Vážení zákazníci,

dovolujeme si Vás upozornit, že na tuto ukázku knihy se vztahují autorská práva, tzv. copyright.

To znamená, že ukázka má sloužit výhradně pro osobní potřebu potenciálního kupujícího (aby čtenář viděl, jakým způsobem je titul zpracován a mohl se také podle tohoto, jako jednoho z parametrů, rozhodnout, zda titul koupí či ne).

Z toho vyplývá, že není dovoleno tuto ukázku jakýmkoliv způsobem dále šířit, veřejně či neveřejně např. umisťováním na datová média, na jiné internetové stránky (ani prostřednictvím odkazů) apod.

redakce nakladatelství BEN – technická literatura
redakce@ben.cz

9 Mapování souřadnic DC

9 Mapování souřadnic DC

Každý, kdo někdy vytvářel program s grafickým výstupem, musel narazit na problém mapování souřadnic. Jde v podstatě o to, jak přizpůsobit souřadnice obrázku, který chceme nakreslit, souřadnicím obrazovky (nebo jiného zařízení).

Jedna z cest, kterou programátoři používají, je vypočítat jakési dělicí poměry mezi fyzickými souřadnicemi zařízení a logickými souřadnicemi obrázku. Tyto poměry slouží pro násobení logických souřadnic, abychom je převedli na fyzické souřadnice a s těmi pak volali funkce pro kreslení.

Druhá, mnohem sofistikovanější možnost je implementovat vlastní funkce, které budou zajišťovat přepočty samy. Tak to provedu v kapitole 9.3.

9.1 Vybrané funkce Win API pro mapování souřadnic

GDI zajišťuje pohodlné mapování souřadnic prostřednictvím několika desítek funkcí, já uvedu jen vybrané funkce, které použiju v příkladu.

9.1.1 SetMapMode

SetMapMode nastaví mapovací mód specifikovaného kontextu zařízení. Mapovací mód definuje jednotky použité pro transformaci logických jednotek na jednotky zařízení a orientaci os x a y. Hlavička:

int SetMapMode(HDC hdc, int fnMapMode);

- **hdc** specifikuje kontext zařízení,
- **fnMapMode** určuje mapovací mód, viz *tab. 9.1*.

Návratová hodnota: Vrací předchozí mód zařízení.

Poznámka: *Před nastavením jakéhokoli mapovacího módu je vybrán implicitní MM_TEXT. To umožňuje pracovat přímo s fyzickými rozměry zařízení. Viz [1] obr. 9.5.*

9.1.2 SetWindowExtEx

SetWindowExtEx nastaví horizontální a vertikální rozsahy pro specifikovaný kontext zařízení. Hlavička:

BOOL SetWindowExtEx(HDC hdc, int nXExtent, int nYExtent, LPSIZE lpSize);

- **hdc** specifikuje kontext zařízení,
- **nXExtent**, **nYExtent** horizontální a vertikální rozsah v logických jednotkách,
- **lpSize** ukazatel na strukturu **SIZE** (viz kapitolu 11.2.2), kam se uloží předchozí rozsahy (pokud uvedete **NULL**, předchozí rozsahy se neuloží).

Tab. 9.1 Mapovací módy

Mód	Význam
MM_ANISOTROPIC	Logické jednotky jsou mapovány na libovolné jednotky s libovolně váženými osami. Je třeba použít funkce **SetWindowExtEx** a **SetViewportExtEx** pro určení jednotek a orientace obou os.
MM_HIENGLISH	Logická jednotka je mapována na 0,001 palce. Osa x roste směrem doprava a osa y roste směrem nahoru.
MM_HIMETRIC	Logická jednotka je mapována na 0,01 mm. Osa x roste směrem doprava a osa y roste směrem nahoru.
MM_ISOTROPIC	Logické jednotky jsou mapovány na libovolné jednotky se stejně váženými osami (jednotky osy x jsou stejné jako jednotky osy y). Pro určení jednotek a orientace je nutno použít funkce **SetWindowExtEx** a **SetViewportExtEx**.
MM_LOENGLISH	Logická jednotka je mapována na 0,01 palce. Osa x roste směrem doprava a osa y roste směrem nahoru.
MM_LOMETRIC	Logická jednotka je mapována na 0,1 mm. Osa x roste směrem doprava a osa y roste směrem nahoru.
MM_TEXT	Logická jednotka je mapována na jeden pixel zařízení. Osa x roste směrem doprava a osa y roste směrem dolů. **Výchozí režim.**
MM_TWIPS	Logická jednotka je mapována na 1/1440 palce (1/20 bodu tiskárny, tj. 1 twip). Osa x roste směrem doprava a osa y roste směrem nahoru.

Návratová hodnota: Při úspěchu vrací **TRUE**, při chybě **FALSE**.

Poznámka: *Volání funkce je neúspěšné pro módy, které mají rozsahy určeny pevně (MM_HIENGLISH, MM_HIMETRIC, MM_LOENGLISH, MM_LOMETRIC, MM_TEXT a MM_TWIPS).*

9.1.3 SetViewportExtEx

SetViewportExtEx nastaví horizontální a vertikální rozsahy výřezu (viewportu) pro určený kontext zařízení. Hlavička:

BOOL SetViewportExtEx(HDC hdc, int nXExtent, int nYExtent, LPSIZE lpSize);

- **hdc** specifikuje kontext zařízení,
- **nXExtent**, **nYExtent** horizontální a vertikální rozsah v jednotkách zařízení (tj. fyzických jednotkách),
- **lpSize** ukazatel na strukturu **SIZE** (viz kapitolu 11.2.2), kam se uloží předchozí rozsahy (pokud uvedete **NULL**, předchozí rozsahy se neuloží).

Návratová hodnota: Při úspěchu vrací **TRUE**, při chybě **FALSE**.

Poznámka: Volání funkce je neúspěšné pro módy, které mají rozsahy určeny pevně (MM_HIENGLISH, MM_HIMETRIC, MM_LOENGLISH, MM_LOMETRIC, MM_TEXT a MM_TWIPS).

9.1.4 SetViewportOrgEx

SetViewportOrgEx nastaví počátek výřezu (viewportu) pro určený kontext zařízení. Hlavička:

BOOL SetViewportOrgEx(HDC hdc, int X, int Y, LPPOINT lpPoint);

- **hdc** specifikuje kontext zařízení,
- **X**, **Y** souřadnice počátku výřezu v jednotkách zařízení (tj. fyzických jednotkách),
- **lpPoint** ukazatel na strukturu **POINT**, kam se uloží předchozí počátek (pokud uvedete **NULL**, předchozí počátek se neuloží).

Návratová hodnota: Při úspěchu vrací **TRUE**, při chybě **FALSE**.

9.1.5 DPtoLP

DPtoLP zkonvertuje souřadnice zařízení na logické souřadnice. Hlavička:

BOOL DPtoLP(HDC hdc, LPPOINT lpPoints, int nCount);

- **hdc** specifikuje kontext zařízení,
- **lpPoint** pole bodů (POINT), které chceme konvertovat,
- **nCount** počet prvků pole **lpPoint**.

Návratová hodnota: Při úspěchu vrací **TRUE**, při chybě **FALSE**.

9.1.6 LPtoDP

L**PtoDP** zkonvertuje logické souřadnice na souřadnice zařízení. Hlavička:

BOOL LPtoDP(HDC hdc, LPPOINT lpPoints, int nCount);

- **hdc** specifikuje kontext zařízení,
- **lpPoint** pole bodů (POINT), které chceme konvertovat,
- **nCount** počet prvků pole **lpPoint**.

Návratová hodnota: Při úspěchu vrací **TRUE**, při chybě **FALSE**.

9.2 Kreslení křivek

Při kreslení grafů získaných měřením nějakých fyzikálních veličin, nebo i pro jiné případy, je třeba provést proložení grafu. Soubor GDI funkcí obsahuje dvě funkce pro kreslení *Bézierových křivek.*

Je pravdou, že tato kapitola nemá mnoho společného s mapováním souřadnic, ale funkci **PolyBezier** použiju v kapitole 9.3, takže pokládám za nutné seznámit s oběma funkcemi právě v tomto okamžiku.

9.2.1 PolyBezier

Nakreslí Bézierovu křivku. Hlavička:

BOOL PolyBezier(HDC hdc, CONST POINT *lppt, DWORD cPoints);

- **hdc** specifikuje kontext zařízení,
- **lppt** pole bodů (POINT), které specifikují křivku,
- **cPoints** počet prvků pole **lppt**.

Návratová hodnota: Při úspěchu vrací **TRUE**, při chybě **FALSE**.

Poznámka: *1. bod v poli udává výchozí bod křivky. 2. a 3. body jsou řídicí body křivky (směrem k nim se křivka ohýbá) a 4. bod je koncový bod křivky. Pokud chceme kreslit výslednou křivku z několika úseků, musíme pro první úsek uvést všechny 4 body, pro další úseky již postačí 3 body (koncový bod předchozího úseku se bere jako výchozí bod následujícího úseku).*

9.2.2 PolyBezierTo

Nakreslí Bézierovu křivku z aktivní polohy pera. Hlavička:

BOOL PolyBezierTo(HDC hdc, CONST POINT *lppt, DWORD cCount);

- **hdc** specifikuje kontext zařízení,
- **lppt** pole bodů (POINT), které specifikují křivku,
- **cPoints** počet prvků pole **lppt**.

Návratová hodnota: Při úspěchu vrací **TRUE**, při chybě **FALSE**.

Poznámka: *Výchozí bod prvního úseku se bere jako aktivní poloha pera. Takže pro popis všech úseků se používají pouze 3 body.*

9.3 Příklad mapování souřadnic

V C++ Builderu založte novou aplikaci. Položkou menu **File|New Unit** *vytvořte další zdrojovou jednotku.*

Nastavte jméno formuláře Name=**MainForm**.

Do formuláře vložte tyto komponenty:

MainMenu: Name=MainMenu, má 2 položky:

Name=MenuSoubor, Caption=Soubor, má 3 položky:

Name=MenuSouborUlozit, Caption=Uložit obrázek…, ShortCut=Ctrl+O,

Caption=– (separátor),

Name=MenuSouborKonec, Caption=Konec, ShortCut=Alt+X,

Name=MenuNastaveni, Caption=Nastavení, má 2 položky:

Menu=MenuNastaveniGraf, Caption=Graf, má 2 podpoložky:

Name=MenuNastaveniGrafProlozeny, Caption=Proložený, RadioItem=true, GroupIndex=10,

Name=MenuNastaveniGrafSpojnicovy, Caption=Spojnicový, RadioItem=true, GroupIndex=10,

Menu=MenuNastaveniBarva, Caption=Barva….

SaveDialog: Name=SaveDialog, Options=[ofOverwritePrompt], DefaultExt=BMP, Filter= Bitmapové obrázky (*.bmp)|*.bmp,

ColorDialog: Name=ColorDialog.

Vygenerujte události **FormPaint** a **FormResize** (události OnPaint a OnResize formuláře) a dále **MenuSouborUlozitClick**, **SouborKonecClick**, **MenuNastaveniGrafProlozenyClick**, **MenuNastaveniGrafSpojnicovyClick** a **MenuNastaveniBarvaClick** (události OnClick položek menu).

Zdrojové soubory uložte pod jmény: MAPMAIN (formulář), MAPPER (jednotka) a MAPMODE (projekt).

Upravte zdrojové soubory podle níže uvedených výpisů.

MAPPER.H:

```
//---------------------------------
#ifndef MapperH
#define MapperH
//---------------------------------
/*TMapper je třída, která zajišťuje elegantní mapování souřadnic
prostřednictvím svých metod*/
class TMapper:public TObject{
public:
 __fastcall TMapper(TCanvas* Canvas,
   int MinX,int MaxX,int MinY,int MaxY,
   int ResX,int ResY,int ClientW=0,int ClientH=0);
 __fastcall ~TMapper();
 long __fastcall TransfX(float X);
 long __fastcall TransfY(float Y);
 long __fastcall NormWidth(long Width);
 long __fastcall NormHeight(long Height);
 void __fastcall MoveTo(float X,float Y);
 void __fastcall LineTo(float X,float Y);
```

```
 void __fastcall DrawNormLine(int dx,int dy);
 void __fastcall TextOut(float X,float Y,
   AnsiString Text,float dx=0,float dy=0);
 void __fastcall SetPenWidth(int Width);
 const int MinX,MaxX,MinY,MaxY;
 const int ResolutionX,ResolutionY;
protected:
 TCanvas *Canvas;
 int ClientW,ClientH;
private:
 void __fastcall SetAnisotropicMapMode();
 int PrevMode;
 SIZE PrevWinExt,PrevViewExt;
 POINT PrevViewOrg;
};
//---------------------------------
#endif
```

MAPPER.CPP:

```
//---------------------------------
#include <vcl\vcl.h>
#pragma hdrstop
#include "Mapper.h"
//---------------------------------
__fastcall TMapper::TMapper(TCanvas* Canvas,
 int MinX,int MaxX,int MinY,int MaxY,
 int ResX,int ResY,int ClientW,int ClientH)
 :MinX(MinX),MaxX(MaxX),MinY(MinY),MaxY(MaxY),
 ResolutionX(ResX),ResolutionY(ResY)
{
/*konstruktor zajistí nastavení mapovacího režimu voláním fce
SetAnisotropicMapMode*/
 TMapper::Canvas=Canvas;
 if((ClientW==0)||(ClientH==0)){
   //pokud uživatel neuvede rozměry plochy, zjisti si je:
   HDC hdc=Canvas->Handle;
   ClientW=GetDeviceCaps(hdc,HORZRES);
   ClientH=GetDeviceCaps(hdc,VERTRES);
 }
 TMapper::ClientW=ClientW;
 TMapper::ClientH=ClientH;
 SetAnisotropicMapMode();
}
//---------------------------------
void __fastcall TMapper::SetAnisotropicMapMode()
{
 HDC hdc=Canvas->Handle;
 //anizotropický mód:
```

```
 PrevMode=SetMapMode(hdc,MM_ANISOTROPIC);
 //rozsahy kontextu:
 SetWindowExtEx(hdc,MaxX*ResolutionX-MinX*ResolutionX,
   MaxY*ResolutionY-MinY*ResolutionY,&PrevWinExt);
 //rozsahy viewportu:
 SetViewportExtEx(hdc,ClientW,-ClientH,&PrevViewExt);
 //počátek viewportu:
 SetViewportOrgEx(hdc,ClientW*(-MinX)/float(MaxX-MinX),
   ClientH*MaxY/float(MaxY-MinY),&PrevViewOrg);
}
//---------------------------------
__fastcall TMapper::~TMapper()
{
//destruktor provede nastavení předchozího mapovacího módu
 HDC hdc=Canvas->Handle;
 SetMapMode(hdc,PrevMode);
 SetWindowExtEx(hdc,PrevWinExt.cx,PrevWinExt.cy,NULL);
 SetViewportExtEx(hdc,PrevViewExt.cx,PrevViewExt.cy,NULL);
 SetViewportOrgEx(hdc,PrevViewOrg.x,PrevViewOrg.y,NULL);
}
//---------------------------------
long __fastcall TMapper::TransfX(float X)
{
//transformace x-ové souřadnice:
 return(X*ResolutionX);
}
//---------------------------------
long __fastcall TMapper::TransfY(float Y)
{
//transformace y-ové souřadnice:
 return(Y*ResolutionY);
}
//---------------------------------
long __fastcall TMapper::NormWidth(long Width)
{
//vrací šířku v log. souřadnicích podle fyz. souřadnic:
  return(Width*float((MaxX-MinX)*ResolutionX)/ClientW+0.5);
}
//---------------------------------
long __fastcall TMapper::NormHeight(long Height)
{
//vrací výšku v log. souřadnicích podle fyz. souřadnic:
  return(Height*float((MaxY-MinY)*ResolutionY)/ClientH+0.5);
}
//---------------------------------
void __fastcall TMapper::MoveTo(float X,float Y)
{
```

```
//upravené MoveTo (volejte tuto metodu a ne metodu kanvasu!)
 Canvas->MoveTo(TransfX(X),TransfY(Y));
}
//---------------------------------
void __fastcall TMapper::LineTo(float X,float Y)
{
//upravené LineTo (volejte tuto metodu a ne metodu kanvasu!)
 Canvas->LineTo(TransfX(X),TransfY(Y));
}
//---------------------------------
void __fastcall TMapper::DrawNormLine(int dx,int dy)
{
/*nakreslí úsečku z aktivního bodu do bodu určeného přírůstky dx
a dy, které se berou ve fyzických souřadnicích:*/
 int x=Canvas->PenPos.x;
 int y=Canvas->PenPos.y;
 Canvas->LineTo(x+NormWidth(dx),y+NormHeight(dy));
}
//---------------------------------
void __fastcall TMapper::SetPenWidth(int Width)
{
//nastaví šířku pera na počet fyzických bodů udaných Width:
 Canvas->Pen->Width=Width*
   float((MaxX-MinX)*ResolutionX)/ClientW+0.5;
}
//---------------------------------
void __fastcall TMapper::TextOut(float X,float Y,
 AnsiString Text,float dx,float dy)
{
//upravené TextOut (volejte tuto metodu a ne metodu kanvasu!)
 Canvas->TextOut(TransfX(X)+dx,TransfY(Y)+dy,Text);
}
```

MAPMAIN.H:

```
//---------------------------------
#ifndef MapMainH
#define MapMainH
//---------------------------------
#include <vcl\Classes.hpp>
#include <vcl\Controls.hpp>
#include <vcl\StdCtrls.hpp>
#include <vcl\Forms.hpp>
#include "Mapper.h"
#include <vcl\Menus.hpp>
#include <vcl\Dialogs.hpp>
//---------------------------------
class TMainForm : public TForm
{
```

```
__published:    // IDE-managed Components
 TMainMenu *MainMenu;
 TMenuItem *MenuSoubor;
 TMenuItem *MenuNastaveni;
 TMenuItem *MenuNastaveniGraf;
 TMenuItem *MenuNastaveniGrafProlozeny;
 TMenuItem *MenuNastaveniGrafSpojnicovy;
 TMenuItem *N1;
 TMenuItem *SouborKonec;
 TMenuItem *MenuSouborUlozit;
 TSaveDialog *SaveDialog;
 TMenuItem *MenuNastaveniBarva;
 TColorDialog *ColorDialog;
 void __fastcall FormPaint(TObject *Sender);
 void __fastcall FormResize(TObject *Sender);
 void __fastcall MenuNastaveniGrafProlozenyClick(TObject
   *Sender);
 void __fastcall MenuNastaveniGrafSpojnicovyClick(TObject
   *Sender);
 void __fastcall MenuSouborUlozitClick(TObject *Sender);
 void __fastcall MenuSouborKonecClick(TObject *Sender);
 void __fastcall MenuNastaveniBarvaClick(TObject *Sender);
private:  // User declarations
 TMapper *Mapper;
 float *Data;        //pole s daty
public:         // User declarations
 __fastcall TMainForm(TComponent* Owner);
 __fastcall ~TMainForm();
 float __fastcall MaxX();
 float __fastcall MinX();
 float __fastcall MaxY();
 float __fastcall MinY();
 void __fastcall Drawing(TCanvas* Canvas,bool smooth=true,
   int cw=0,int ch=0);
 const int Size;     //počet prvků pole Data
};
//---------------------------------
extern TMainForm *MainForm;
//---------------------------------
#endif
```

MAPMAIN.CPP:

```
//---------------------------------
#include <vcl\vcl.h>
#pragma hdrstop
#include "MapMain.h"
#include <float.h>
//---------------------------------
```

```
#pragma resource "*.dfm"
TMainForm *MainForm;
//---------------------------------
__fastcall TMainForm::TMainForm(TComponent* Owner)
 : TForm(Owner),Size(40)
{
 Color=clWhite;
 Mapper=NULL;
 Data=new float[2*Size];  //alokace pole
 //dočasné pole pro snadnou inicializaci a naplnění Data:
 //sudé prvky jsou x-ové souřadnice bodů,
 //liché prvky jsou y-ové souřadnice bodů:
 /*data představují údaje získané měřením V-A charakteristiky
 diody KZ260/5V6:*/
 float TempData[]={
   -5.61,-17.29, -5.62,-17.29, -5.62,-17.24, -5.62,-17.19,
   -5.62,-17.19, -5.62,-16.99, -5.62,-14.84, -5.60,-10.30,
   -5.56,-5.76, -5.26,-1.86, -5.11,-0.34, -4.64,-0.10,
   -4.17,0.00, -3.67,0.00, -3.18,0.00, -2.69,0.00,
   -2.19,0.00, -1.71,0.00, -1.20,0.00, -0.72,0.00,
   -0.21,0.00, 0.27,0.00, 0.66,0.98, 0.76,4.98,
   0.79,9.57, 0.81,14.26, 0.83,17.68, 0.83,17.58,
   0.83,17.53, 0.83,17.38, 0.82,17.33, 0.82,17.29,
   0.83,17.19, 0.83,17.14, 0.83,17.09, 0.82,17.09,
   0.83,16.99, 0.82,16.99, 0.82,16.94, 0.82,16.94
 };
 /*
 klasické řešení:
 for(int i=0;i<2*Size;i++)
   Data[i]=TempData[i];
 */
 //vtipné použití funkce z kapitoly 7.4.5:
 CopyMemory(Data,TempData,2*Size*sizeof(float));
 MenuNastaveniGrafProlozeny->Checked=true;
}
//---------------------------------
__fastcall TMainForm::~TMainForm()
{
 if(Mapper){
   delete(Mapper);
   Mapper=NULL;
 }
 delete(Data);
}
//---------------------------------
float __fastcall TMainForm::MaxX()
{
```

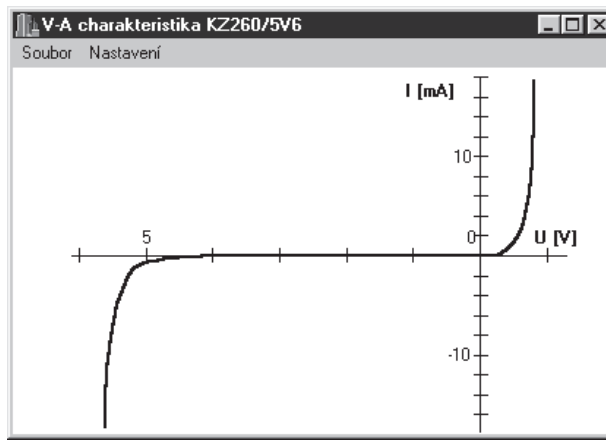
```
//určí maximální x-ovou souřadnici v Data:
 float MaxX=FLT_MIN;
 for(int i=0;i<Size;i++)
   MaxX=MaxX<Data[2*i]?Data[2*i]:MaxX;
 return(MaxX);
}
//---------------------------------
float __fastcall TMainForm::MinX()
{
//určí minimální x-ovou souřadnici v Data:
 float MinX=FLT_MAX;
 for(int i=0;i<Size;i++)
   MinX=MinX>Data[2*i]?Data[2*i]:MinX;
 return(MinX);
}
//---------------------------------
float __fastcall TMainForm::MaxY()
{
//určí maximální y-ovou souřadnici v Data:
 float MaxY=FLT_MIN;
 for(int i=0;i<Size;i++)
   MaxY=MaxY<Data[2*i+1]?Data[2*i+1]:MaxY;
 return(MaxY);
}
//---------------------------------
float __fastcall TMainForm::MinY()
{
//určí minimální y-ovou souřadnici v Data:
 float MinY=FLT_MAX;
 for(int i=0;i<Size;i++)
   MinY=MinY>Data[2*i+1]?Data[2*i+1]:MinY;
 return(MinY);
}
//---------------------------------
void __fastcall TMainForm::Drawing(TCanvas* Canvas,
 bool smooth,int cw,int ch)
{
/*funkce, která zajistí kreslení do kanvasu určeného Canvas.
 smooth určuje, zda se kreslení provádí pomocí Bézierových
 křivek nebo spojováním bodů
 cw a ch jsou souřadnice client area v jednotkách zařízení*/

 //zruš předchozí Mapper a vytvoř nový:
 if(Mapper){
   delete(Mapper);
   Mapper=NULL;
 }
```

```
//Mapper vytvoř podle prvků pole Data:
Mapper=new TMapper(Canvas,MinX()-1.5,MaxX()+1.5,
  MinY()-1.5,MaxY()+1.5,
  100,100,cw,ch);
Canvas->Pen->Color=clBlack;
Mapper->SetPenWidth(1);
//kreslení os:
Mapper->MoveTo(MinX()-0.5,0);
Mapper->LineTo(MaxX()+0.5,0);
Mapper->MoveTo(0,MinY()-0.5);
Mapper->LineTo(0,MaxY()+0.5);
//kreslení dílků os:
for(int x=MinX()-0.5;x<=MaxX()+0.5;x++){
  Mapper->MoveTo(x,0);
  Mapper->DrawNormLine(0,5);
  Mapper->DrawNormLine(0,-10);
}
for(int y=0;y<=MaxY()+0.5;y+=2){
  Mapper->MoveTo(0,y);
  Mapper->DrawNormLine(5,0);
  Mapper->DrawNormLine(-10,0);
}
for(int y=0;y>=MinY()-0.5;y-=2){
  Mapper->MoveTo(0,y);
  Mapper->DrawNormLine(5,0);
  Mapper->DrawNormLine(-10,0);
}
//kreslení popisů os:
Canvas->Font->Style=Canvas->Font->Style<<fsBold;
Mapper->TextOut(0,MaxY(),"I [mA]",
  -1.5*Canvas->TextWidth("I [mA]"),0);
Mapper->TextOut(MaxX(),0,"U [V]",
  0,1.5*Canvas->TextHeight("U [V]"));
/*generování popisů os-raději nuťte uživatele, aby
specifikoval sám*/
Canvas->Font->Style=Canvas->Font->Style>>fsBold;
Mapper->TextOut(-5,0,"5",
  -0.5*Canvas->TextWidth("5"),1.5*Canvas->TextHeight("5"));
Mapper->TextOut(0,10,"10",
  -1.5*Canvas->TextWidth("10"),Canvas->TextHeight("10")/2);
Mapper->TextOut(0,-10,"-10",
  -1.5*Canvas->TextWidth("-10"),Canvas->TextHeight("-10")/2);
Mapper->TextOut(0,0,"0",
  -1.5*Canvas->TextWidth("0"),1.5*Canvas->TextHeight("0"));
Mapper->SetPenWidth(2);  //šířka pera 2 pixely
Canvas->Pen->Color=ColorDialog->Color;
if(!smooth){
```

```
   //spojnicový graf:
   Mapper->MoveTo(Data[0],Data[1]);
   for(int i=1;i<Size;i++)
     Mapper->LineTo(Data[2*i],Data[2*i+1]);
 }
 else{
   //hladký graf:
   POINT *Points=new POINT[Size];    //pole bodů v log. souř.
   for(int i=0;i<Size;i++){
     Points[i].x=Mapper->TransfX(Data[2*i]);
     Points[i].y=Mapper->TransfY(Data[2*i+1]);
 }
 //kreslení
 PolyBezier(Canvas->Handle,Points,Size);
 delete(Points);          //pole už nepotřebujeme
 }
}
//---------------------------------
void __fastcall TMainForm::FormPaint(TObject *Sender)
{
//obnovování obrazu:
 Drawing(Canvas,MenuNastaveniGrafProlozeny->Checked,
   ClientWidth,ClientHeight);
}
//---------------------------------
void __fastcall TMainForm::FormResize(TObject *Sender)
{
//změna velikosti=>překresli vše
 Invalidate();
}
//---------------------------------
void __fastcall TMainForm::MenuNastaveniGrafProlozenyClick(
 TObject *Sender)
{
//proložený graf:
 MenuNastaveniGrafProlozeny->Checked=true;
 Invalidate();
}
//---------------------------------
void __fastcall TMainForm::MenuNastaveniGrafSpojnicovyClick(
 TObject *Sender)
{
//spojnicový graf:
 MenuNastaveniGrafSpojnicovy->Checked=true;
 Invalidate();
}
//---------------------------------
```

```
void __fastcall TMainForm::MenuSouborUlozitClick(TObject
 *Sender)
{
//uložení bitmapy prac. plochy pomocí GetFormImage:
 if(SaveDialog->Execute()){
   //metoda GetFormImage vrací bitmapu, kterou sama alokuje:
   Graphics:: TBitmap *b=GetFormImage();
   b->SaveToFile(SaveDialog->FileName);
   delete(b);        //bitmapu musíme zrušit my!
 }
}
//---------------------------------
void __fastcall TMainForm::MenuSouborKonecClick(TObject
 *Sender)
{
 Close();
}
//---------------------------------
void __fastcall TMainForm::MenuNastaveniBarvaClick(TObject
 *Sender)
{
//volba barvy křivky:
 TColor OldColor=ColorDialog->Color;
 if(!ColorDialog->Execute())
   ColorDialog->Color=OldColor;
 else
   Invalidate();
}
```



Obr. 9.1 Aplikace po spuštění (spojnicový graf)