# Line Drawing Algorithms

CS116A

Chris Pollett

Aug 30, 2004.

# Introduction

- Coordinates
- OpenGL Types, Point Drawing
- OpenGL line drawing
- DDA Algorithm
- Bresenham's Algorithm

# Coordinates

- Location on monitor called screen coordinates
- The minimum (x,y,z) and maximum (x,y,z) coordinates of an object called coordinate extent (bounding box). In 2d bounding rectangle
- Screen coordinates y called scan line number, x called column number
- (0,0) will be bottom left of screen. Can have absolute or relative positioning

# OpenGL and Coordinates

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(xmin, xmax, ymin, ymax);
```

# OpenGL Types and Point Drawing

- GLbyte, GLshort, GLint, GLfloat, Gldouble, Glboolean
- The call to set a point will be of the form glVertex*, where * says dimensions and type. Ex. glVertex2i (), glVertex3f

# More point drawing…

- To draw points we'd use code:

```
glBegin(GL_POINTS);
    glVertex2i(50,100);
    glVertex2i(75,150);
    glVertex2i(100,200);
glEnd();
```

# OpenGL Line Drawing

- If we want to draw lines could do:

```
int p1[] = {50,100}; /*define p2, etc like this */
glBegin(GL_LINES); /* GL_LINE_STRIP gives
   polyline, GL_LINE_LOOP for closed polyline */
     glVertex2iv(p1); /* the v is for vector--
     We need it because p1 is an array*/
     glVertex2iv(p2);
     glVertex2iv(p3);
glEnd();
```

# DDA algorithm

- Digital Differential Analyzer (DDA)
- Recall equation of a line y=mx+b
- Want to draw lines without gaps.
- Assume line segment specified using endpoints $(X_0,Y_0)$ and $(X_{end},Y_{end})$. So $m=(Y_{end}-Y_0)/(X_{end}-X_0)$.
- Let $(X_i,Y_i)$ be i+1st point we plot.
- First, will consider the case $|m| <1$. What do we do if $|m|=1$?

# More DDA

- Assume $X_i, Y_i$ each specified with floats
- To plot a point, the coordinate $X_{i+1}$ will be just $X_i+1$ and $Y_{i+1}$ will be $Y_i+m$. When plotted the actual point comes from round the Y value.
- If $|m|>1$ then interchange the roles of X and Y: $Y_{i+1}$ will be just $Y_i+1$ and $X_{i+1}$ will be $X_i +1/m$.
- Drawback of this algorithm is uses floating point operations

# Bresenham's Algorithm

- Advantage only integer values used.
- To keep things simple assume $0<m<1$.
- From position $(X_i,Y_i)$ one of $(X_i+1,Y_i)$ or $(X_i+1,Y_i+1)$ will be the next point to plot.
- Let $Y=m(X_i+1) + b$

# More Bresenham's Algorithm

- $d_{lower} = Y - Y_i$
- $d_{upper} = (Y_i+1)-Y$
- Sign of $d_{upper} - d_{lower}$ says which of $Y_i$ or $Y_i+1$ to use.
- Turns out $p_i=\Delta x(d_{upper} - d_{lower})$ is integer. Sign of this says same thing.
- Moreover, can compute $p_{i+1}$ fast

# Still more Bresenham

- $p_{i+1} = p_i + 2\Delta y - 2\Delta x(Y_{i+1} - Y_i)$
- $p_0 = 2\Delta y - \Delta x$