

Przykład 11.4 Kolejny problem to przygotowanie planszy do wykonania następującego zadania. Należy zmienić kolor całej planszy na inny niż początkowy. Naciśnięcie przycisku myszki na danej kratce powoduje zmianę jej koloru, ale żeby nie było zbyt łatwo, zmienia się również kolor krutek sąsiednich (przyległych w wierszu lub kolumnie). Ilustrują to rysunki 11.7 i 11.8 (projekt *zamiana.dpr*).

Rysunek 11.7: Sytuacja początkowa w projekcie *zamiana.dpr*

Naszą planszę umieścimy w komponencie *DrawGrid1* (ustawione właściwości tego komponentu pozostawiamy do przeanalizowania Czytelnikowi). Każdą kratkę tego komponentu będziemy wypełniać zgodnie z wartościami ustawionymi w tablicy `kolor`; 1 oznacza kolor zielony, a 2 to kolor czerwony. Ponieważ maksymalne rozmiary komponentu *DrawGrid1* to 10*10, tablica `kolor` powinna mieć rozmiary 12*12. Ponadto konieczne należy przesunąć przyporządkowanie pomiędzy wartościami umieszczonymi w tablicy `kolor`, a kolorami w komponencie *DrawGrid1*.

Rysunek 11.8: Sytuacja po zmianie koloru danej kratki

Otóż kolorami w pierwszym wierszu (o indeksie równym 0) komponentu *DrawGrid1* powinien sterować drugi wiersz (o indeksie równym 1) tablicy `kolor` itd. Takie samo przyporządkowanie obowiązuje dla kolumn, to znaczy, że kolorami w pierwszej kolumnie (o indeksie równym 0) komponentu *DrawGrid1* steruje druga kolumna (o indeksie równym 1) tablicy `kolor` itd. Pozwala to uniknąć kłopotów związanych ze sprawdzaniem, czy można zmienić kolor kratek sąsiednich dla kratki leżącej na obrzeżu komponentu *DrawGrid1*.

Deklarację tablicy `kolor` umieszczamy w części **private** deklaracji klasy `TForm1`.

```
private
    kolor: array[1..12,1..12] of boolean;
```

Metoda wykonywana przy tworzeniu formularza to `FormCreate`.

```
procedure TForm1.FormCreate(Sender: TObject);
```

```

var
  i,j: integer;
begin
  { w całej planszy ma być kolor zielony }
  for i:=1 to 12 do
  for j:=1 to 12 do
    kolor[i,j] := true;
  { początkowe rozmiary planszy to 4*4 }
  DrawGrid1.RowCount := 4;
  DrawGrid1.ColCount := 4;
end;

```

Wypełnienie odpowiednim kolorem każdej kratki komponentu *DrawGrid1* realizuje metoda *DrawGrid1DrawCell*. Jak już doskonale wiemy, w momencie zajścia zdarzenia *OnDrawCell* metoda ta jest wykonywana dla każdej kratki. Sterowanie wypełnianiem odbywa się na podstawie wartości umieszczonych w tablicy *kolor*, ale należy pamiętać o przesunięciu przyporządkowania (numer wiersza i numer kolumny należy zwiększyć o 1).

```

procedure TForm1.DrawGrid1DrawCell(Sender: TObject;
ACol, ARow: Integer; Rect: TRect; State: TGridDrawState);
{ metoda wypełniania odpowiednim kolorem każdej kratki
komponentu DrawGrid1 }
begin
  { numer wiersza i kolumny należy zwiększyć o 2 ponieważ
  tablica kolorów jest przesunięta względem siatki }
  if kolor[ARow+2,ACol+2] then
    begin
      { wypełnienie kratki kolorem zielonym }
      DrawGrid1.Canvas.Brush.Color := clGreen;
      DrawGrid1.Canvas.FillRect(Rect);
    end
  else
    begin
      { wypełnienie kratki kolorem czerwonym }
      DrawGrid1.Canvas.Brush.Color := clRed;
      DrawGrid1.Canvas.FillRect(Rect);
    end;
end;
end;

```

Po naciśnięciu klawisza myszki na danej kratce komponentu *DrawGrid1* chcemy zmienić kolor tej kratki i wszystkich krutek sąsiednich w wierszu lub kolumnie. Operację tę realizuje metoda *DrawGrid1SelectCell* wykonywana po zajściu zdarzenia *OnSelectCell*. Zwróćmy jeszcze uwagę, że ze względu na przesunięcie przyporządkowania pomiędzy tablicą *kolor*, a kolorami umieszczonymi w komponencie *DrawGrid1* nie musimy sprawdzać, czy nie wyszliśmy poza tablicę. Na przykład dla kratki leżącej powyżej wybranej jest wykonywana następująca instrukcja:

```
kolor[ARow,ACol-1] := not kolor[ARow,ACol-1];
```

Nie ma konieczności sprawdzania, czy przypadkiem wartość *ACol-1* nie jest równa -1, ponieważ wcześniej zostały wykonane instrukcje:

```
ARow := ARow + 2;  
ACol := ACol + 2;
```

A oto treść metody *DrawGrid1SelectCell*:

```
procedure TForm1.DrawGrid1SelectCell(Sender: TObject;  
ACol, ARow: Integer; var CanSelect: Boolean);  
{ metoda wykonywana po wybraniu danej komórki - naciśnięcie  
przycisku myszki na tej komórce }  
begin  
  { zwiększenie numeru wiersza i kolumny }  
  ARow := ARow + 2;  
  ACol := ACol + 2;  
  
  { zamiana koloru danej kratki zapamiętana na razie  
  w tablicy kolor }  
  kolor[ARow,ACol] := not kolor[ARow,ACol];  
  { zamiana koloru kratki leżącej powyżej }  
  kolor[ARow,ACol-1] := not kolor[ARow,ACol-1];  
  { zamiana koloru kratki leżącej poniżej }  
  kolor[ARow,ACol+1] := not kolor[ARow,ACol+1];  
  { zamiana koloru kratki leżącej na lewo }  
  kolor[ARow-1,ACol] := not kolor[ARow-1,ACol];  
  { zamiana koloru kratki leżącej na prawo }  
  kolor[ARow+1,ACol] := not kolor[ARow+1,ACol];
```

```
    { wymuszenie ponownego wypełnienia komponentu DrawGrid1
      kolorami zapamiętanymi w tablicy kolor }
    DrawGrid1.Visible := false;
    DrawGrid1.Visible := true;
end;
```

W projektowanej aplikacji jest oczywiście niezbędne umożliwienie powrotu do sytuacji początkowej. A oto odpowiednia metoda wykonywana po naciśnięciu klawisza myszki na przycisku *Button1*:

```
procedure TForm1.Button1Click(Sender: TObject);
{ powrót do sytuacji początkowej }
var
  i,j: integer;
begin
  { ma być kolor zielony w całej planszy }
  for i:=1 to 12 do
  for j:=1 to 12 do
    kolor[i,j] := true;

    { wymuszenie ponownego wypełnienia komponentu DrawGrid1
      kolorami zapamiętanymi w tablicy kolor }
    DrawGrid1.Visible := false;
    DrawGrid1.Visible := true;
end;
```

Na zaprojektowanym formularzu są umieszczone dwie listwy przewijania *ScrollBar1* oraz *ScrollBar2* sterujące liczbą kolumn i wierszy w komponencie *DrawGrid1*. Po przesunięciu suwaka w tych komponentach zachodzi zdarzenie *OnChange* i dlatego należy wypełnić treścią poniższe dwie metody.

```
procedure TForm1.ScrollBar1Change(Sender: TObject);
{ ustawienie liczby kolumn w komponencie DrawGrid1 }
begin
  DrawGrid1.ColCount := ScrollBar1.Position;
end;
```

```
procedure TForm1.ScrollBar2Change(Sender: TObject);
{ ustawienie liczby wierszy w komponencie DrawGrid1 }
```

```
begin  
    DrawGrid1.RowCount := ScrollBar2.Position;  
end;
```