q8

Thursday, May 02, 2024 9:09 PM

8) Write a program to draw Hermite/Bezier curve.

```
// curve
// write a program to draw  Hermite/Bezier curve.


#include <iostream>
#include <cmath>
#include <vector>
#include <utility>
#include <math.h>


#include <graphics.h>


#define M_PI 3.14159265358979323846


using namespace std;


double find_x(double u, vector<pair<double, double>> control_points)
{
    return (control_points[0].first * pow((1 - u), 3)) + (control_points[1].first *
3 * u * pow(1 - u, 2)) +
          (control_points[2].first * 3 * pow(u, 2) * (1 - u)) +
(control_points[3].first * 1 * pow(u, 3));
}


double find_y(double u, vector<pair<double, double>> control_points)
{
    return (control_points[0].second * pow((1 - u), 3)) +
(control_points[1].second * 3 * u * pow(1 - u, 2)) +
(control_points[2].second * 3 * pow(u, 2) * (1 - u)) +
(control_points[3].second * 1 * pow(u, 3));
}


int main()
{
    vector<pair<double, double>> control_points;
    control_points.push_back(make_pair(100, 100));
    control_points.push_back(make_pair(200, 30));
    control_points.push_back(make_pair(150, 90));
    control_points.push_back(make_pair(400, 300));


    int gd = DETECT, gm;
    char pathtodriver[] = "";
    initgraph(&gd, &gm, pathtodriver);


    double t = 0.0;
    while (t <= 1.0)
    {


        putpixel(find_x(t, control_points), find_y(t, control_points), RED);
        t += 0.001;
    }


    getch();
    closegraph();
}

//=================
```

```
// write a program to draw  Hermite/Bezier curve.
#include <iostream>
#include <cmath>
#include <vector>
#include <utility>
#include "C:\Users\krishna\Desktop\cpp\dda.cpp"


#include <graphics.h>


#define M_PI 3.14159265358979323846


using namespace std;


// p1  ,  p4 ,   R1 ,  R4


double find_x(double t, vector<pair<double, double > > control_points){
    double t3 = pow(t,3);
    double t2 = pow(t,2);



    return  control_points[0].first*( 2*t3 - 3*t2 + 1) + control_points[1].first *
(-2*t3  + 3*t2) +
            control_points[2].first*( t3 -2*t2 + t ) + control_points[3].first*( t3
- t2) ;
}


double find_y(double t, vector<pair<double, double > > control_points){
       double t3 = pow(t,3);
       double t2 = pow(t,2);
    return  control_points[0].second*( 2*t3 - 3*t2 + 1) +
control_points[1].second *(-2*t3  + 3*t2) +
           control_points[2].second*( t3 -2*t2 + t ) +
control_points[3].second *( t3 - t2) ;
}




int main(){
    vector<pair<double, double > > control_points;  // P1 , P4     ,,  R1 ,
R2
    control_points.push_back(make_pair(0,0));
    control_points.push_back(make_pair(400,400));
    control_points.push_back(make_pair(90,400));
    control_points.push_back(make_pair(0,400));



     int gd = DETECT, gm;
    char pathtodriver[] = "";
    initgraph(&gd, &gm, pathtodriver);


double t =0.0;
    while(t <=1.0){
     //  cout<<find_x(t,control_points)<<" , "<<find_y(t,control_points)
<<endl;
     if(t<=0.5){
          putpixel(find_x(t,control_points), find_y(t,control_points), RED);
     }else{
          putpixel(find_x(t,control_points), find_y(t,control_points), BLUE);
     }

       t += 0.00001;
   }


getch();
closegraph();
```

```
}
```