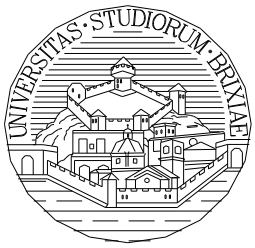
Università degli Studi di Brescia
Facoltà di Ingegneria
Dipartimento di Elettronica per l'Automazione



Elaborazione di traiettorie per robot non olonomi

Elaborato del corso di Robotica

Docente: prof. Riccardo Cassinis

Marco Baiguera mat.27353

Anno Accademico 1999/2000

Riassunto

Questo elaborato esamina il problema di portare un robot da un punto orientato del piano (cioè da un punto in cui è fissata la direzione del robot) ad un altro punto orientato, percorrendo una traiettoria con raggio di curvatura minimo non inferiore ad un valore assegnato (raggio di curvatura minimo di un robot non olonomo).
La soluzione proposta impiega curve cubiche di Bezier per disegnare una traiettoria che conduca il robot dal vettore di partenza (v0) al vettore di arrivo (v1)
Quando l'utilizzo di una sola curva di Bezier si è dimostrato inefficace od inefficiente sono state utilizzate due curve di Bezier contigue percorse in direzione opposta (manovra di parcheggio).
Un algoritmo iterativo esplora le curve di Bezier che risolvono il problema dello spostamento selezionando quelle che soddisfano il criterio di curvatura.
Utilizzando l'ambiente Borland Delphi è stata sviluppata un'interfaccia che permette di sperimentare la ricerca di una traiettoria a singola curva o di parcheggio e di archiviare in un file i punti costituenti la traiettoria ottenuta.

Presentazione del problema

Il problema consiste nel condurre un robot autonomo non olonomo dal punto di partenza a quello di arrivo rispettando due condizioni: che sia fissata la direzione di partenza e di arrivo del robot e che la traiettoria non compia curve di raggio inferiore ad un raggio di curvatura dato.
Lo spazio delle soluzioni è intuitivamente illimitato, lo scopo di questo elaborato è introdurre vincoli (oltre al raggio minimo di curvatura) per isolare una soluzione efficiente del problema cioè una traiettoria di lunghezza dello stesso ordine di grandezza della distanza da percorrere e ricavabile in un tempo pari ad alcuni secondi.
Il primo vincolo introdotto consiste nell'utilizzo di curve cubiche di Bezier.
Per minimizzare la lunghezza del percorso è stata utilizzata ove necessario una traiettorie di parcheggio cioè una suddivisione dell'intero percorso in due tratti (due curve di Bezier) di cui il primo percorso con velocità positiva (rispetto alla direzione di partenza del robot) ed il secondo con velocità negativa (in retromarcia).
Per limitare il tempo di elaborazione si è reso indispensabile bilanciare l'accuratezza della ricerca e la risoluzione della traiettoria trovata (circa 100 punti).

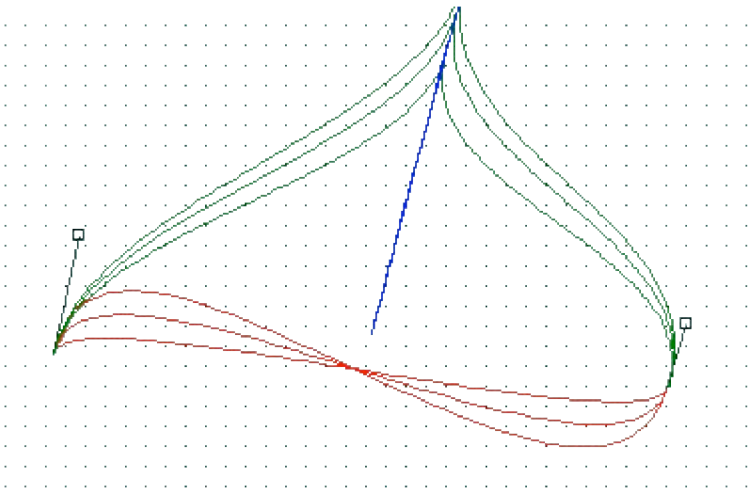


Figura 0 Le curve in rosso sono traiettorie a singola curva con curvatura minima 20,30,40; le curve in verde sono traiettorie di parcheggio con curvatura minima 40,50,60: per questi ultimi valori non esiste una traiettoria a curva singola di lunghezza ragionevole.

Introduzione alle curve di Bezier

Le curve cubiche di Bezier sono definite da quattro punti (in ordine di percorso p0, p1, p2, p3) e godono di due importanti proprietà: congiungono il punto p0 al punto p3 con una curva tangente in p0 al segmento p0-p1 e in p3 al segmento p2-p3 inoltre sono completamente contenute nell'inviluppo convesso dei quattro punti di definizione. La prima proprietà consente di collegare semplicemente i punti orientati di partenza e di arrivo (eventualmente con una sola Bezier), la seconda proprietà offre un rudimentale controllo sulla lunghezza della curva, semplicemente limitando la distanza dei punti di controllo (p1 e p2) dai punti di partenza e arrivo (p0, p3).

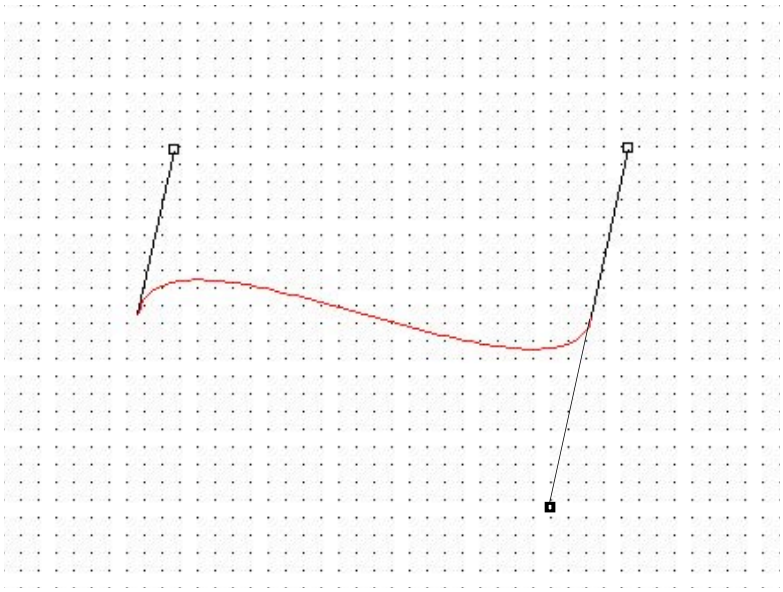


Figura 1 La curva in rosso è una curva di Bezier descritta dai punti p0,p1,p2,p3. Il punto p4 è simmetrico di p2 rispetto a p3

Esistono infinite soluzioni equivalenti al problema di traslazione del robot ottenute fissando i punti p0 e p3 e spostando i punti p1 e p2 lungo due rette che rispettino la direzione fissata dai vettori p0-p1 e p2-p3. Il punto p4 è simmetrico di p2 rispetto a p3 e rappresenta la direzione assunta nel punto p3 da un robot che percorra la curva partendo dal punto p0 con direzione p1. Risulta più naturale descrivere il problema in termini di vettore di partenza (p0-p1) e vettore di arrivo (p3-p4) piuttosto che ricorrere al punto p2 che caratterizza la curva di Bezier impiegata.
Il raggio di curvatura medio di una curva di Bezier (così come la lunghezza) intuitivamente cresce con la distanza dei punti p1 e p2 dai punti p0 e p3.
L'equazione che esprime le ascisse e le ordinate di una curva di Bezier in funzione di un parametro (t) sono espresse in figura 3.

$$x(t) = (1-t)^3 x_{p0} + 3t(1-t)^2 x_{p1} + 3t^2(1-t)x_{p2} + t^3 x_{p3}$$

$$y(t) = (1-t)^3 y_{p0} + 3t(1-t)^2 y_{p1} + 3t^2(1-t)y_{p2} + t^3 y_{p3}$$

Figura 2 Equazioni parametriche di definizione di una curva cubica di Bezier

Curvatura puntuale di una curva piana

Si definisce curvatura puntuale di una curva piana la derivata della tangente alla curva rispetto alla posizione lungo la curva (lunghezza della curva) in un punto, espressa in figura 3 per una curva piana parametrica, ed è pari all'inverso del raggio di curvatura in quel punto.

$$\frac{d\Theta}{ds} = \frac{dt}{ds}\frac{d\left(\tan^{-1}\left(\frac{y_t}{x_t}\right)\right)}{dt} = \frac{dt}{ds}\left(\frac{\frac{x_t \cdot y_{tt} - y_t \cdot x_{tt}}{x_t^{\,2}}}{1+\left(\frac{y_t}{x_t}\right)^2}\right) = \frac{x_t \cdot y_{tt} - y_t \cdot x_{tt}}{\left(x_t^{\,2} + y_t^{\,2}\right)^{\frac{3}{2}}}$$

$$x_t = \frac{dx(t)}{dt} \qquad x_{tt} = \frac{d^2x(t)}{dt^2}$$

$$y_t = \frac{dy(t)}{dt} \qquad y_{tt} = \frac{d^2y(t)}{dt^2}$$

Figura 3 Espressione della curvatura puntuale per una curva piana parametrica

A causa della complessità della formula di curvatura applicata alle curve di Bezier (quarto grado in t) è impossibile risolvere l'equazione differenziale per ottenere in forma chiusa l'espressione dei minimi del raggio di curvatura. Per ogni istanza del problema è necessario analizzare con un algoritmo iterativo le curve di Bezier candidate alla soluzione, calcolandone in tutti i punti il raggio di curvatura e confrontandolo con il raggio minimo dato.

La manovra di parcheggio

Quando i vettori di partenza e di arrivo formano un angolo ottuso (e per raggi di curvatura paragonabili alla distanza da percorrere) una sola curva di Bezier non è sufficiente a coprire l'intera rotazione necessaria perché il robot si orienti come il vettore destinazione oppure la traiettoria ottenuta è molto lunga, percorrendo uno o più giri intorno al punto di partenza. In questi casi si ottiene una traiettoria più efficiente (più corta) utilizzando due curve di Bezier consecutive, di cui una percorsa con velocità positiva ed una con velocità negativa (rispetto alla direzione del robot), cioè una traiettoria di parcheggio. Anche in questo caso le soluzioni possibili sono infinite, di conseguenza è stato introdotto un ulteriore vincolo sulla posizione del punto di parcheggio p5: appartenente alla retta parallela alla bisettrice dei due vettori e passante per il punto medio tra p0 e p3 (retta di parcheggio).
Il problema di principale di traslazione è così decomposto nel problema della ricerca di due curve di Bezier in cui uno dei vettori è il vettore di partenza o di arrivo e il secondo vettore giace sulla retta di parcheggio. Queste due curve sono in genere di facile individuazione perché l'angolo tra i vettori di partenza e di arrivo e il vettore di parcheggio è sempre acuto.
In alcuni casi il punto di parcheggio potrebbe essere scelto in uno qualsiasi dei semipiani individuati dalla retta p0-p3, per semplicità è sempre cercata la soluzione in cui la prima manovra è a velocità positiva, cioè nella direzione del vettore p0-p1.

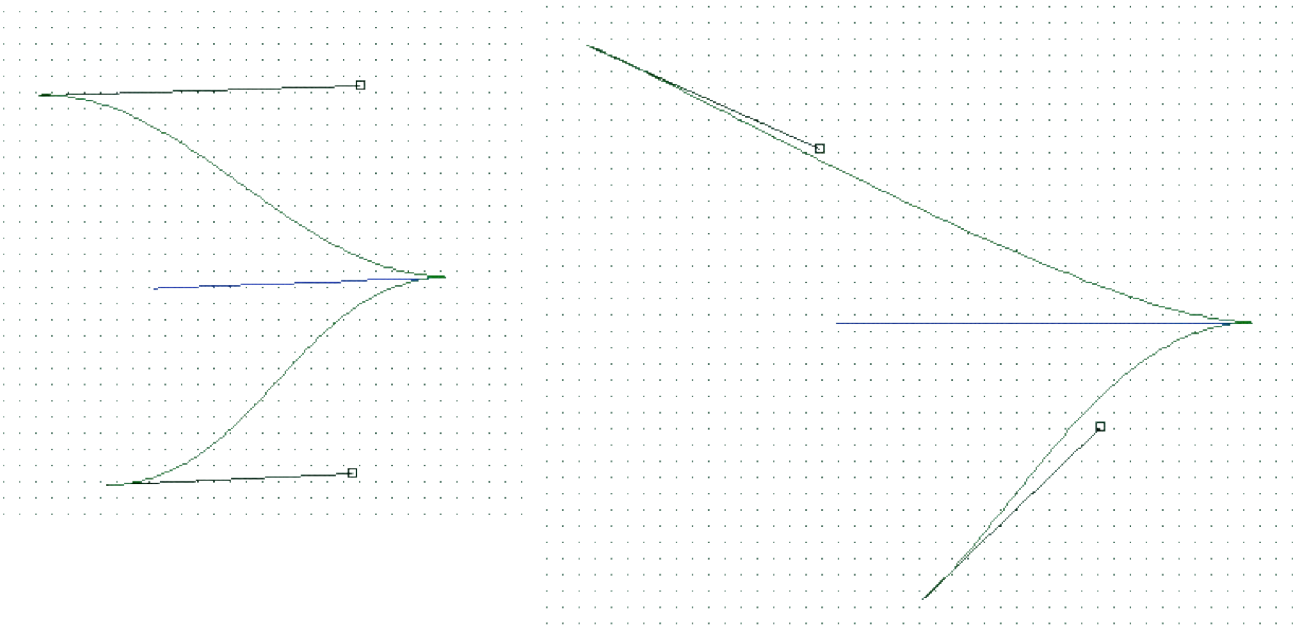


Figura 4 Alcuni esempi di traiettorie di parcheggio

L’algoritmo risolutivo in breve

Il cuore dell’algoritmo risolutivo è costituito da una procedura di ricerca di una singola curva di Bezier che riceve in ingresso quattro punti (in ordine di percorso p0, p1, p2, p3) traslati nel sistema di riferimento con origine in p0 oltre al raggio di curvatura minimo.
La procedura CercaCurvaBezier calcola iterativamente il raggio di curvatura minimo delle curve di Bezier ottenute ponendo uguale a k il modulo dei vettori p0-p1 e p2-p3 ed incrementando k a partire da un valore base fino ad ottenere una soluzione accettabile.
La ricerca si interrompe se k supera di 20 volte il valore del raggio di curvatura minimo.
Questo limite è necessario per evitare cicli infiniti quando non esiste alcuna soluzione ma talvolta potrebbe interrompere la ricerca anche quando la soluzione esiste.
La ricerca di una traiettoria di parcheggio è basata sull’allontanamento del punto di parcheggio dal punto medio di p0-p3 finché la ricerca di una curva di Bezier non è positiva per entrambe le curve p0-parcheggio, parcheggio-p3.

Strumenti utilizzati per l’implementazione dell’algoritmo

Per la semplificazione delle formule algebriche relative al calcolo della curvatura puntuale è stato usato il software Derive 5 per Windows che offre la possibilità di esportare le formule in linguaggio C e Pascal.
Per l’implementazione del software è stato utilizzato l’ambiente Borland Delphi 4 per Windows, un sistema di sviluppo rad (rapid application development) programmato in Object Pascal.

Il software realizzato

Il software realizzato consente di sperimentare gli algoritmi di cui si è discusso. L’interfaccia è costituita principalmente da una griglia in cui si specificano per punti (con il mouse in sequenza i punti p0, p1, p3, p4) i vettori di partenza e di arrivo, dopodiché la soluzione trovata viene disegnata (in rosso se soluzione a curva unica, in giallo se traiettoria di parcheggio, in blu il punto di parcheggio, figura 5).
Nella parte superiore della finestra:
a sinistra:

- le coordinate attuali del mouse nel sistema di riferimento dello schermo
- il pulsante CLS: reinizializza lo schermo
- il pulsante REDRAW: ridisegna l’ultima curva calcolata
- il pulsante INPUT: permette di inserire da tastiera le coordinate dei quattro punti che definiscono i vettori

al centro:

- un controllo con cui è possibile impostare il raggio di curvatura
- le coordinate degli ultimi quattro punti selezionati con il mouse

a destra:

- il valore del minimo raggio di curvatura della soluzione visualizzata (in verde a destra) oppure la scritta “Nessuna soluzione trovata” (in rosso).

I menù permettono di:

- esportare la traiettoria in un file ascii
- scegliere la ricerca di una traiettoria a curva unica o di una traiettoria di parcheggio o lasciare al sistema la scelta (impostazione Auto).

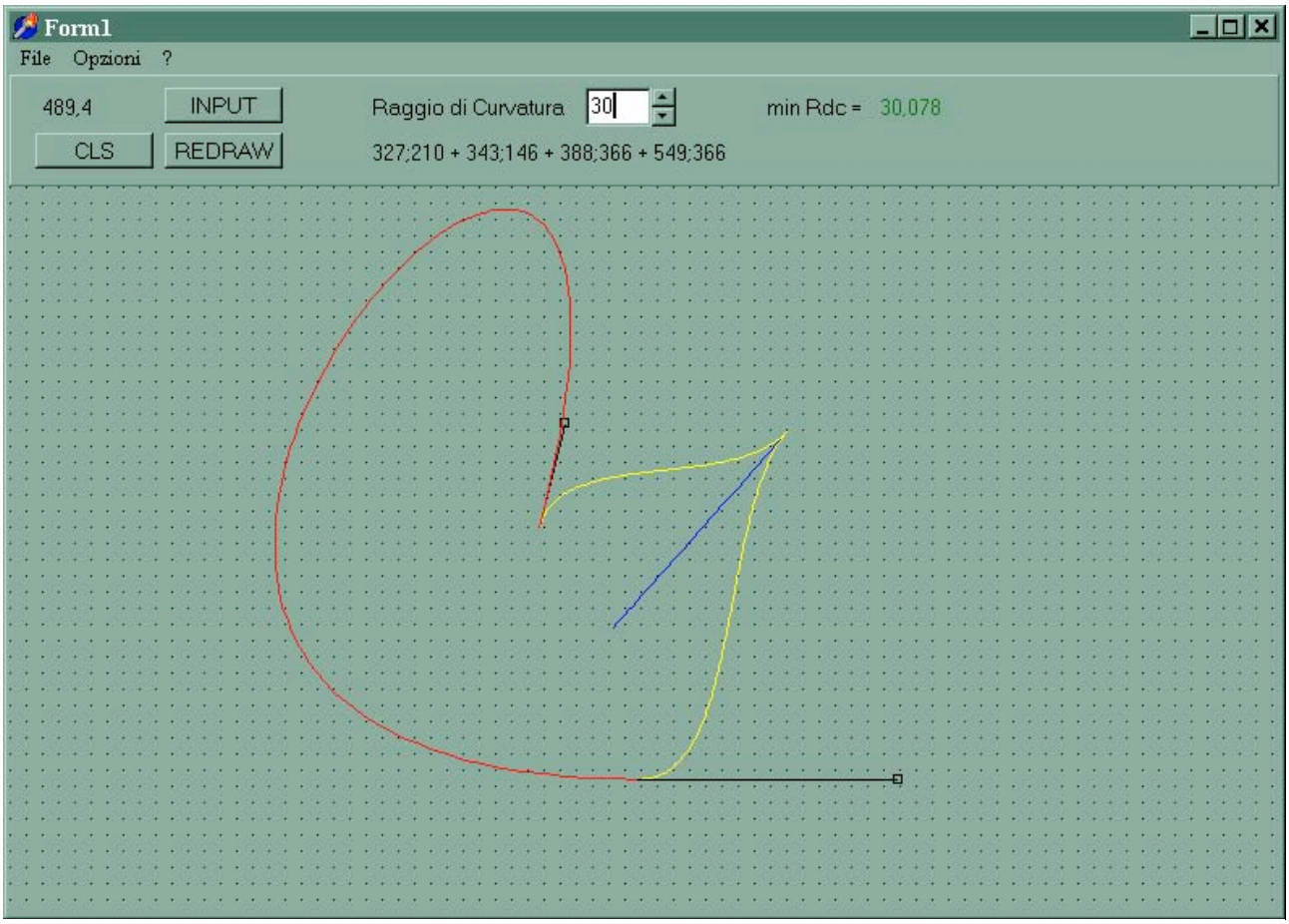


Figura 5 Uno screenshot del software realizzato

```
#V0(x0,y0;x1,y1):      0,000      0,000 ;   144,000    -64,000
#V1(x2,y2;x3,y3):    207,000   -341,000 ;   317,000   -235,000

Minimo Rdc=100

tinc=    0,010   Npassi=202

Traiettoria [x] [y]:
#     1      0,000       0,000
#     2      2,635      -1,180
#     3      5,353      -2,415
#     4      8,153      -3,703
.
.
#   199    212,583    -335,466
#   200    210,727    -337,339
#   201    208,866    -339,184
#   202    207,000    -341,000
```

Figura 6 Un esempio del file ASCII di esportazione delle traiettorie elaborate

Conclusioni e sviluppi futuri

Durante lo svolgimento di questo elaborato è emersa la difficoltà di trattare analiticamente il problema, a causa dell'elevato numero di variabili in gioco. Nonostante ciò l'algoritmo a "forza bruta" di ricerca della curva di Bezier e del punto di parcheggio si rivela quasi sempre molto efficiente.
Nei casi in cui è scelta in modo automatico dal software la traiettoria di parcheggio è più efficiente rispetto alla corrispondente traiettoria a curva singola.
La posizione del punto di parcheggio utilizzata dall'algoritmo proposto è fissata per tutte le configurazioni dei vettori di partenza e di arrivo anche se è ipotizzabile che per alcune particolari configurazioni sia più efficiente porre il punto di parcheggio più vicino all'uno o all'altro dei vettori.
Il maggior limite dell'algoritmo è la scarsa precisione della condizione di interruzione della ricerca (la distanza del punto p1 dal punto p0 cioè la lunghezza del vettore di partenza oppure la distanza del punto di parcheggio dal punto medio di p0-p3), cosicché la ricerca è interrotta senza valutare se una soluzione esiste oppure no ma soltanto ipotizzando che la traiettoria sia troppo lunga e quindi inutile.

Appendice A: Sorgenti del programma

Bezier.Dpr:

```
program Bezier1;

uses
  Forms,
  unit1 in 'unit1.pas' {Form1},
  Unit2 in 'Unit2.pas' {Form2};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm2, Form2);
  Application.Run;
end.
```

Unit1.Pas:

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus, ExtCtrls, Buttons, StdCtrls,math, ComCtrls;

type TFpoint = record
     X: Single;
     Y: Single;
     end;

type T4punti = array[0..3] of TFpoint;
type Tpunti = array of TFpoint;

type
  TForm1 = class(TForm)
    MainMenu1: TMainMenu;
    Panel1: TPanel;
    SpeedButton1: TSpeedButton;
    LabelC: TLabel;
    LabelP: TLabel;
    SpeedButton2: TSpeedButton;
    LabelRdc: TLabel;
    Label2: TLabel;
    File1: TMenuItem;
    Salvatraiettoria1: TMenuItem;
    EditRdc: TEdit;
    Label1: TLabel;
    N1: TMenuItem;
    Opzioni: TMenuItem;
    DuePunti: TMenuItem;
    Parc: TMenuItem;
    SpeedButton3: TSpeedButton;
    UpDown1: TUpDown;
    Auto1: TMenuItem;
    PaintBox1: TPaintBox;
    Label3: TLabel;
    Label4: TLabel;
```

```
    Label5: TLabel;
    Label6: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure PaintBox1MouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure PaintBox1Paint(Sender: TObject);
    procedure PaintBox1MouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure SpeedButton1Click(Sender: TObject);
    procedure SpeedButton2Click(Sender: TObject);
    procedure Duepunti1Click(Sender: TObject);
    procedure DuePuntiClick(Sender: TObject);
    procedure ParcClick(Sender: TObject);
    procedure EditRdcChange(Sender: TObject);
    procedure SpeedButton3Click(Sender: TObject);
    procedure Salvatraiettoria1Click(Sender: TObject);
    procedure N1Click(Sender: TObject);
    procedure Auto1Click(Sender: TObject);
    procedure PaintBox1Click(Sender: TObject);
  private
    { Private declarations }
    contpunti : integer;
    Bezier : Tpunti;
    tracciavettore : boolean;
    OrigineVettore : Tpoint;
    PuntiBezier : Tpunti;
    IntervalloGriglia : integer;
  public
    { Public declarations }

    Vettori : array [0..4] of Tpoint;
    raggio : single;

    procedure DisegnaGriglia;
    procedure DisegnaVettori(Nvettore : integer = 0);
    procedure DisegnaPunti;
    function Schermo2Robot(PuntiS : array of Tpoint) : T4punti;
    function Robot2Schermo(Unpunto : TFpoint;Origine : Tpoint) : Tpoint;
    function CercaCurvaBezier(var QuattroPuntiRobot : array of TFpoint;
                              rdcdato : single;maxk : single =0) : single;
    Procedure CalcolaCurvaBezier(QuattroPuntiRobot : array of TFpoint;var
Punti:Tpunti);
    procedure DisegnaCurva(Origine : Tpoint; Acanvas : TCanvas; Punti :
Tpunti;color : Tcolor = clhighlight);
    function Intersezione(QuattroPuntiRobot : array of TFpoint) : TFpoint;
    procedure ElaboraParcheggio(QuattroPuntischermo : array of Tpoint);
    procedure ProcessaVettori;
    function Dist(puntoA : TFpoint;puntoB : TFpoint) : single;
    function atan(x0,y0,x1,y1 : single) : single;
    function convergenti(PuntiR : T4punti) : boolean;
end;

var
  Form1: TForm1;

const
  tinc : single = 0.01;

implementation

uses Unit2;
```

```
{$R *.DFM}

function Tform1.atan(x0,y0,x1,y1 : single) : single; {x1 -> x0}
var beta : single;
begin
  if x0=x1 then
     if y0>y1 then beta := pi/2
     else beta := -pi/2
  else
     if x0>x1 then
        beta := ARCTAN((y0-y1)/(x0-x1))
     else beta := ARCTAN((y0-y1)/(x0-x1)) + pi;
  result := beta;
end;


procedure TForm1.FormCreate(Sender: TObject);
begin
  contpunti := -1;
  IntervalloGriglia := 10;
  raggio := StrToFloat(EditRdc.text);
end;

procedure TForm1.PaintBox1MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var
  UnPunto : Tpoint;
  UnFPunto : TFpoint;
  PuntiR : T4punti;
begin
  begin
  if ContPunti = -1 then COntpunti :=0;

  unpunto.x := X;
  unpunto.y := Y;

  Vettori[contpunti] := unpunto;

  begin
     contpunti := (contpunti +1) mod 4;

     with paintBox1.Canvas do
     begin
       Pixels[X,Y] := clwhite;

       Pen.color := clred;
       Brush.color := clred;
       if (contpunti=2) then
          begin
             PaintBox1Paint(Sender);
             DisegnaVettori(1);
          end;
       if (contpunti=0) then
          begin
             DisegnaVettori(2);
          end;
     end;

     LabelP.Caption := Format('(%d;%d) + (%d;%d) + (%d;%d) +
(%d;%d)',[Vettori[0].x,Vettori[0].y,
                    Vettori[1].x,Vettori[1].y,Vettori[2].x,Vettori[2].y,
                    Vettori[3].x,Vettori[3].y]);
```

```
     With Form2 do
          begin
          EditX0.text := IntToStr(Vettori[0].x);
          EditX1.text := IntToStr(Vettori[1].x);
          EditX2.text := IntToStr(Vettori[2].x);
          EditX3.text := IntToStr(Vettori[3].x);
          EditY0.text := IntToStr(Vettori[0].Y);
          EditY1.text := IntToStr(Vettori[1].Y);
          EditY2.text := IntToStr(Vettori[2].Y);
          EditY3.text := IntToStr(Vettori[3].Y);
          end;

     if (contpunti=0) then
        ProcessaVettori;
  end
end;
end;

procedure TForm1.PaintBox1Paint(Sender: TObject);
begin
  DisegnaGriglia;
  if ContPunti <> -1 then
     if Contpunti = 0 then DisegnaVettori
        else if Contpunti = 2 then DisegnaVettori(1);
end;

procedure TForm1.PaintBox1MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
  LabelC.Caption := IntToStr(X) + ',' + IntToStr(Y);
{  if tracciaVettore then
     begin
     with PaintBox1.Canvas do
          begin
          Pen.color := color;
          Lineto(OrigineVettore.X,OrigineVettore.Y);
          Pen.color := clred;
          LineTo(x,y);
          end;
     end;}
end;

procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
  DisegnaGriglia;
end;

procedure TForm1.SpeedButton2Click(Sender: TObject);
begin
  Form2.Show;
end;

procedure TForm1.Duepunti1Click(Sender: TObject);
begin
  Duepunti.checked := true;
end;

function TForm1.Schermo2Robot(PuntiS: array of Tpoint): T4punti;
var puntiR : T4punti;
begin
  puntiR[0].x:=0;
```

```
  puntiR[0].y:=0;

  puntiR[1].x:=PuntiS[1].x - PuntiS[0].x;
  puntiR[1].y:=PuntiS[0].y - PuntiS[1].y;

  puntiR[2].x:=PuntiS[2].x - PuntiS[0].x;
  puntiR[2].y:=PuntiS[0].y - PuntiS[2].y;

  puntiR[3].x:=PuntiS[3].x - PuntiS[0].x;
  puntiR[3].y:=PuntiS[0].y - PuntiS[3].y;

  result := puntiR;
end;

function TForm1.CercaCurvaBezier (var QuattroPuntiRobot: array of TFpoint;
                                  rdcdato: single;maxk : single = 0) : single;

{         In input i quattro punti caratteristici della bezier in ordine di
percorso     }

var
   k : single;
   rdc,minrdc : single;
   alpha,beta : single;
   d03,x11,y11,x22,y22,x33,y33 : single;
   t : single;
   num,n2,den : single;
   Pinter:TFpoint;
   rapporto : single;
   ca,sa,cb,sb : single;
   d1,d2 : single;

begin
  minrdc := 0;

  x11 := QuattroPuntiRobot[1].x;
  x22 := QuattroPuntiRobot[2].x;
  x33 := QuattroPuntiRobot[3].x;
  y11 := QuattroPuntiRobot[1].y;
  y22 := QuattroPuntiRobot[2].y;
  y33 := QuattroPuntiRobot[3].y;

  d03 := sqrt(sqr(x33)+sqr(y33));

  k := raggio/2;

  alpha := atan(x11,y11,0,0);
  beta := atan(x22,y22,x33,y33);

  ca := cos(alpha);
  sa := sin(alpha);
  cb := cos(beta);
  sb := sin(beta);

  if maxk = 0 then maxk := 20*raggio;

  while (minrdc < rdcdato) do

begin
  Application.ProcessMessages;

  rapporto := abs(minrdc - rdcdato)/rdcdato;
```

```
  if  rapporto > 0.5 then k:=k*1.1
  else
  begin
  if rapporto > 0.05 then k:=k*1.01
  else k:=k+1;
  end;

  x11 := K*ca;
  y11 := K*sa;

  x22 := x33 + k*cb;
  y22 := y33 + k*sb;

  t := 0;
  minrdc :=1000;

  while (((t <= 1) and (minrdc > rdcdato)) and (k <= maxk)) do

  begin
    n2 := abs(Power(t,4)*(9*SQR(x11)+6*x11*(x33-3*x22)+9*SQR(x22)-
       6*x22*x33+SQR(x33)+SQR(3*y11-3*y22+y33))-4*Power(t,3)*(6*SQR(x11)+
       x11*(2*x33-9*x22)+3*SQR(x22)-x22*x33+(2*y11-y22)*(3*y11-3*y22+y33))+
       2*SQR(t)*(11*SQR(x11)+x11*(x33-11*x22)+2*SQR(x22)+11*SQR(y11)+y11*
       (y33-11*y22)+2*SQR(y22))-4*t*(2*SQR(x11)-x11*x22+y11*(2*y11-
y22))+SQR(x11)+
       SQR(y11));

    num := (3*Power(n2,3/2));

    den :=(2*(SQR(t)*(x11*(3*y22-2*y33)+x22*(y33-3*y11)+x33*(2*y11-y22))+
       t*(y11*(3*x22-x33)-x11*(3*y22-y33))+x11*y22-x22*y11));

    rdc := abs(num/den);

    if rdc < minrdc then
       begin
       minrdc := rdc;
{       if LabelRdc.Font.Color = clgreen
       then LabelRdc.Font.Color := clred
       else LabelRdc.Font.Color := clgreen;}
       end;

    t := t + tinc;
  end;

end;

  if k <= maxk then
  begin
  QuattroPuntiRobot[1].x:=x11;
  QuattroPuntiRobot[2].x:=x22;
  QuattroPuntiRobot[3].x:=x33;
  QuattroPuntiRobot[1].y:=y11;
  QuattroPuntiRobot[2].y:=y22;
  QuattroPuntiRobot[3].y:=y33;
  result := minrdc;
  LabelRdc.Font.Color := clgreen;
  LabelRdc.Caption := Format('%8.3f',[minrdc]);
  end
  else
  begin
  LabelRdc.Font.Color := clred;
```

```
  LabelRdc.Caption := 'Nessuna soluzione trovata';
  result := -1;
  end;
end;

Procedure TForm1.CalcolaCurvaBezier(QuattroPuntiRobot : array of TFpoint;var
Punti:Tpunti);
var
  f : single;
begin
  f := 0;
  while f <= 1 do
    begin
    Punti[ceil(f/tinc)].x := (Power((1-f),3)*Quattropuntirobot[0].x +
                  3*f*sqr(1-f)*Quattropuntirobot[1].x +
                  3*sqr(f)*(1-f)*Quattropuntirobot[2].x +
                  power(f,3)*Quattropuntirobot[3].x );

    Punti[ceil(f/tinc)].y := (Power((1-f),3)*Quattropuntirobot[0].y +
                  3*f*sqr(1-f)*Quattropuntirobot[1].y +
                  3*sqr(f)*(1-f)*Quattropuntirobot[2].y +
                  power(f,3)*Quattropuntirobot[3].y );

    f := f + tinc;
    end;
end;

procedure TForm1.DisegnaCurva(Origine : Tpoint; Acanvas: TCanvas; Punti:
Tpunti;color : Tcolor);
var
   f : single;
   unpunto : Tpoint;

begin
  f := 0;
  with ACanvas do
  begin
  Pen.color := color;

  Unpunto.x := ceil(Punti[0].x) + Origine.x;
  Unpunto.y := ceil(Punti[0].y) + Origine.y;

  PenPos := Unpunto;

  while f <= 1 do
  begin
  Lineto ((ceil(Punti[ceil(f/tinc)].x) + Origine.x),(Origine.y -
ceil(Punti[ceil(f/tinc)].y)));
  f := f + tinc;
  end;
  end;
end;

function TForm1.Intersezione(QuattroPuntiRobot: array of TFpoint): TFpoint;
var
   unpunto : TFpoint;
   den : single;
begin
   den := (QuattroPuntiRobot[1].x*(QuattroPuntiRobot[2].y-
QuattroPuntiRobot[3].y)-
      QuattroPuntiRobot[1].y*(QuattroPuntiRobot[2].x-QuattroPuntiRobot[3].x));
   if den = 0 then
```

```
      begin
      unpunto.x := (QuattroPuntiRobot[1].x + QuattroPuntiRobot[3].x)/2;
      unpunto.y := (QuattroPuntiRobot[1].y* + QuattroPuntiRobot[2].y)/2;
      end
   else
   begin
   unpunto.x :=
QuattroPuntiRobot[1].x*(QuattroPuntiRobot[3].x*QuattroPuntiRobot[2].y-
               QuattroPuntiRobot[2].x*QuattroPuntiRobot[3].y)/den;
   unpunto.y :=
QuattroPuntiRobot[1].y*(QuattroPuntiRobot[3].x*QuattroPuntiRobot[2].y-
               QuattroPuntiRobot[2].x*QuattroPuntiRobot[3].y)/den;
   end;
   result := unpunto;
end;

function TForm1.Robot2Schermo(Unpunto: TFpoint; Origine: Tpoint): Tpoint;
var puntoS : Tpoint;
begin
  puntoS.x := ceil(Unpunto.x + Origine.x);
  puntoS.y := ceil(Origine.y - Unpunto.y );
  result := puntoS;
end;

function TForm1.dist(puntoA : TFpoint;puntoB : TFpoint) : single;
begin
  result := sqrt(sqr(puntoA.x - puntoB.x) + sqr(puntoA.y - puntoB.y));
end;

procedure TForm1.DisegnaGriglia;
var i,j : integer;
begin
  i := 0;
  j := 0;
  with PaintBox1.Canvas do
  begin
  Brush.Color := clbtnface;
  FillRect(ClientRect);
  end;

  while i <= PaintBox1.Height do
        begin
        j := 0;
        while j <= PaintBox1.Width do
              begin
              PaintBox1.Canvas.Pixels[j,i] := clBtnText;
              j := j + IntervalloGriglia;
              end;
        i := i + IntervalloGriglia;
        end;
end;

procedure TForm1.DisegnaPunti;
begin

end;

procedure TForm1.DisegnaVettori(Nvettore : integer =0);
var Unrect : Trect;
begin
  with PaintBox1.Canvas do
       begin
```

```
       if (Nvettore=1) or (Nvettore=0) then
       begin
       Pen.color := clBlack;
       Brush.color := clblack;
       penpos := Vettori[0];
       Lineto(Vettori[1].x,Vettori[1].y);
       unrect.Left := Vettori[1].X-3;
       unrect.Top := Vettori[1].Y-3;
       unrect.Right := Vettori[1].X+3;
       unrect.Bottom := Vettori[1].Y+3;
       FrameRect(unrect);
       end;
       if (Nvettore=2) or (Nvettore=0) then
       begin
       Pen.color := clBlack;
       Brush.color := clblack;
       penpos := Vettori[2];
       Lineto(Vettori[3].x,Vettori[3].y);
       unrect.Left := Vettori[3].X-3;
       unrect.Top := Vettori[3].Y-3;
       unrect.Right := Vettori[3].X+3;
       unrect.Bottom := Vettori[3].Y+3;
       FrameRect(unrect);
       end;
       end;
end;

procedure TForm1.DuePuntiClick(Sender: TObject);
begin
  DuePunti.checked := true;
end;

procedure TForm1.ParcClick(Sender: TObject);
begin
  Parc.checked := true;
end;

procedure TForm1.ProcessaVettori;
var
   PuntiR:T4punti;
   UnFpunto:TFpoint;
   rdc : single;
begin
  if (Parc.Checked) or (Auto1.checked and Convergenti(Schermo2Robot(Vettori)))
then

     begin {Elabora parcheggio}
     ElaboraParcheggio(Vettori);
     end   {Elabora parcheggio}

  else
        begin
        PuntiR := Schermo2Robot(Vettori);
        UnFPunto.x := 2*PuntiR[2].x-PuntiR[3].x;
        UnFPunto.y := 2*PuntiR[2].y-PuntiR[3].y;
        PuntiR[3] := PuntiR[2];
        PuntiR[2] := UnFPunto;

        rdc := CercaCurvaBezier(PuntiR,Raggio);
        if  rdc > 0 then
           begin
           Setlength(Bezier,ceil(1/tinc));
```

```
           CalcolaCurvaBezier(PuntiR,Bezier);
           DisegnaCurva(Vettori[0],PaintBox1.Canvas,Bezier,clred);
           end
        end;
end;

procedure TForm1.EditRdcChange(Sender: TObject);
begin
  Try
  Raggio := StrToFloat(EditRdc.Text);
  Except
  On EConvertError do Raggio := 50;
  End;
end;

procedure TForm1.SpeedButton3Click(Sender: TObject);
begin
  Form1.DisegnaVettori;
  Form1.ProcessaVettori;
end;

procedure TForm1.ElaboraParcheggio(QuattroPuntischermo : array of Tpoint);
var
   PuntiR,NuoviPunti,NuoviPuntiB : T4punti;
   UnFpunto,Pinter,Pmedio : TFpoint;
   alpha,beta,gamma,delta : single;
   h : single;
   unPuntoS : Tpoint;
   rdc1,rdc2 : single;
   Bezier1,Bezier2 : Tpunti;
   f : single;
   Origine2 : Tpoint;
   semipiano : boolean; {in quale semipiano è V1 rispetto alla congiungente
P0,P3}
   piumeno : integer;
   cg,sg : single;
begin
     PuntiR := Schermo2Robot(QuattroPuntischermo);

     UnFpunto := PuntiR[3];
     PuntiR[3] := PuntiR[2];
     PuntiR[2] := UnFPunto;

     PMedio.x := (PuntiR[0].x + PuntiR[3].x)/2;
     PMedio.y := (PuntiR[0].y + PuntiR[3].y)/2;
     Pinter := Intersezione(PuntiR);

     NuoviPunti[0] := PuntiR[0];
     NuoviPunti[1] := PuntiR[1];
     NuoviPunti[2] := PMedio;

     h:=raggio/2;
     rdc1 := -1;
     rdc2 := -1;

     alpha := (atan(PuntiR[1].x,PuntiR[1].y,0,0));
     beta := (atan(PuntiR[2].x,PuntiR[2].y,PuntiR[3].x,PuntiR[3].y));

     gamma := (alpha + beta)/2;

     if abs(beta - alpha) > pi then gamma := gamma + pi;
```

```
     cg := cos(gamma);
     sg := sin(gamma);


     Label3.Caption := Format('%6.2f',[alpha/pi*180]);
          Label4.Caption := Format('%6.2f',[beta/pi*180]);
          Label5.Caption := Format('%6.2f',[gamma/pi*180]);
          Label6.Caption := Format('%6.2f',[(alpha+beta)/pi*180]);

     NuoviPunti[3].x := Pmedio.x + h * cos(gamma);
     NuoviPunti[3].y := PMedio.y + h * sin(gamma);

     unPuntoS := Robot2Schermo(NuoviPunti[3],QuattroPuntiSchermo[0]);
     PaintBox1.Canvas.PenPos := unPuntoS;

     NuoviPuntiB[0]:=NuoviPunti[0];

     while (((rdc1<0) or (rdc2<0)) and (h<10*raggio)) do

     begin
     NuoviPunti[3].x := Pmedio.x + h * cg;
     NuoviPunti[3].y := PMedio.y + h * sg;

     unPuntoS := Robot2Schermo(NuoviPunti[3],QuattroPuntiSchermo[0]);
     PaintBox1.Canvas.Pen.Color :=clblue;
     PaintBox1.Canvas.LineTo(unPuntoS.x,unPuntoS.y);

     rdc1 := CercaCurvaBezier (NuoviPunti,raggio);
     if rdc1 > 0 then
     begin

     Origine2 := Robot2Schermo(Nuovipunti[3],QuattroPuntiSchermo[0]);

     NuoviPuntiB[1].x:=NuoviPunti[2].x - NuoviPunti[3].x;
     NuoviPuntiB[1].y:=NuoviPunti[2].y - NuoviPunti[3].y;

     NuoviPuntiB[2].x:=PuntiR[2].x - NuoviPunti[3].x;
     NuoviPuntiB[2].y:=PuntiR[2].y - NuoviPunti[3].y;

     NuoviPuntiB[3].x:=PuntiR[3].x - NuoviPunti[3].x;
     NuoviPuntiB[3].y:=PuntiR[3].y - NuoviPunti[3].y;

     rdc2 := CercaCurvaBezier (NuoviPuntiB,raggio);
     if (rdc2 > 0) then
        begin
        setlength(Bezier2,ceil(1/tinc));

        CalcolaCurvaBezier(NuoviPuntiB,Bezier2);

        setlength(Bezier1,ceil(1/tinc));

        CalcolaCurvaBezier(NuoviPunti,Bezier1);

        setlength(Bezier,2*ceil(1/tinc));

        f := 0;
        while f <= 1 do
              begin
              Bezier[ceil(f/tinc)] := Bezier1[ceil(f/tinc)];

              Bezier[ceil(1/tinc)+ceil(f/tinc)].X := Bezier1[ceil(1/tinc)-1].x +
Bezier2[ceil(f/tinc)].X;
```

```
              Bezier[ceil(1/tinc)+ceil(f/tinc)].Y := Bezier1[ceil(1/tinc)-1].y +
Bezier2[ceil(f/tinc)].Y;
              f := f+tinc;
              end;

        DisegnaCurva(Origine2,Form1.Paintbox1.Canvas,Bezier2,clgreen);


DisegnaCurva(QuattroPuntiSchermo[0],Form1.Paintbox1.Canvas,copy(Bezier,0,ceil(1/
tinc)),clgreen);

        if rdc1<=rdc2 then LabelRdc.Caption := Format('%8.3f',[rdc1])
           else LabelRdc.Caption := Format('%8.3f',[rdc2]);
        end;
     end;

     h := h * 1.1;
     end;

end;

procedure TForm1.Salvatraiettoria1Click(Sender: TObject);
var f : single;
    ExportFile : TextFile;
    PuntiR : T4punti;
    i : integer;
begin
  AssignFile(ExportFile,'Bezier.Dat');
  Rewrite(ExportFile);
  PuntiR := Schermo2Robot(Vettori);
  writeln(ExportFile,Format('#V0(x0,y0;x1,y1):  %8.3f  %8.3f ; %8.3f  %8.3f',

[PuntiR[0].x,PuntiR[0].y,PuntiR[1].x,PuntiR[1].y]));
  writeln(ExportFile,Format('#V1(x2,y2;x3,y3):  %8.3f  %8.3f ; %8.3f  %8.3f',

[PuntiR[2].x,PuntiR[2].y,PuntiR[3].x,PuntiR[3].y]));
  writeln(ExportFile);
  writeln(ExportFile,'Minimo Rdc=',FloatToStr(Raggio));
  writeln(ExportFile);
  write(ExportFile,'tinc=',Format('%8.3f  ',[tinc]));
  if Parc.Checked then
  writeln(ExportFile,'Npassi=',2*ceil(1/tinc))
  else
  writeln(ExportFile,'Npassi=',ceil(1/tinc));
  writeln(ExportFile);
  writeln(ExportFile,'Traiettoria [x] [y]:');
  i := 1;
  f := 0;
  while f <= 1 do
        begin
        Write(ExportFile,Format('#%5d  ',[i]));
        Writeln(ExportFile,Format('%8.3f  ',[Bezier[ceil(f/tinc)].X]),'
',Format('%8.3f  ',[Bezier[ceil(f/tinc)].Y]));
        f := f+tinc;
        inc(i);
        end;
  if Parc.Checked then
  begin
  f := 0;
  while f <= 1 do
        begin
        Write(ExportFile,Format('#%5d  ',[i]));
```

```
        Writeln(ExportFile,Format('%8.3f
',[Bezier[ceil(1/tinc)+ceil(f/tinc)].X]),' ',Format('%8.3f
',[Bezier[ceil(1/tinc)+ceil(f/tinc)].Y]));
        f := f+tinc;
        inc(i);
        end;

  end;

  System.Close(ExportFile);
end;

procedure TForm1.N1Click(Sender: TObject);
var s1,s2 : string;
begin
  s1 := 'CercaBezier realizzato da Marco Baiguera AA 1999/2000';
  s2 := 'About CercaBezier';
  Application.MessageBox(pchar(s1),pchar(s2),mb_OK);
end;

procedure TForm1.Auto1Click(Sender: TObject);
begin
  Auto1.checked := true;
end;

function TForm1.convergenti(PuntiR: T4punti): boolean;
var Pinter : TFpoint;
begin
  Pinter := intersezione(PuntiR);
  if ((dist(PuntiR[0],Pinter) < dist(PuntiR[1],Pinter)) and
     (dist(PuntiR[2],Pinter) < dist(PuntiR[3],Pinter))) or
     ((dist(PuntiR[0],Pinter) > dist(PuntiR[1],Pinter)) and
     (dist(PuntiR[2],Pinter) > dist(PuntiR[3],Pinter)))
  then
  result := true
  else
  result := false;
end;

procedure TForm1.PaintBox1Click(Sender: TObject);
begin
if Raggio > 999 then
     begin
     Raggio := 999;
     EditRdc.Text := FloatToStr(Raggio);
     end;
end;

end.
```

Unit2.pas:

```
unit Unit2;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm2 = class(TForm)
```

```
    EditX0: TEdit;
    EditY0: TEdit;
    EditX1: TEdit;
    EditY1: TEdit;
    EditX2: TEdit;
    EditY2: TEdit;
    EditX3: TEdit;
    EditY3: TEdit;
    Button1: TButton;
    Label1: TLabel;
    Label2: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

uses Unit1,math;

{$R *.DFM}

procedure TForm2.Button1Click(Sender: TObject);
var
   TantiPunti : Tpunti;
   PuntiR : T4punti;

begin
  Form1.Vettori[0].X := StrToInt(EditX0.text);
  Form1.Vettori[0].Y := StrToInt(EditY0.text);
  Form1.Vettori[1].X := StrToInt(EditX1.text);
  Form1.Vettori[1].Y := StrToInt(EditY1.text);
  Form1.Vettori[2].X := StrToInt(EditX2.text);
  Form1.Vettori[2].Y := StrToInt(EditY2.text);
  Form1.Vettori[3].X := StrToInt(EditX3.text);
  Form1.Vettori[3].Y := StrToInt(EditY3.text);
  Hide;
  Form1.DisegnaVettori;
  Form1.ProcessaVettori;

end;

end.
```

Appendice B: Bibliografia

Tutta la documentazione sulle curve piane in generale e sulle curve di Bezier in particolare è stata reperita tramite Internet, con il motore di ricerca www.google.com ricercando le parole chiave bezier,curve o curvature.

Indice

Riassunto 2

Presentazione del problema 3

Introduzione alle curve di Bezier 4

Curvatura puntuale di una curva piana 5

La manovra di parcheggio 6

L'algoritmo risolutivo in breve 7

Strumenti utilizzati per l'implementazione dell'algoritmo 7

Il software realizzato 7

Conclusioni e sviluppi futuri 9

Appendice A: sorgenti del programma 10

Appendice B: bibliografia 24

Indice 24