Dodatek B Grafika w C++ Builder

TCanvas

W środowisku C++ Builder grafika oparta jest na idei "płótna" (ang. canvas). Płótno jest obiektem klasy ***TCanvas*** i jak każdy obiekt w C++ Builder ma swoje właściwości i metody. Obiektu klasy ***TCanvas*** nie używa się jednak samego. ***Canvas*** jest obiektem należącym do innego obiektu i służy jako powierzchnia do rysowania (używa się go w takich obiektach jak np. `Form, Bitmap, Image, PaintBox, Printer`). Niezależnie od tego jaki obiekt posiada płótno (***Canvas***) kreślenie grafiki zawsze odbywa się tak samo, tak więc do rysowania na dowolnym obiekcie zawsze będzie służył ten sam kod. Obiekty graficzne kreślimy na płótnie pracując w układzie współrzędnych kartezjańskich. W C++ Builder początek układu współrzędnych znajduje się w lewym górnym rogu, oś `Y` biegnie z góry do dołu, natomiast oś `X` – od lewej do prawej krawędzi płótna.

Klasa ***TCanvas*** zawiera funkcje wyświetlające prymitywy (linie, łuki okręgów, elipsy, wielokąty, tekst).

Przykład: funkcja rysująca linię i wyświetlająca tekst

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
   Canvas->Pen->Color = clBlue;
   Canvas->MoveTo(10,10);
   Canvas->LineTo(100,100);
   Canvas->Brush->Color = clBtnFace;
   Canvas->Font->Name = "Arial";
   Canvas->TextOut( Canvas->PenPos.x, Canvas->PenPos.y,"Hello World!" );
}
```

Funkcje graficzne w C++ Builder

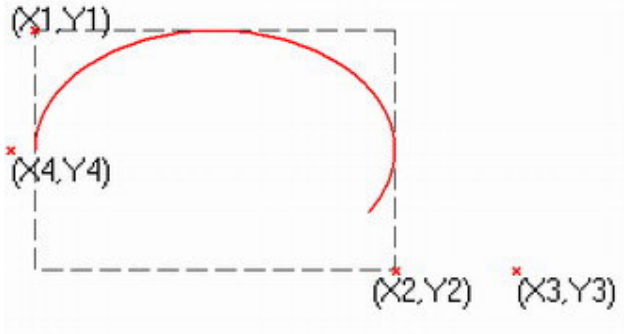
Arc

Funkcja rysująca łuk elipsy wpisanej w prostokąt o danych wierzchołkach i danych końcach łuku

```
void __fastcall Arc(int X1, int Y1, int X2, int Y2, int X3, int Y3,
                    int X4, int Y4);
```

Przykład:

```
{
  TRect R = GetClientRect();  // Pobiera wspolrzedne  biezacego okna
  Canvas->Arc(R.Left, R.Top, R.Right, R.Bottom, R.Right, R.Top, R.Left,
              R.Top);
}
```



Ellipse

Funkcja rysująca elipsę wpisaną w prostokąt o danych wierzchołkach (lewym górnym i prawym dolnym)

```
void __fastcall Ellipse(int X1, int Y1, int X2, int Y2);
```

(X1,X2) – współrzędne lewego górnego wierzchołka prostokąta,
(X2,Y2) – współrzędne prawego dolnego wierzchołka prostokąta .

```
void __fastcall Ellipse(TRect Rect);
```

gdzie *Trect* jest następującą strukturą:

```
typedef struct
       { int left;
         int top;
         int right;
         int bottom;
       }TRect;
```

Jeśli prostokąt jest kwadratem, wyświetlony zostanie okrąg.

Przykład:

```
{
  Canvas->Brush->Color = clRed;
  Canvas->Brush->Style = bsDiagCross;
  Canvas->Ellipse(0, 0, 200, 100);
}
```

FloodFill

Funkcja wypełniająca ograniczony obszar bieżącą barwą

```
enum TFillStyle {fsSurface, fsBorder};
void __fastcall FloodFill(int X,int Y, TColor Color, TfillStyle FillStyle);
```

(X,Y) - współrzędne dowolnego punktu leżącego w danym obszarze
fsSurface – kolor obszaru
fsBorder – kolor brzegu obszaru

Przykład:

```
Canvas->FloodFill(ClientWidth/2, ClientHeight/2, clBlack, fsBorder);
```

LineTo

Rysowanie linii od aktualnej pozycji pióra *(PenPos)* do podanej pozycji

```
void __fastcall LineTo(int X, int Y);
```

Przykład:

```
{
 Canvas->MoveTo(0, 0);
Canvas->LineTo(X, Y);
}
```

MoveTo

Przeniesienie pióra do podanej pozycji

```
void __fastcall MoveTo(int X, int Y);
void __fastcall TForm1::FormMouseMove(TObject *Sender, TShiftState Shift,
                                      int X, int Y)
```

Przykład:

```
{
  Canvas->MoveTo(0, 0);
  Canvas->LineTo(X, Y);
}
```

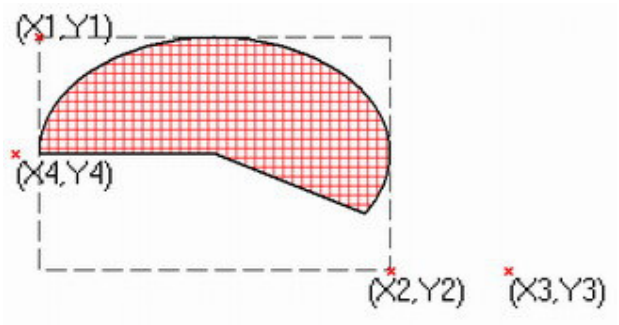
Pie

Rysowanie wycinka elipsy wypełnionego bieżącym stylem i barwą

```
void __fastcall Pie(int X1, int Y1, int X2, int Y2, int X3, int Y3,
                    int X4, int Y4);
```

Przykład:

```
{
Canvas->Pie(100, 100, 400, 300, 500, 300, 80, 200);
}
```



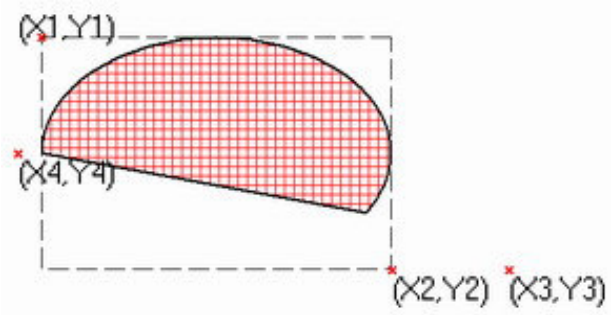
Chord

Rysowanie odcinka elipsy

```
void __fastcall Chord(int X1, int Y1, int X2, int Y2, int X3, int Y3,
                      int X4, int Y4);
```

Przykład:

```
{
Canvas->Chord(100, 100, 400, 300, 500, 300, 80, 200);
}
```



Polygon

Rysowanie wielokąta o podanych wierzchołkach

```
void __fastcall Polygon(const TPoint * Points, const int Points_Size);
```

Przykład:

```
{
```

```
  TPoint points[4];
  points[0] = Point(10,10);
  points[1] = Point(30,10);
  points[2] = Point(130,30);
  points[3] = Point(240,120);
  Canvas->Polygon(points, 3);
}
```

gdzie `Tpoint` jest następującą strukturą:

```
typedef struct
       { int x;
         int y;
       }TPoint;
```

Polyline

Rysowanie łamanej o podanych wierzchołkach

```
void __fastcall Polyline(const Types::TPoint* Points,
                         const int Points_Size);
```

Przykład:

```
{
  Tpoint points[5];
  points[0].x = 40;
  points[0].y = 10;
  points[1].x = 20;
  points[1].y = 60;
  points[2].x = 70;
  points[2].y = 30;
  points[3].x = 10;
  points[3].y = 30;
  points[4].x = 60;
  points[4].y = 60;
  Canvas->Polyline(points,4);
}
```

Rectangle

Rysowanie prostokąta (o podanych lewym górnym i prawym dolnym wierzchołku)

```
void __fastcall Rectangle(int X1, int Y1, int X2, int Y2);
void __fastcall Rectangle(TRect Rect);
```

Przykład:

```
{
int x, y;
Canvas->Rectangle(x, y, x + random(400), y + random(400));
}
```

FillRect

Rysowanie wypełnionego prostokąta bieżącym stylem i barwą wypełnienia

Przykład:

```
Canvas->FillRect(Rect(0,0,100,100));
```

RoundRect

Rysowanie prostokąta o zaokrąglonych wierzchołkach

```
void __fastcall RoundRect(int X1, int Y1, int X2, int Y2, int X3, int Y3);
```

`(X1,Y1)` – lewy górny wierzchołek prostokąta
`(X2,Y2)` – prawy dolny wierzchołek prostokąta
`X3, Y3` – osie elipsy (łuki eliptyczne zaokrąglają wierzchołki prostokąta)

Refresh

Metoda odświeżająca obiekt

```
void __fastcall Refresh(void);
```

Przykładowa procedura wywołania:

```
Form1->Refresh();
```

Metoda odświeża i przywraca stan domyślny formularza - wszystkie piksele otrzymują kolor określony we właściwości `Color` formularza

Brush

Ustalanie koloru i stylu wypełniania

Ustalenie koloru wypełnienia:

```
{
Canvas->Brush->Color = clGreen;
}
```

Ustalenie stylu wypełnienia:

```
{
   Canvas->Brush->Style = bsSolid;
}
```

Predefiniowane style wypełnienia:
`bsSolid, bsClear, bsHorizontal, bsVertical, bsFDiagonal, bsBDiagonal, bsCross, bsDiagCross.`

TextOut

Wypisanie ciagu znaków na ekranie od podanego miejsca

```
void __fastcall TextOut(int X, int Y, const AnsiString Text);
```

Przykład:

```
{
 Canvas->TextOut(100, 100, "Witaj Świecie!");
}
```

Ustalenie koloru tekstu:

```
{
Canvas->Font->Color = clRed;
}
```

Ustalenie rozmiaru czcionki:

```
{
Canvas->Font->Size = 13;
}
```

Ustalenie rodzaju czcionki:

```
{
Canvas->Font->Name = "Times New Roman";
}
```

Ustalenie stylu czcionki:

```
{
Canvas->Font->Style = TFontStyles()<< fsBold << fsUnderline;
}
```

Predefiniowane style czcionki: `fsBold, fsItalic, fsUnderline, fsStrikeOut`

Pixels

Wyświetlanie punktu w podanym kolorze

Przykład:

```
{
 Canvas->Pixels[123][10] = clRed;
}
```

Pen

Sprawdzenie pozycji pióra:

```
{
  TPoint p;
  p=Canvas->PenPos;
//wypisanie pozycji piora
 Canvas->TextOut(200,200, IntToStr(p.x) + "," + IntToStr(p.y));
}
```

Ustalenie koloru pióra (koloru rysowania):

```
{
  Canvas->Pen->Color = clBlue;
}
```

Ustalenie stylu rysowania linii:

```
{
 Canvas->Pen->Style = psSolid;
}
```

Predefiniowane style linii: `psSolid, psDash, psDot, psDashDot,psDashDotDot, psClear`

Ustalenie grubości linii:

```
{
 Canvas->Pen->Width = 3;
}
```

Kolory

`TColor` jest typem danych zawierającym 4-bajtowe liczby całkowite z zakresu `[-0x7FFFFFFF-1 ... 0x7FFFFFFF]` (w zapisie szesnastkowym). Najniższe 3 bajty reprezentują nasycenie koloru trzema podstawowymi barwami w modelu RGB (red, green, blue).
Liczba `0x000000FF` reprezentuje intensywny kolor czerwony (red) bez domieszki pozostałych dwóch barw, podobnie liczba `0x0000FF00` reprezentuje intensywny kolor zielony (green), a liczba `0x00FF0000` – intensywny kolor niebieski (blue). `0x00000000` reprezentuje kolor czarny (brak nasycenia barwami), `0x00FFFFFF` oznacza kolor biały (pełne nasycenie trzema podstawowymi

barwami). Najwyższy bajt decyduje o wyborze palety kolorów – jeśli jest równy `0`, wówczas otrzymujemy kolory z palety systemowej.
Aby zmienić właściwość `Color` wybranego obiektu możemy tej właściwości przypisać liczbę typu `TColor` lub jedną z predefiniowanych stałych z modułu `Graphics`.
Lista stałych kolorów z modułu `Graphics`:

`clNone`	`clMoneyGreen`	`clInactiveCaption`	`clBtnShadow`
`clAqua`	`clNavy`	`clMenu`	`clGrayText`
`clBlack`	`clOlive`	`clWindow`	`clBtnText`
`clBlue`	`clPurple`	`clWindowFrame`	`clInactiveCaptionText`
`clCream`	`clRed`	`clMenuText`	`clBtnHighlight`
`clDkGray`	`clSilver`	`clWindowText`	`cl3DDkShadow`
`clFuchsia`	`clSkyBlue`	`clCaptionText`	`cl3DLight`
`clGray`	`clTeal`	`clActiveBorder`	`clInfoText`
`clGreen`	`clWhite`	`clInactiveBorder`	`clInfoBk`
`clLime`	`clYellow`	`clAppWorkSpace`	`clGradientActiveCaption`
`clLtGray`	`clScrollBar`	`clHighlight`	`clGradientInactiveCaption`
`clMaroon`	`clBackground`	`clHightlightText`	`clDefault`
`clMedGray`	`clActiveCaption`	`clBtnFace`	