**SSN COLLEGE OF ENGINEERING, KALAVAKKAM  
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
UCS1712 – GRAPHICS AND MULTIMEDIA LAB ------------------------------------------------------------------------------------------------------------**

**Lab Exercise 2 : DDA Line Drawing Algorithm in C++ using OpenGL**

**Aim:**

To perform DDA line drawing algorithm in C++ using OpenGL

**Algorithm:**

ProcedurelineDDA(xa,xb,ya,yb:integer);

Var

dx,dy,steps,k : integer;  
 xIncrement, yIncrement, x , y: real; Begin

dx:=xb-xa;  
 dy:=yb-ya;  
 ifabs(dx)>abs(dy)thensteps:=abs(dx) elsesteps:=abs(dy)  
 xIncrement:=dx/steps;  
 yIncrement:=dy/steps;  
 x:=xa;  
 y:=ya;

setPixel(round(x),round(y),1); fork:=1 to steps do  
 Begin

x:=x+xIncrement;  
 y:=y+yIncrement;  
 setPixel(round(x), round(y),1); End

End {lineDDA}

**To plot points that make up the line with endpoints (x0,y0) and (xn,yn) using the DDA line drawing algorithm.**

Case 1: +ve slope Left to Right line  
Case 2: +ve slope Right to Left line  
Case 3: -ve slope Left to Right line

Case 4: -ve slope Right to Left line

Each case has two subdivisions -

**(i) |m|<= 1 (ii) |m|>1  
Note that all four cases of line drawing must be given as test cases.**

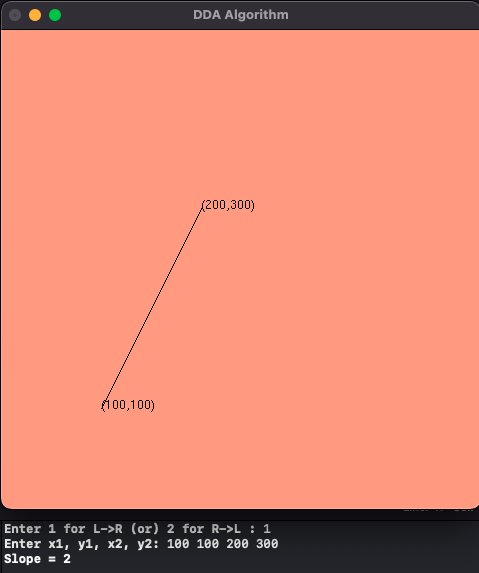
***Source Code:***

| #include<GLUT/glut.h> #include<iostream> #include<cmath> #include<string> using namespace std; void drawString(float x, float y, const char \*string){  glRasterPos2f(x, y);  for(const char\* c = string;\*c!='\0';c++)  glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12, \*c); } void myInit() {  glClearColor(1.0,0.6,0.5,0.0);  glPointSize(1);  glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  gluOrtho2D(0.0,480.0,0.0,480.0); } void myDisplay() {  glClear(GL\_COLOR\_BUFFER\_BIT);  int x1,y1,x2,y2,dx,dy,choice;  float x,y;  cout<<"Enter 1 for L->R (or) 2 for R->L : ";  cin>>choice;  cout<<"Enter x1, y1, x2, y2: ";  cin>>x1>>y1>>x2>>y2;  glColor3f(0.0,0.0,0.0);  glBegin(GL\_POINTS);  glVertex2f(x1,y1);  glEnd();    dx=x2-x1;  dy=y2-y1;  float m=(float)dy/dx;  cout<<"Slope = "<<m<<endl;    /\* Positive Slope L->R \*/  if(choice==1 && m>=0.0){  x=x1;  y=y1;  if(abs(m)<=1.0){  while(x!=x2){  y=y+m;  x+=1;  glBegin(GL\_POINTS);  glVertex2f(x,round(y));  glEnd();  }  }  else{  while(y!=y2){  x=x+(1/m);  y+=1;  glBegin(GL\_POINTS);  glVertex2f(round(x),y);  glEnd();  }  }  }  /\* Positive Slope R->L \*/  else if(choice==2 && m>=0){  x=x2;  y=y2;  if(abs(m)<=1.0){  while(x!=x1){  y=y-m;  x-=1;  glBegin(GL\_POINTS);  glVertex2f(x,round(y));  glEnd();  }  }  else{  while(y!=y1){  x=x-(1/m);  y-=1;  glBegin(GL\_POINTS);  glVertex2f(round(x),y);  glEnd();  }  }  }  /\* Negative Slope L->R \*/  else if(choice==1 && m<0){  x=x1;  y=y1;  if(abs(m)<=1.0){  while(x!=x2){  y=y-abs(m);  x+=1;  glBegin(GL\_POINTS);  glVertex2f(x,round(y));  glEnd();  }  }  else{  while(y!=y2){  x=x+(1/m);  y-=1;  glBegin(GL\_POINTS);  glVertex2f(round(x),y);  glEnd();  }  }  }  /\* Negative Slope R->L \*/  else if(choice==2 && m<0){  x=x2;  y=y2;  if(abs(m)<=1.0){  while(x!=x1){  y=y+abs(m);  x-=1;  glBegin(GL\_POINTS);  glVertex2f(x,round(y));  glEnd();  }  }  else{  while(y!=y1){  x=x-abs(1/m);  y+=1;  glBegin(GL\_POINTS);  glVertex2f(round(x),y);  glEnd();  }  }  }    drawString(x1,y1,("("+to\_string(x1)+","+to\_string(y1)+")").c\_str());  drawString(x2,y2,("("+to\_string(x2)+","+to\_string(y2)+")").c\_str());  glFlush(); } int main(int argc,char\* argv[]) {  glutInit(&argc,argv);  glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);  glutInitWindowSize(480,480);  glutCreateWindow("DDA Algorithm");  glutDisplayFunc(myDisplay);  myInit();  glutMainLoop();  return 1; } |
| --- |

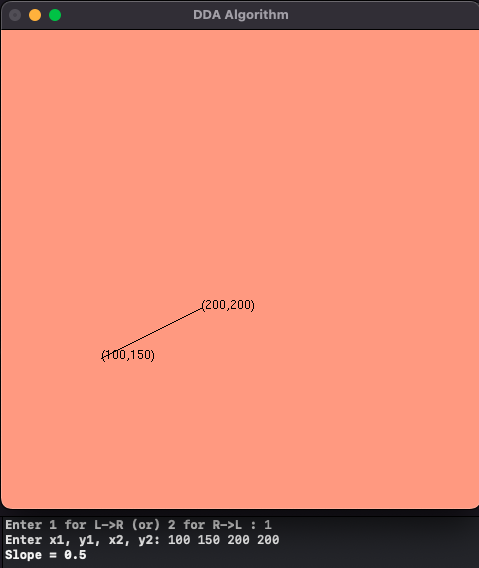
***Output:***

***Case 1: +ve slope Left to Right line***

***m>1***

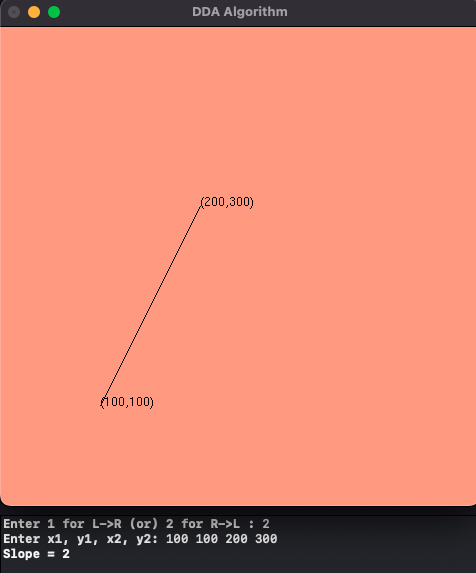
******

***m<=1***

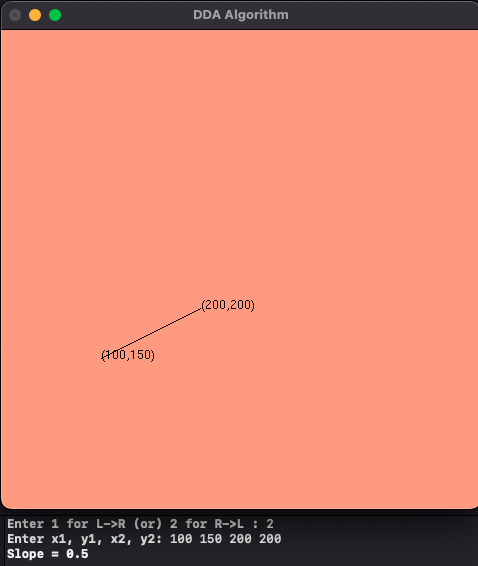
******

***Case 2: +ve slope Right to Left line***

***m>1***

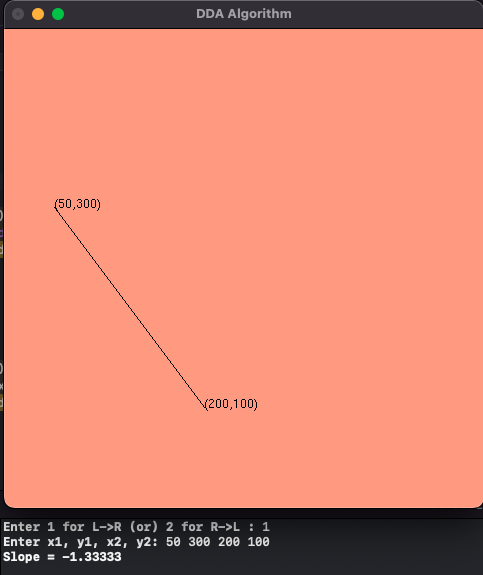
******

**m<=1**

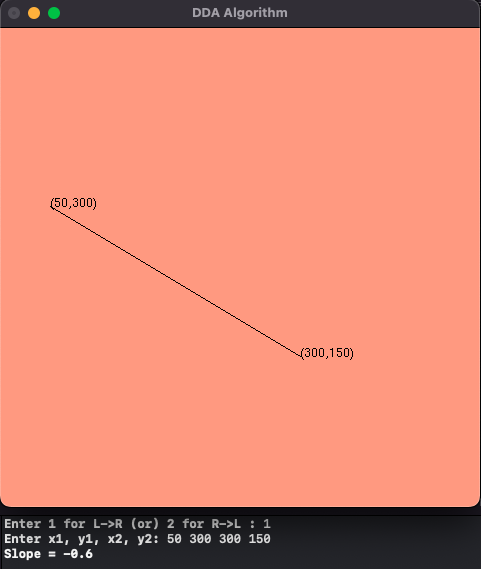
****

**Case 3: -ve slope Left to Right line**

**m>1**

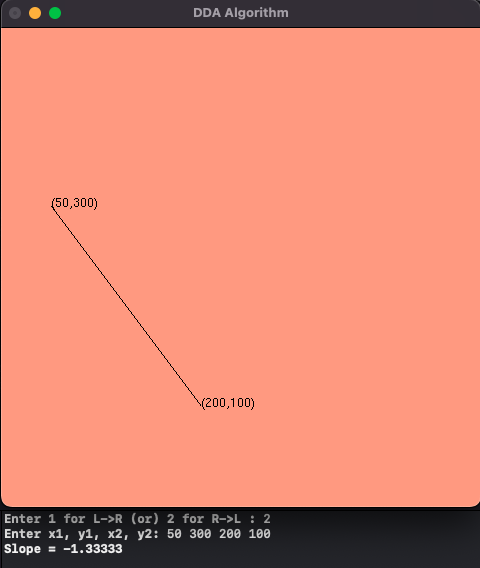
****

**m<=1**

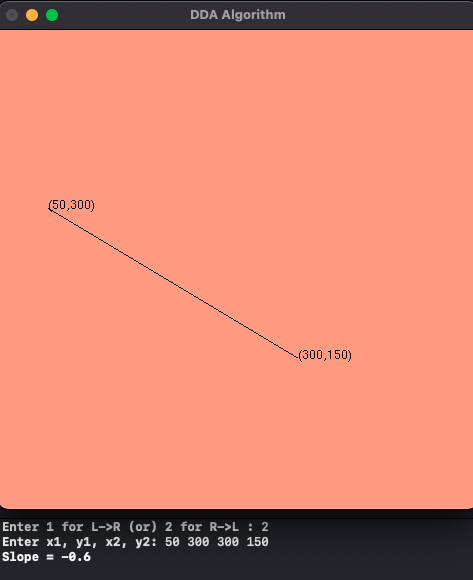
****

**Case 4: -ve slope Right to Left line**

**m>1**

****

**m<=1**

****

**Learning Outcome;**

Learnt how to perform DDA algorithm in C++ using OpenGL