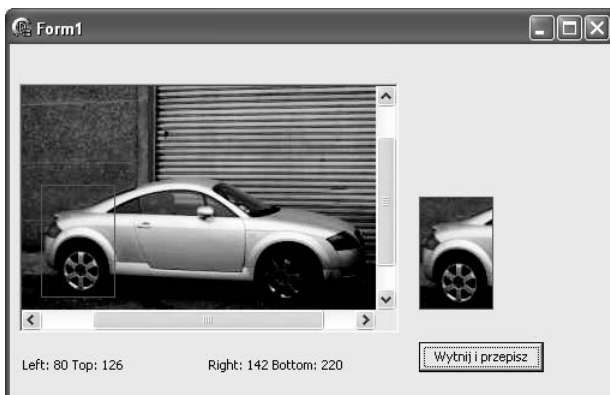
procedure TForm1.FormMouseMove(Sender: TObject;Shift: TShiftState; X, Y: Integer);
begin
  if ssLeft in Shift then //czy lewy przycisk myszy wciśnięty
    Canvas.LineTo(x,y);
end;
```

### Przykład 13.14.

Napisz program umożliwiający rysowanie prostokątów po komponencie typu TImage i wyświetlanie na etykietach współrzędnych zaznaczonego prostokąta — rysunek 13.5.

**Rysunek 13.5.**

*Wycinanie  
prostokątnego  
obszaru z obrazu  
typu .BMP*

**Rozwiązanie**

Na formularzu wstaw komponent TScrollBox i wewnątrz niego komponent TImage z zakładki *Additional* tak, aby lewe górne wierzchołki obu komponentów pokrywały się. Dla komponentu TImage ustaw właściwość Autosize na true i za pomocą właściwości Picture załaduj do niego obraz typu .BMP (nie .JPG). Zmień kształt kursora na +. Dodaj do formularza jeszcze dwie etykiety TLabel.

Wykorzystaj zdarzenie OnMouseDown występujące po naciśnięciu przycisku myszy na komponencie TImage i zdarzenie OnMouseMove występujące przy przesuwaniu kursora myszy nad komponentem TImage.

Zadeklaruj zmienne globalne:

```
StartX, StartY, StopX, StopY: Integer;
bmppom: TBitmap;
```

W procedurze obsługi zdarzenia OnCreate dla formularza zapamiętaj obraz .BMP z komponentu Image1 w zmiennej pomocniczej bmppom.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  bmppom:=TBitmap.Create;
  bmppom.Assign(Image1.Picture.Bitmap);
end;
```

**Zapis:**

```
bmppom.Assign(Image1.Picture.Bitmap);
```

oznacza kopiowanie mapy bitowej (obrazu) z komponentu Image1 do zmiennej bmppom, natomiast zapis:

```
bmppom:=Image1.Picture.Bitmap;
```

oznacza przypisanie wskaźnika do bitmapy z komponentu Image1 do zmiennej bmppom. Jeśli więc nie kopiujemy bitmapy, tylko wskaźnik do obszaru zajmowanego przez nią, to nie można wykorzystać tej zmiennej do zapamiętania poprzedniego obrazu w komponencie Image1.

W procedurze obsługi zdarzenia `OnMouseDown` dla komponentu `Image1` zapamiętaj początkowe współrzędne kursora myszy.

```
procedure TForm1.Image1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  StartX := X;
  StartY := Y;
end;
```

W procedurze obsługi zdarzenia `OnMouseMove` dla komponentu `Image1` zaznaczane są prostokątne obszary.

```
procedure TForm1.Image1MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
  //przesunięcie kursora poza obraz powoduje, że procedura obsługi zdarzenia zwraca
  //wartości spoza zakresu komponentu Image1
  if x<0 then x:=0;
  if y<0 then y:=0;
  if x>Image1.ClientWidth then x:=Image1.ClientWidth;
  if y>Image1.ClientHeight then y:=Image1.ClientHeight;

  with Image1 do
  if ssLeft in Shift then //czy lewy myszy przycisk wciśnięty
  begin
    Image1.Picture.Bitmap:=bmppom; //odtworzenie poprzedniej bitmapy
    Image1.Canvas.Pen.Color:=clred; //ustawienie koloru obrisu
    Image1.Canvas.Brush.Style:=bsclear; //styl bez wypełnienia
    //rysowanie prostokąta
    Image1.Picture.Bitmap.Canvas.Rectangle(StartX, StartY, X, Y);
    StopX:=X; StopY:=Y; //zapamiętanie współrzędnych potrzebnych do przykładu 13.15.
```

//pozostała część kodu wyświetla na etykiecie współrzędne dwóch przeciwległych wierzchołków i nie jest niezbędna do prawidłowego działania programu

```
//wyświetlanie współrzędnych dwóch przeciwległych wierzchołków
if X > StartX then //rysowanie w prawo od punktu startowego
begin
  Label1.Caption:= ' Left: ' + IntToStr(StartX);
  Label2.Caption:= ' Right: ' + IntToStr(X);
end;
if X<=StartX then //else
begin //rysowanie w lewo od punktu startowego
  Label1.Caption:= ' Left: ' + IntToStr(X);
  Label2.Caption:= ' Right: ' + IntToStr(StartX);
end;

if Y > StartY then //rysowanie w dół od punktu startowego
begin
  Label1.Caption:=Label1.Caption+' Top: ' +
    IntToStr(StartY);
  Label2.Caption:=Label2.Caption+' Bottom: ' + IntToStr(Y);
end
else
begin //rysowanie w górę od punktu startowego
  Label1.Caption :=Label1.Caption+ ' Top: ' + IntToStr(Y);
```

```

Label2.Caption :=Label2.Caption+ ' Bottom: ' +
                                IntToStr(StartY);
end;
end;
end;

```

### Przykład 13.15.

Napisz program umożliwiający po kliknięciu przycisku przekopiowanie zaznaczonego obszaru w przykładzie 13.14 do drugiego komponentu typu TImage — rysunek 13.5.

### Rozwiązanie

Na formularzu wstaw przycisk TButton. W procedurze obsługi zdarzenia OnClick wpisz instrukcje, jak w procedurze poniżej.

```

procedure TForm1.Button2Click(Sender: TObject);
var i,j,w,h:integer;
begin
  w:=ABS(StopX-StartX); //obliczenie szerokości zaznaczonego obszaru
  h:=ABS(StopY-StartY); //obliczenie wysokości zaznaczonego obszaru

  //ustalenie szerokości i wysokości bitmapy komponentu Image2
  Image2.Picture.Bitmap.Width:=w;
  Image2.Picture.Bitmap.Height:=h;

  //ustalenie współrzędnych lewego górnego wierzchołka zaznaczonego prostokąta
  if StartX>StopX then StartX:=StopX;
  if StartY>StopY then StartY:=StopY;

  //przepisanie zaznaczonego obszaru z komponentu Image1 do komponentu Image2
  //piksel po pikselu
  for i := 0 to w-1 do
    for j := 0 to h-1 do
      Image2.Picture.Bitmap.Canvas.Pixels[i,j]:=
        Image1.Picture.Bitmap.Canvas.Pixels[StartX+i,StartY+j];
  end;
end;

```

## Przykłady animacji w Delphi

W programowaniu stosuje się różne techniki animacji. Jednym z prostszych sposobów jest rysowanie obiektu, następnie kasowanie i ponowne rysowanie w innym miejscu. Wadą tego rozwiązania jest trudność w uzyskaniu płynności ruchu obiektów.

Inna metoda polega na zastosowaniu dwóch obszarów, na których rysujemy. W danej chwili widoczny jest tylko jeden z nich. Drugi jest wówczas modyfikowany i wyświetlany w miejsce pierwszego dopiero po zakończeniu operacji.

W zadaniach przykładowych zastosowano pierwszy sposób animacji. Udało się uzyskać odpowiednią płynność ruchu obiektów, dlatego nie wykorzystano sposobu z użyciem dwóch obszarów rysowania.

**Przykład 13.16.**

Wykonaj następującą animację: kółko o średnicy 30 pkt przesuwa się od lewego do prawego brzegu formularza i z powrotem.

**Rozwiązanie**

W procedurze obsługi przerwania od *Timera* wpisz:

```
{ $J+ }  
procedure TForm1.Timer1Timer(Sender: TObject);  
const x1:integer=0;  
      y1:integer=100;  
      krok:integer=5;  
begin  
    //kasowanie obiektu  
    Canvas.Brush.color:=Color; //kolor formularza  
    Canvas.Pen.color:=Color;   //kolor pióra  
    Form1.Canvas.Ellipse(x1,y1,x1+30,y1+30);  
    //rysowanie kółka kolorem czerwonym  
    Canvas.Brush.color:=clRed;  
    x1:=x1+krok;  
    Canvas.Ellipse(x1,y1,x1+30,y1+30);  
    if x1+30>= Clientwidth then krok:=-krok;  
    if x1<=0 then krok:=-krok;  
end;
```

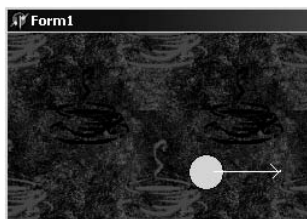
Dyrektywa `{ $J+ }` przed treścią procedury włącza opcję kompilatora umożliwiającą zmianę wartości stałych typowanych (ang. *Assignable typed constants*). Opcja ta powinna być standardowo włączona, ale jeśli nie mamy pewności, lepiej dodać dyrektywę `{ $J+ }`.

**Przykład 13.17.**

Wykonaj animację tak jak w zadaniu poprzednim, gdy formularz jest wypełniony wzorem — rysunek 13.6.

**Rysunek 13.6.**

*Animacja z tłem*

**Rozwiązanie**

Na formularzu umieść przycisk `TButton` i komponent `TTimer`. Właściwość `Interval` ustaw na 200 ms, a właściwość `Enabled` na `false`. Treść procedur obsługi przycisku i przerwania od *Timera* przedstawiono poniżej.

Zadeklaruj zmienną globalną:

```

var Bitmap, Bitmap1: TBitmap;
//załadowanie obrazu .BMP do zmiennej Bitmap
procedure TForm1.FormCreate(Sender: TObject);
begin
    //W procedurze FormCreate należy przeczytać bitmapę z pliku.
    Bitmap:=TBitmap.Create;
    Bitmap.LoadFromFile('c:\windows\kawa.bmp');
end;

// procedura pobiera prostokątny fragment formularza i uruchamia Timer
procedure TForm1.Button1Click(Sender: TObject);
var x,y:integer;
begin
    // utworzenie obiektu Bitmap1
    Bitmap1 := TBitmap.Create;
    Bitmap1.Width:=ClientWidth;
    Bitmap1.Height:=30;
    //pobranie prostokątnego wycinka formularza - obszaru, po którym będzie się poruszało
    //kółko
    for x:=0 to ClientWidth-1 do
        for y:=0 to 29 do
            Bitmap1.Canvas.Pixels[x,y]:=Form1.Canvas.Pixels[x,y+100];
    Timer2.Enabled:=true; //w Inspektorze Obiektów zablokuj Timer2
end;

{$J+} // procedura obsługi przerwania od Timera - rysowanie i kasowanie obiektu co 200 ms
procedure TForm1.Timer2Timer(Sender: TObject);
const x1:integer=0;
        y1:integer=100;
        krok:integer=5;
var x,y:integer;
begin
    //jeśli zwiększymy rozmiar formularza, to trzeba w procedurze obsługi zdarzenia
    //OnResize jeszcze raz pobrać bitmapę
    Canvas.Draw(0,y1,Bitmap1); //wyświetlenie wcześniej pobranego paska formularza,
                                //kasowanie obiektu

    //rysowanie kółka
    Canvas.Ellipse(x1,y1,x1+30,y1+30);
    x1:=x1+krok;
    if x1+29>=ClientWidth then krok:=-krok;
    if x1<=0 then krok:=-krok;
end;

//wypełnianie formularza bitmapą
procedure TForm1.FormPaint(Sender: TObject);
var x, y: Integer;
begin
    y := 0;
    while y < Height do
        begin
            x := 0;
            while x < Width do
                begin
                    Canvas.Draw(x, y, Bitmap);
                    x := x + Bitmap.Width;
                end;
            end;

```

```

        y := y + Bitmap.Height;
    end;
end;

// procedura TFormDestory zwalnia pamięć
// zajmowaną przez bitmapy
procedure TForm1.FormDestory(Sender: TObject);
begin
    Bitmap.Free;
    Bitmap1.Free;
end;

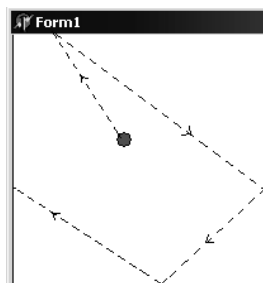
```

### Przykład 13.18.

Wykonaj animację polegającą na przemieszczaniu się kulki w losowych kierunkach w prostokątnym obszarze o wymiarach (0, 0, 200, 200). Wykorzystaj komponent TPaintBox z zakładki *System* — rysunek 13.7.

#### Rysunek 13.7.

*Animacja  
niebieskiej kulki*



### Rozwiązanie

Na formularzu umieść komponent TPaintBox i TTimer. Komponent TPaintBox jest stosowany do wyświetlania (kreślenia) grafiki, która ma być ograniczona do obszaru prostokątnego. Korzystając z komponentu TPaintBox, programista nie musi kontrolować, czy obszar ten nie został przekroczony — jeśli narysowany obiekt nie mieści się wewnątrz komponentu TPaintBox, to zostaje obcięty. Dodatkowo zawarty w nim rysunek możemy przesuwać po formularzu, zmieniając właściwości Left i Top tego komponentu. Procedura przedstawiona poniżej działa poprawnie z komponentem TPaintBox i bez niego — wtedy kulka przesuwa się po formularzu.

W zadaniu można również dodać przycisk, który będzie włączał zegar (animację) po wpisaniu w procedurze obsługi instrukcji Timer1.Enabled:=true; (wcześniej należy zegar zablokować w okienku Inspektora Obiektów — Enabled=true).

```

{$J+}
procedure TForm1.Timer1Timer(Sender: TObject);
const x:integer=6;
      y:integer=6;
      krokx:integer=6;
      kroky:integer=6;
begin
    with PaintBox1.Canvas do

```

```

begin
  //czyszczenie prostokąta
  Brush.Color:=clWhite;
  Rectangle(0,0,200,200);
  //obliczenie współrzędnych
  x:=x+kroxx;
  y:=y+kroky;
  //rysowanie koła w kwadracie o boku
  // równym 6 pikseli
  Brush.Color:=clBlue;
  Ellipse(x-6, y-6, x+6, y+6);
  if (x>Paintbox1.Width-6) then
  begin
    kroxx:=6+Random(5);
    kroxx:=-kroxx;
  end;
  if (y>Paintbox1.Height-6) then
  begin
    kroky:=6+Random(5);
    kroky:=-kroky;
  end;
  if (x<=6) then kroxx:=-kroxx;
  if (y<=6) then kroky:=-kroky;
end; //with
end;

```

Korzystając z Inspektora Obiektów właściwości `Interval` komponentu `Timer1` przypisz wartość 50.

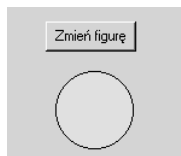
### Przykład 13.19.

Umieść na formularzu komponent typu `TButton` i `TShape`. Zadaniem przycisku jest wyświetlanie po każdym kliknięciu na przemian kółka lub prostokąta.

Po naciśnięciu klawiszy strzałek komponent `Shape` przesuwa się zgodnie z kierunkiem strzałki — rysunek 13.8.

#### Rysunek 13.8.

*Przesuwanie koła  
za pomocą  
klawiszy strzałek*



Aby klawisze strzałek nie były przechwytywane przez komponent `Button1`, należy ustawić dla niego właściwość `TabStop` na `false`.

### Rozwiązanie

Wstaw komponenty `TButton` i `TShape`. Dla komponentu `TShape` ustaw właściwość `Shape` na `stCircle` i właściwość `Brush\Color` na `clYellow`. W procedurze obsługi kliknięcia przycisku wpisz instrukcje, jak poniżej:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if Shape1.Shape=stCircle then Shape1.Shape:=stRectangle
  else Shape1.Shape:=stCircle;
  Form1.ActiveControl:=nil;
end;
```

W celu sprawdzenia klawiszy strzałek wykorzystaj zdarzenie OnKeyDown dla formularza. Treść procedury obsługi tego zdarzenia przedstawiono poniżej:

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  case Key of
    vk_Right: Shape1.Left:=Shape1.Left+10; //strzałka w prawo
    vk_Left:  Shape1.Left:=Shape1.Left-10; //strzałka w lewo
    vk_Up:    Shape1.Top:=Shape1.Top-10; //strzałka w górę
    vk_Down: Shape1.Top:=Shape1.Top+10; //strzałka w dół
  end;
end;
```