Computer Graphics (22318)

Unit V : Introduction to Curves.

(weightage - 12 marks)

Prangal Saure (SY - comps)

5.1 : Curve generation : Arc generation using DDA algorithm, Interpolation.

Questions

Explain curve generation using interpolation. (4m).

Explain the process of DDA using arc. (4m).

Define fractal lines. (2m)

Interpolation

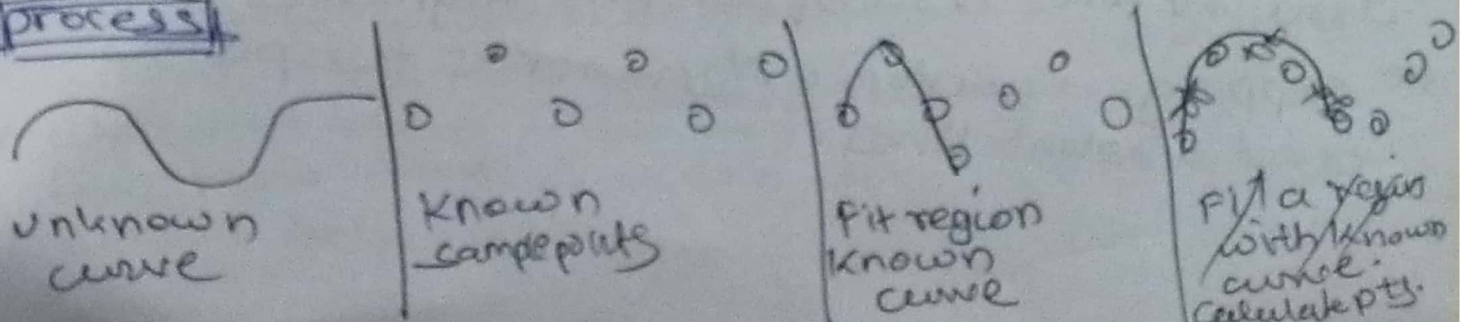
* A curve having no simple mathematical definition can be drawn by using interpolation.
* Interpolation is method of constructing new data points within range of discrete set of known data points.

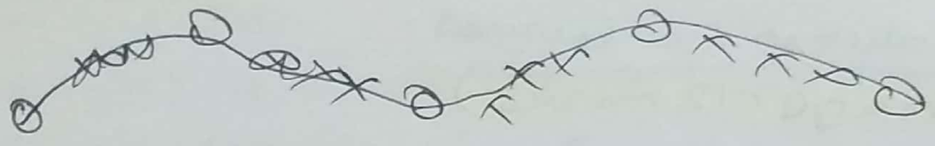
Rules for Interpolation :

1. The curve is specified by set of control points.
2. Positions of the control points controls the shape of curve.
3. This control points are fitted with continuoes polynomial function.
4. If the curve passes through the control points then it is called interpolation.

* Lagrange interpolation techniques is of of type of Interpolation.
* This technique is used when we have to draw curve by determining intermidate points between known sample point.
* When the given data points are not evenly distributed we can use this interpolation method to find solution.

process

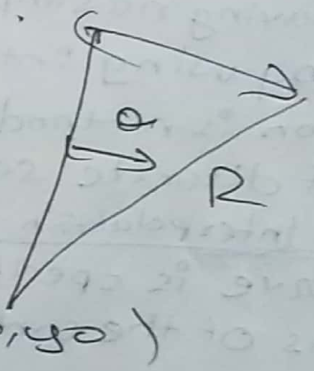




DDA Arcgenration

* Arc is a part of circle. If we join n numbers of arc then we get a complete circle.
* Digital differential Analyzer Algorithm uses the differential Equation of curve.
* A curve is an infinetely large set of points.
* Each point has two neighbours except endpoints.

$x_0\ y_0$ - Center of circle/curvature.
R = Radius of curvature.
θ - Angle.
x, y - The starting point of arc.



— Considering input we have to get a arc.

— This Eqn of arc in angle parameter can be given as.

$$x = R\cos A + x_0$$
$$y = R\sin A + y_0$$

* It is easy to implement.
* It is easy for generating differential eqns for ellipses and circle.
* Ellipse and circle can be implemented in hardware. So the display device can be capable of drawing arcs as well as line segments.
* Clipping algorithm only works for points and straight lines.

Fractal lines

* Fractals is a complex picture created using iteration and a single formula.
* Sometimes, object cannot be drawn with a given eqn or with a given geometry.
* Example, mountains, clouds.
* Examples so, their shape cannot be defined so in this case, we use fractals.
* It is an never ending pattern & it is infinity.

5.2 Types of curve : Hilbert's curve, Koch curve, B-spline, Bezier curve.

Types of curve
- → Beizer Curve } Spline curve representation
- → B-spline curve
- → Hilbert curve } Fractals representation.
- → Kotch Curve

Questions

Q1 Explain Koch curve neat diagram. (4M) ✱✱

Q2 write 'C' program to generate Hilbert's curve. (6M/4M) ✱ ✱✱

Q3 Draw Beizer curve, (2M)

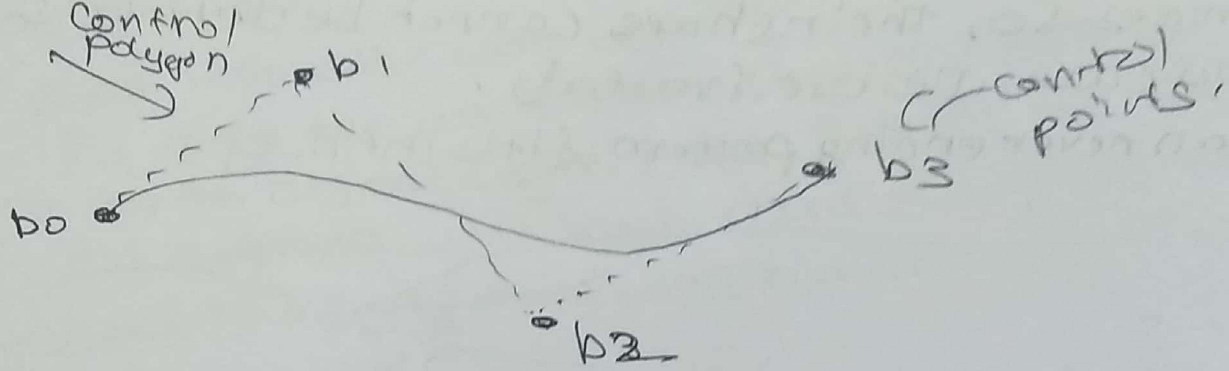
Beizer Curve

* The Beizer Curve is adequate/efficient for most graphic application. This curve requires four points (control).
* This four control points ~~p~~completely specifies the curve additional points cannot be ~~additional points can~~ added like B-Spline curve.
* We cannot extend Beizer curve, but we can take four more points & we can construct ~~four more~~ second Beizer curve.

Beizer curve is a parametric curve defined.
by set of control points,

Two points are end of curve.
Others determine the shape of curve.

The following curve is an example,

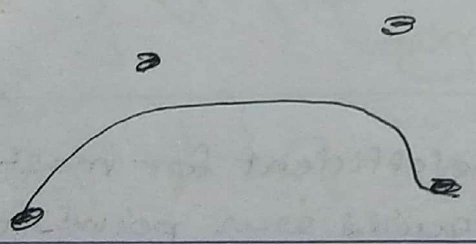


The beizer curve is defined by a set of
control points b0, b1, b2 and b3.

b0 & b3 are end points. that control
point at end.

b1 & b2, determine shape.

Another Example



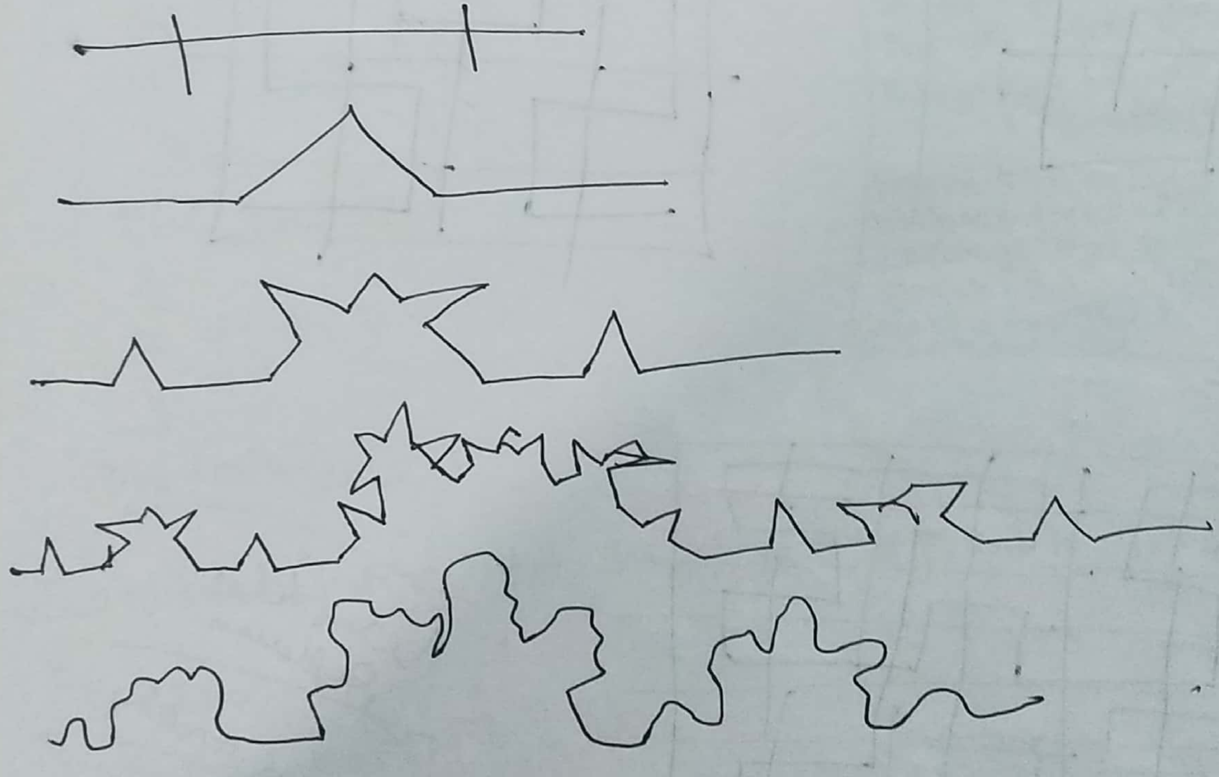
Properties

* They generally follow the shape of control polygon.
  which consists of segments join the control points.
* They always pass through first and last control pts.
* A Beizer curve generally follows the shape of the
  defining polygon.
* No straight line intersects a Beizer curve more times
  than it intersect control polygon.

Kotch Curve / Koch curve

* In Koch curve, begin at line segment.
* Divide line segment into three equal part.
* Replace center part by same measure as equilaterell triangle without third line.
* The again follow the same iteration
* This will give the curve which starts and ends at same place as the orginal segment but is built of nequal lengths segments, with each 1/3rd of orginal length.
* The curve has gained more wiggles and its length is 16/9 times orginal.

$$\boxed{D = \log_3 4 / \log 3 = 1.261}$$



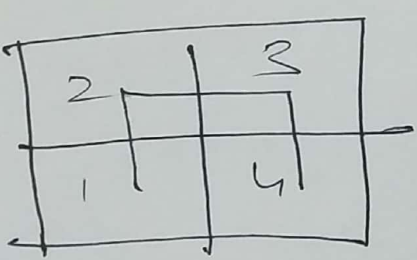
It is an fratical curve

It follows iteratione

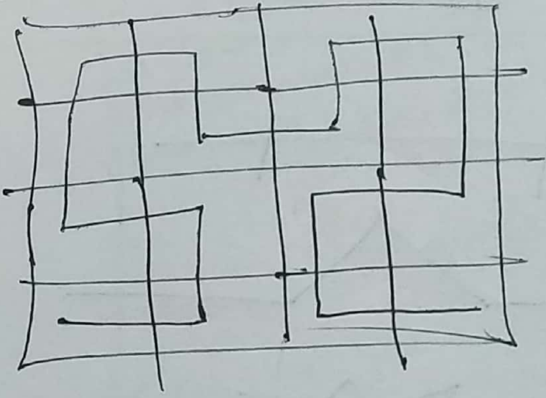
Hilbert curve

- The Hilbert curve is a space filling curve that visits every point in square grid with size of 2x2, 4x4, 8x8, 16x16 or any power of 2.
- It was described by David Hilbert in 1892.
- Application - used in image processing especially compression.
- This Hilbert curve can be built by following successive approximation.
- It fractal representation.

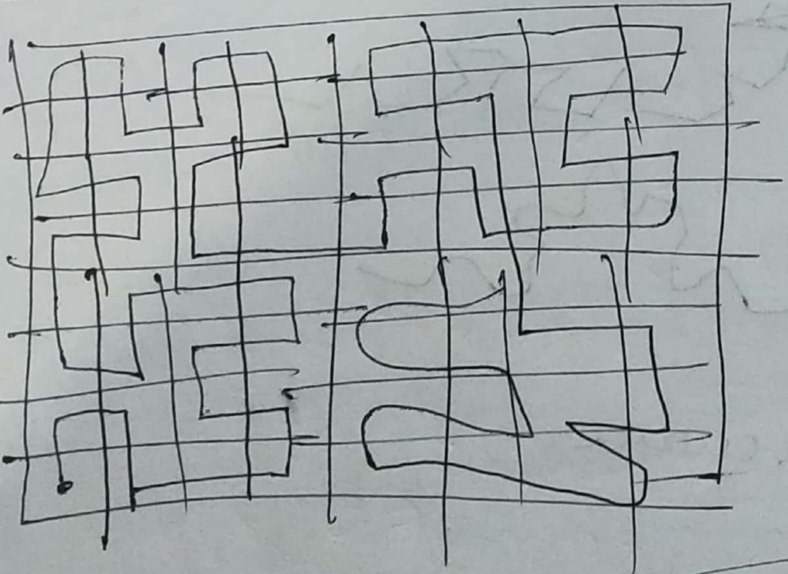
First Approximation



Second



Third



Working explain onwards

It follows iteration

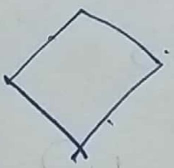
'C' Program for Hilbert Curve

```
#include<stdio.h>
#include<math.h>
#include<graphics.h>
void move(int j, int h, int &x, int &y)
 {
   if(j==1)
   {
    y-=h;
    }
   if(j==2)
   {
    x+=h;
    }
   if(j==3)
   {
    y+=h;
    }
   if(j==4)
   {
    x-=h;
   }
  lineto(x,y);
 }
void hilbert(int r, int d, int l, int u, int i, int h, int &x, int &y)
 { if(i>0)
 {
  hilbert(d,r,u,l,i,h,x,y);
  move(r,h,x,y);
  delay(100);
 hilbert(r,d,l,u,i,h,x,y);
  move(d,h,x,y);
 ~~hilbert~~ delay(100);
  hilbert(r,d,lu,i,h,x,y);
   move(l,h,x,y);
  delay(100);
}}hilbert(u,ld,r,i,h,x,y);
```

```
int main(){
int n,x1,y1;
int x0=50,y0=150,x,y,h=1
r=2,d=3,l=4,u=1;
clrscr();
printf("give value of n");
scanf("%d",&n);
x=x0;
y=y0;
int gm,gd=DETECT;
initgraph(&gd,&gm,"C:\\turbo3
          \\bgi");
moveto(x,y);
hilbert(r,d,l,u,n,h,x,y);
delay(100);
getch();
}closegraph();
return 0;
```

Hilbert curve

```
void move( int direction , int step, int& x, int& y )
{
  if ( direction = = 0)
    {
     y - = step;
    }
  else if (direction = = 1)
    {
     x + = step;
    }
  else if ( direction = = 2)
    {
     y + = step;
    }
  else if (direction = = 3)
    {
     x - = step;
    }
  line (x,y);
}
void hilbert (int right, int down, int left, int up,
              int iterations, int step, int &x, int &y)
 {
  if (iterations > 0)
    { iterations -- ;
    hilbert (d , r , u , l , i          )
    move (right, step, x, y)
    hilbert (r,
```



d, r, u, ~~l~~ ⇒ right
r, d, l, u ⇒ down.

~~ch~~ r, d, l, u ⇒ left.
u, l, d, r ⇒