Lab Report

Week 5

Ritobroto Maitra (1301CS50)

16/03/2016

Title

- Implementing

1. Basic Digital Differential Analyzer (DDA) algorithm (both Symmetric and Simple DDA).
2. Bresenham's line drawing algorithm and circle drawing algorithm.
3. A circle using Bresenham's line drawing algoritthm

Code Snippet : Simple DDA

OpenGL

```
void LineDDA(void) {
  /* (X1 Y1) and (X2 Y2) are the points we are
   * drawing ines between
   */
  double dx=(X2-X1);
  double dy=(Y2-Y1);
  double steps;
  float xInc,yInc,x=X1,y=Y1;

  steps=(abs(dx)>abs(dy))?(abs(dx)):(abs(dy));
  xInc=dx/(float)steps;
  yInc=dy/(float)steps;

  glClear(GL_COLOR_BUFFER_BIT);
  glBegin(GL_POINTS);
  glVertex2d(x,y);
  int k;

  for(k=0;k<steps;k++)
  {
    x+=xInc;
    y+=yInc;
    glVertex2d(round_value(x), round_value(y));
  }
  glEnd();
  glFlush();
}
```

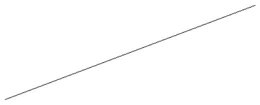

Figure 1 – Line drawn using Simple DDA

Code Snippet : Symmetric DDA

MATLAB

```
x1 = 100;
y1 = 200;
x2 = 500;
y2 = 500;

dx = x2-x1;
dy = y2-y1;

if dx > dy
    step = dx;
else
    step = dy;
end

incx = dx/step;
incy = dy/step;

x = round(x1:incx:x2);
y = round(y1:incy:y2);

plot(x, y);

axis([0, 1000, 0, 1000]);
```

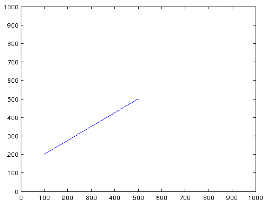

FIGURE 2 – Line drawn using Simple DDA (MATLAB)

Code Snippet : Symmetric DDA

```
void LineDDA(void) {
  /* (X1 Y1) and (X2 Y2) are the points we are
   * drawing ines between
   */
  double dx=(X2-X1);
  double dy=(Y2-Y1);
  double steps;
  float xInc,yInc,x=X1,y=Y1;

  steps=(abs(dx)>abs(dy))?(abs(dx)):(abs(dy));
  xInc=dx/(float)steps;
  yInc=dy/(float)steps;

  glClear(GL_COLOR_BUFFER_BIT);
  glBegin(GL_POINTS);
  glVertex2d(x,y);
  int k;

  for(k=0;k<steps;k++)
  {
    x+=xInc;
    y+=yInc;
    glVertex2d(round_value(x), round_value(y));
  }
  glEnd();
  glFlush();
}
```

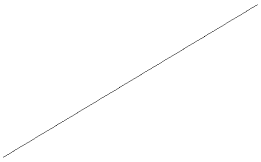

FIGURE 3 – Line drawn using Symmetric DDA

MATLAB

```
x1 = 100;
y1 = 200;
x2 = 500;
y2 = 500;

dx = x2-x1;
dy = y2-y1;

while dx>1 || dy>1
    dy = dy/2;
    dx = dx/2;
end

x = round(x1:dx:x2);
y = round(y1:dy:y2);

plot(x, y);

axis([0, 1000, 0, 1000]);
```

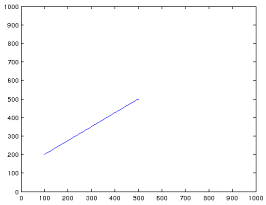

FIGURE 4 – Line drawn using Symmetric DDA

Code Snippet : Bresenham Line Drawing

```
void LineDDA(void)
{
  double dx=(X2-X1);
  double dy=(Y2-Y1);
  float xInc,yInc,x=X1,y=Y1;
  /* Find out whether to increment x or y */
  int dStart = 2*dy - dx;
  int dE = 2*dy;
  int dNE = 2*(dy - dx);
  /* Clears buffers to preset values */
  glClear(GL_COLOR_BUFFER_BIT);

  /* Plot the points */
  glBegin(GL_POINTS);
  /* Plot the first point */
  glVertex2d(x,y);
  int d;
  /* For every step, find an intermediate vertex */
  d = dStart;
  while(x<X2 && y<Y2)
  {
    if(d<0){
      d = d + dE;
    }else{
      d = d + dNE;
      y = y + 1;
    }
    x = x + 1;
    printf("%0.6lf %0.6lf\n",floor(x), floor(y));
    glVertex2d(round_value(x), round_value(y));
  }
  glEnd();

  glFlush();
}
```

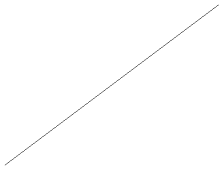

Figure 5 – Line drawn using Bresenham

MATLAB

```
dx = x2-x1;
dy = y2-y1;

d = 2*dy-dx;
dE = 2*dy;
dNE = 2*(dy-dx);

x = x1;
y = y1;

arrayx = zeros(1, dx);
arrayy = zeros(1, dx);

k=2;
arrayx(1)=x;
arrayy(1)=y;
while x<x2 && y<y2
    if d<0
        d = d + dE;
    else
        d = d + dNE;
        y = y + 1;
    end
    x = x+1;
    arrayx(k) = x;
    arrayy(k) = y;
    k = k + 1;
end
plot(arrayx, arrayy);
axis([0, 1000, 0, 1000]);
```

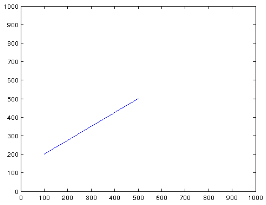

Figure 6 – Line drawn using Bresenham(MATLAB)

Code Snippet : Bresenham Circle Drawing

```c
void CircleDDA(void)
{
  int x=0,y=R, xc=0, yc=0;
  /* Find out whether to increment x or y */
  int p;
  /* Clears buffers to preset values */
  glClear(GL_COLOR_BUFFER_BIT);

  /* Plot the points */
  glBegin(GL_POINTS);
  /* Plot the first point */

  x=0;
  y=R;
  glVertex2d(x, y);
  p=1-R;
  for(x=0;x<=y;x++)
  {
    if (p<0){
      p=(p+(2*x)+1);
    }else{
      y=y-1;
      p=p+((2*(x-y)+1));
    }
    glVertex2d(xc+x,yc-y);
    glVertex2d(xc-x,yc-y);
    glVertex2d(xc+x,yc+y);
    glVertex2d(xc-x,yc+y);
    glVertex2d(xc+y,yc-x);
    glVertex2d(xc-y,yc-x);
    glVertex2d(xc+y,yc+x);
    glVertex2d(xc-y,yc+x);
  }
  glEnd();

  glFlush();
}
```

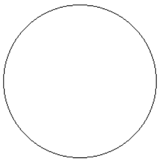

Figure 7 – Circle drawn using Brassenhams

MATLAB

```
R = 200;

d = 1-R;
x = 0;
y = R;

arrayx = zeros(1, R);
arrayy = zeros(1, R);

k=2;
arrayx(1)=x;
arrayy(1)=y;
while x<=y
    if d<0
        d = d + (2*x) + 1;
    else
        d = d + (2*(x-y)) + 1;
        y = y - 1;
    end
    x = x+1;
    arrayx(k) = x;
    arrayy(k) = y;
    k = k + 1;
end

arrayx = arrayx(find(arrayy,1,'first'):find(arrayy,1,'last'));
arrayy = arrayy(find(arrayy,1,'first'):find(arrayy,1,'last'));

farrayx = [arrayx fliplr(arrayy) arrayy fliplr(arrayx) -arrayx -fliplr(arrayy) -arrayy -fliplr(arrayx)];
farrayy = [arrayy fliplr(arrayx) -arrayx -fliplr(arrayy) -arrayy -fliplr(arrayx) arrayx fliplr(arrayy)];

plot(farrayx, farrayy);

axis([-500, 500, -500, 500]);
```

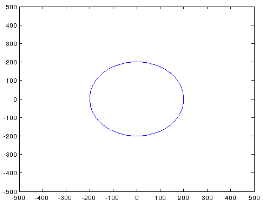

Figure 8 – Circle drawn using Brassenhams(MATLAB)

Discussion

	Digital Differential Analyzer	**Bresenhams Line Drawing Algorithm**
Arithmetic	Floating points	Integer Arithmetic.
Operations	Uses multiplication and division.	Only Addition Subtraction
Speed	Slower due to real arithmetic	Faster
Accuracy	Not as accurate and efficient as Bresenham.	More efficient and much accurate.
Roundoff	Round off to integer nearest to the line.	No round off, takes incremental values.