Drawing Figures

DRAW DOTS

- A function using suffix i "expects" a 32-bit integer, but your system might translate int as a 16-bit integer

```
void drawDot(int x, int y) {
 glBegin(GL_POINTS);      // draws a dot at (x, y)
  glVertex2i(150, 130);
 glEnd();
}
```

- A better option will be to use:

```
void drawDot(GLint x, GLint y) {
 glBegin(GL_POINTS);      // draws a dot at (x, y)
  glVertex2i(150, 130);
 glEnd();
}
```

Drawing dot constellations

- A dot constellation is a pattern of dots or points.

Example :The Big Dipper

- The names and coordinates of the eight stars in the Big Dipper, a familiar sight in the night sky are given by the following ordered triplets:
  - {Dubhe, 289, 190}, {Merak, 320, 128}, {Phecda, 194, 101}, {Alioth, 129, 83}, {Mizar, 75, 73}, {Alcor, 74, 74}, {Alkaid, 20, 10}
- Since, we have few points here, so they can be hardwired into the source code.
- For larger data sets, it is more convenient to store them in a file and then read from that file to plot these points.
- Things to do:
  - Play with changing the color and size of the points.
  - Change the background color.

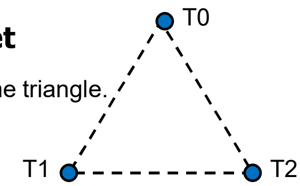
Drawing dot constellations

Example : Drawing the Sierpinski Gasket

- Sierpinski's gasket is a fractal.
- It can be produced by calling the drawDot() function many times.
- Denoting a kth point as $p_k = (x_k, y_k)$ each point is based on the previous point.
- The procedure goes like this:

Drawing dot constellations

Example : Drawing the Sierpinski Gasket

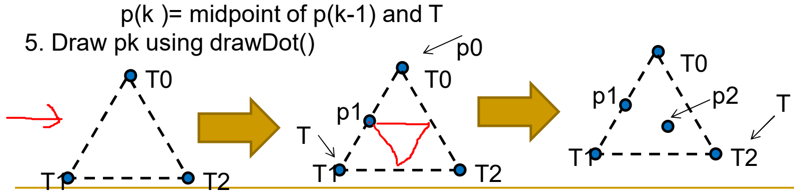


1. Choose three fixed points T0, T1 and T2 to form some triangle.

2. Choose the initial point p0 to be drawn by selecting one of the points T0, T1 and T2 at random.

Now iterate the following steps until the pattern is satisfactorily filled in or the maximum number of iterations has reached.

3. Choose one of the three points T0, T1 and T2 at random, and call it T.
4. Construct the next point p(k) as the mid-point between T and the previous point p(k-1). That is
   p(k)= midpoint of p(k-1) and T
5. Draw pk using drawDot()



Drawing dot constellations

Example: Drawing the Sierpinski Gasket

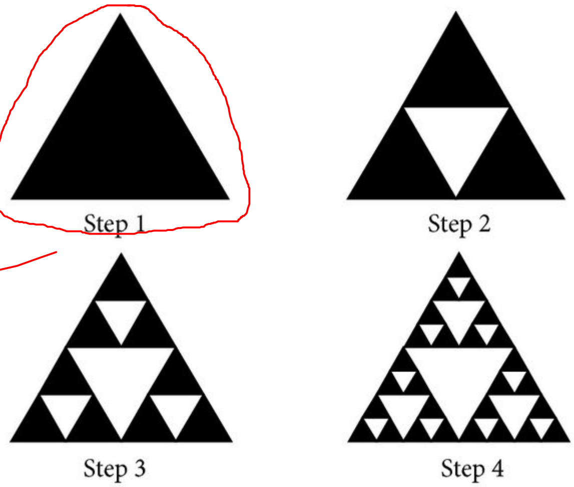
- How to go about it?
  - It is convenient to define a simple class GLintPoint, that describes a point whose coordinates are integers:

```
class GLintPoint {
public:
        GLint x, y;
};
```

  - We then move on to build and initialize an array of three such points T[0], T[1] and T[2].
  - There is no need to store each point, since we simply want to draw it and move on. So we set up a variable point to hold this changing point.
  - We use i=random(3) to choose one of the points T[i] at random. This function is defined as:

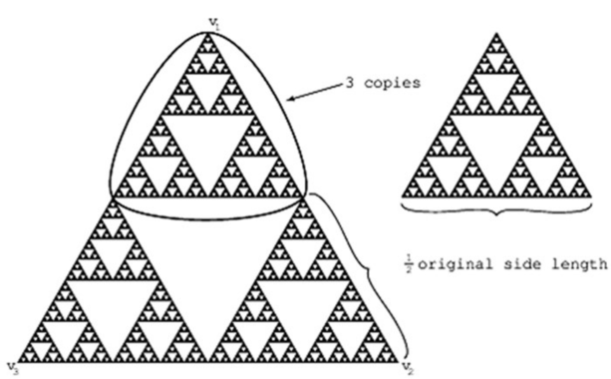
```
int random(int m) {
        return rand()%m;
}
```

Sierpinski Gasket



7

Sierpinski Gasket



8

Drawing dot constellations

Example : Drawing the Sierpinski Gasket

```
void Sierpinski(void) {
      GLintPoint T[3] = { {10,10}, {300, 30}, {200, 300} };
      int index = random(3);
      GLintPoint point = T[index];
      drawDot( point.x, point.y);
      for (int i=0; i<1000; i++) {
           index = random(3);
           point.x = ( point.x + T[index].x ) / 2;
           point.y = ( point.y + T[index].y ) / 2;
           drawDot( point.x, point.y);
      }
      glFlush();
}
```


Drawing dot constellations

Example: Simple "Dot Plots"

- This example deals with learning the behaviour of some mathematical function f(x) as x varies.
- Suppose we have $$f(x) = e^{-x}\cos(2\pi x)$$
  where x varies from x=0 to x=4.
- To plot this function we "sample" it at a collection of equispaced x-values and plot a dot at each coordinate pair (x, f(x)).
- Choosing a suitable increment, say 0.005, between consecutive x-values, the basic process will run as follows:

```
glBegin(GL_POINTS);
         for (GLdouble x=0; x<4.0; x += 0.005)
                  glVertex2d(x, f(x));
glEnd();
glFlush
```

Drawing dot constellations

Example :Simple "Dot Plots"

- Problem:
  1. The picture produced is impossibly tiny, because the values of x from 0 to 4 are mapped to only first four pixels at the bottom of the screen window.
  2. The negative values of f(x) lie below the window and are not visible.
- Solution:
  1. Scaling x: The first problem is solved if we scale x and then plot it. Consider a screen of width "screenWidth", the scaled x values can be obtained as

     sx = x * screenWidth / 4.0;

     So for x = 0, sx = 0 and for x = 4.0, sx = screenWidth.

3/25/2021 11

11

Drawing dot constellations

Example : Simple "Dot Plots"

- Solution:
  2. Scaling and Shifting y: The second problem is solved if we place the plot at the center of the screen window. Consider a screen of height "screenHeight", the scaled and shifted y values can be obtained as

     sy = (y + 1.0)* screenHeight / 2.0;

     So for y = -1.0, sy = 0 and for y = 1.0, sy = screenHeight.
- Note:
  - The conversion from x to sx and from y to sy are of form:

    sx = A x + B
    sy = C y + D → Affine Transformations
  - For properly chosen values of A, B, C and D.
  - A and C are scaling coefficients and
  - B and D are shifting coefficients.

3/25/2021 12

12

Drawing dot constellations

Example : Simple Dot Plot – Complete Program (1/2)

```
#include <math.h>
#include <gl/Gl.h>
#include <gl/glut.h>
const int screenWidth = 640;  // width of the screen window in pixels
const int screenHeight =480;  // height of the screen window in pixels
GLdouble A, B, C, D;                    // scaling and shifting coefficients
void myInit(void) {
        glClearColor(1.0, 1.0, 1.0, 0.0);     // background color is set to white
        glColor3f(0.0, 0.0, 0.0); // drawing color is set to black
        glPointSize(2.0);                   // a dot is 2 by 2 pixels
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0.0, (GLdouble)screenWidth, 0.0, (GLdouble)screenWeight);
        A = screenWidth / 4.0;
        B = 0.0
        C = D = screenHeight / 2.0;
}
```


Drawing dot constellations

Example : Simple Dot Plot – Complete Program (2/2)

```
void myDisplay(void) {
        glClear(GL_COLOR_BUFFER_BIT);                    // clear the screen
        glBegin(GL_POINTS);                              // draw the points
        for (GLdouble x=0; x<4.0; x += 0.005) {
             GLdouble func = exp(-x)*cos(2*3.14159265*x);
             glVertex2d( A*x + B, C*func + D );  }
        glEnd(); glFlush();
}
void main(int argc, char **argv) {
        glutInit(&argc, argv);                           // initialize the toolkit
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);     // set display mode
        glutInitWindowSize(screenWidth, screenHeight);   // set window size
        glutInitWindowPosition(100, 150);          // set window position on screen
        glutCreateWindow("Dot Plot of a Function");
        glutDisplayFunc(myDisplay);                // register display function
        myInit();
        glutMainLoop();                            // go for a perpeptual loop
}
```

Drawing polylines and polygons

Homework : Drawing line graphs

- A line graph is straight forward extension of the “dot plot” example.
- Suppose we have the following function to plot:
  $f(x) = 300 - 100 \cos(2\pi x/100) + 30 \cos(4\pi x/100)$
  $+ 6 \cos(6\pi x/100)$
  as x varies in steps of 3 for 100 steps.
- As a blowup of this figure would show a sequence of connected line segments; in a normal sized picture, they blend to give an impression of a smoothly varying curve.
- We need to do two changes in the code of previous example.
  1. Calculate the scaling and shifting coefficients A, B, C and D appropriately for the above function.
  2. Instead of using GL_POINTS, use GL_LINE_STRIP.
- The rest of the code remains the same.