

Graphics Primitives

Eric C. McCreath

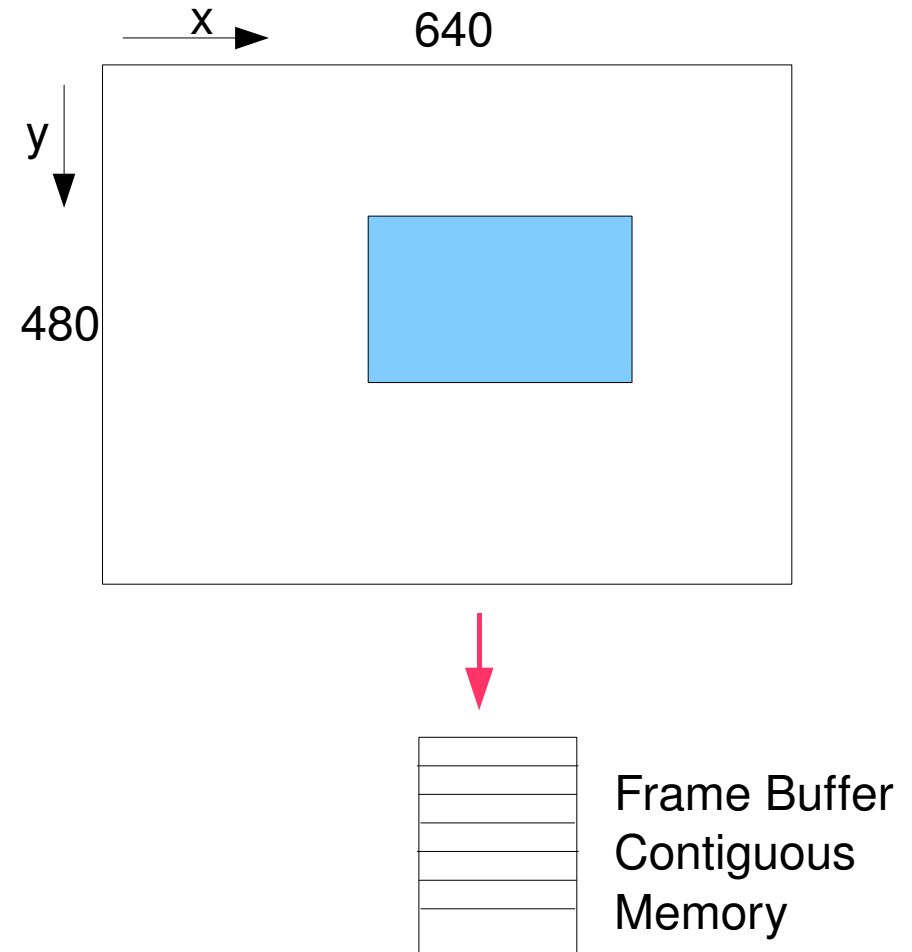
School of Computer Science
The Australian National University
ACT 0200 Australia

ericm@cs.anu.edu.au

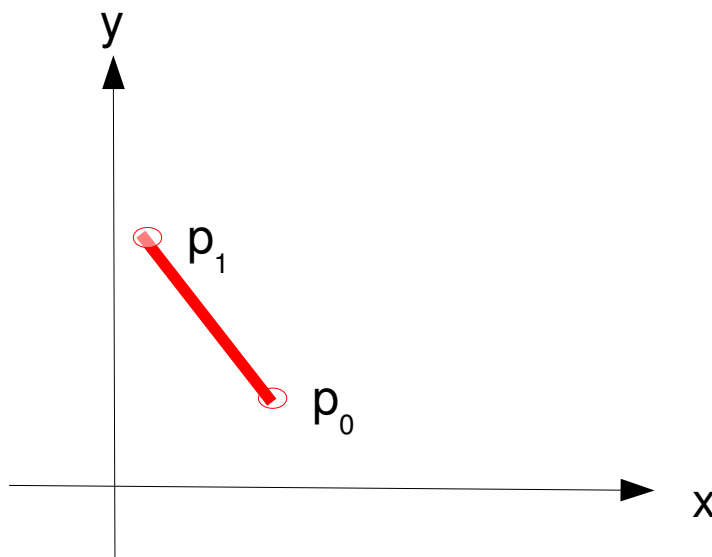
- Coordinates, Frame Buffers, Pixels – Ch3
- Line Drawing(Bresenham's), Circle Drawing - Ch3
- Java Graphics - see the Java API

Note these slides do not contain much content, most of this will be given via demonstration during the lecture.

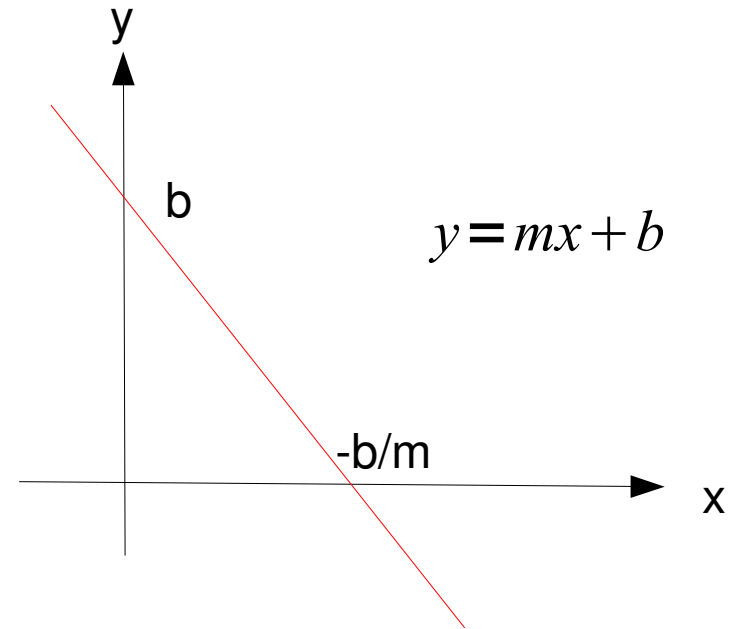
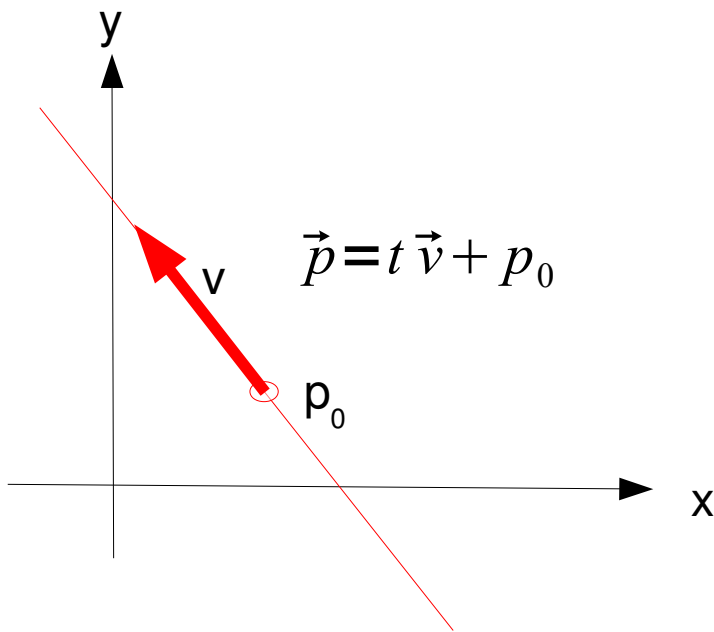
- screen co-ordinates
- absolute/relative co-ordinates
- Which way is up will depend on graphics API you are using.
- A function must map co-ordinates(relative/window/screen) to addresses in the frame buffer.



- We often want to be able to draw a line segment.



- How do you describe a line? (get out your highschool math)



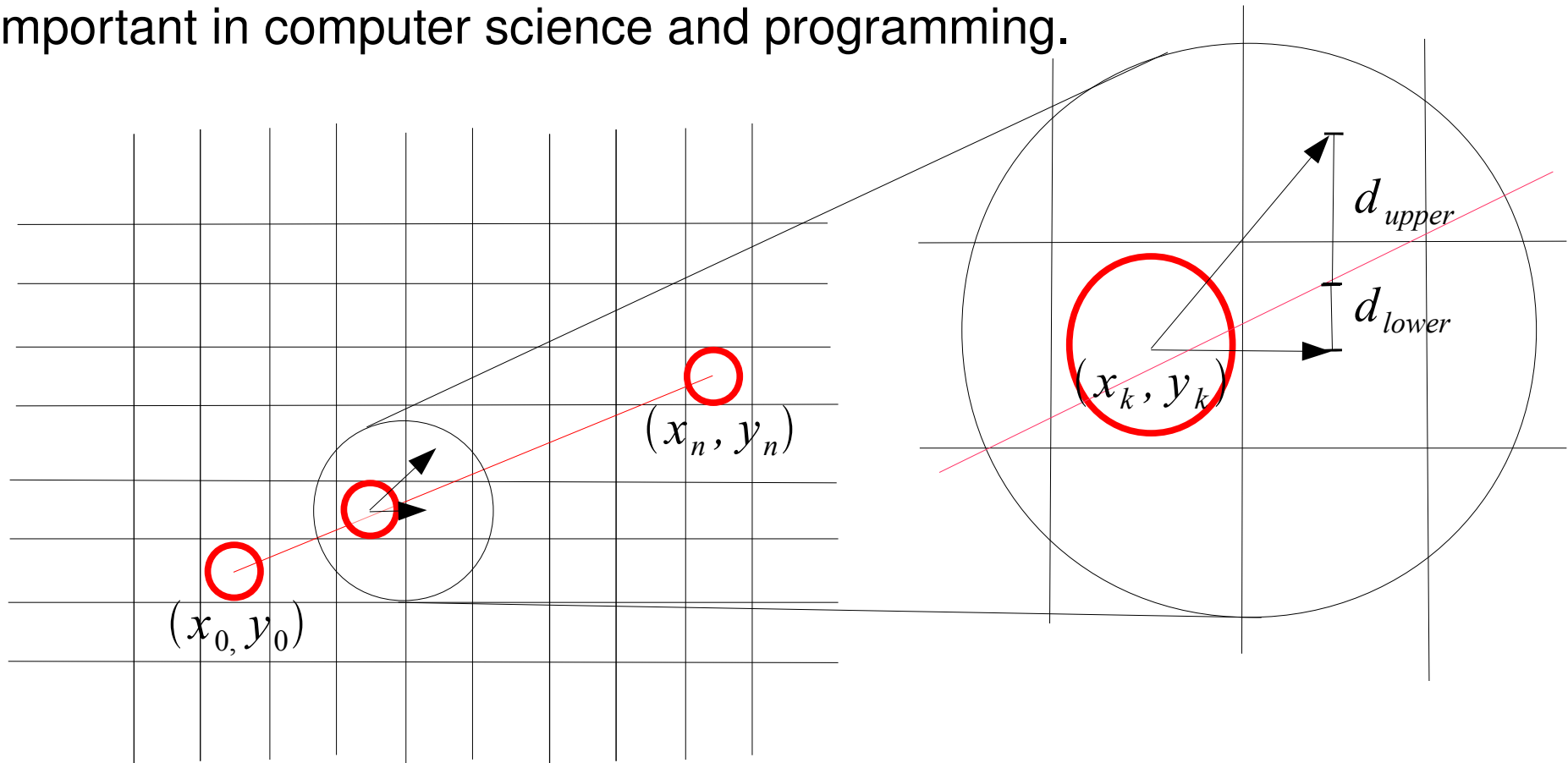
- Digital Differential Analyzer (DDA) Algorithm – use floating point operations and add the change in position for each pixel. The pixel location is rounded.

```
private static void lineDDA(BufferedImage buff, int x1, int y1, int x2,
                           int y2, int rgb) {

    int dx = x2 - x1;           Could be improved by multiplying values by step
    int dy = y2 - y1;           and doing calculations in integer arithmetic.
    float x = x1;               Still needs 2 divisions for every point in the line segment.
    float y = y1;

    int steps = Math.max(Math.abs(dx), Math.abs(dy));
    float xinc = (float) dx / steps;
    float yinc = (float) dy / steps;
    buff.setRGB(Math.round(x), Math.round(y), rgb);
    for (int i = 0; i < steps; i++) {
        x += xinc;
        y += yinc;
        buff.setRGB(Math.round(x), Math.round(y), rgb);
    }
}
```

- Bresenham's Line Algorithm transforms the line drawing algorithm to simplify the main loop of line drawing. This type of approach is important in computer science and programming.



$$y = mx + b$$

Assume : $0 \leq m < 1$

$$m = \frac{\Delta y}{\Delta x} = \frac{y_n - y_0}{x_n - x_0}$$

- We want a way of deciding to go up or stay on the same line.

$$y = m(x_k + 1) + b$$

$$d_{lower} = y - y_k = m(x_k + 1) + b - y_k$$

$$d_{upper} = y_k + 1 - y = y_k + 1 - m(x_k + 1) - b$$

If this is positive we go up otherwise we go across.

Let :

$$p_k = \Delta x (d_{lower} - d_{upper})$$

$$= 2 \Delta y x_k - 2 \Delta x y_k + c$$

$$p_{k+1} = 2 \Delta y x_{k+1} - 2 \Delta x y_{k+1} + c$$

$$p_{k+1} - p_k = 2 \Delta y (x_{k+1} - x_k) - 2 \Delta x (y_{k+1} - y_k)$$

$$= \begin{cases} 2 \Delta y, & \text{if } p_k < 0 \\ 2 \Delta y - 2 \Delta x, & \text{otherwise} \end{cases}$$

We have :

$$p_0 = 2 \Delta y - \Delta x$$

Using the difference formula we can calculate:

$$p_1, p_2, p_3, \dots, p_{n-1}$$

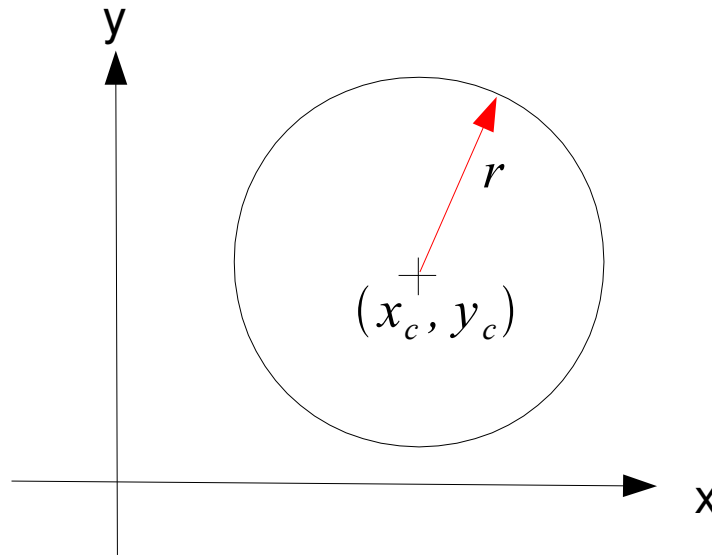
and draw the line as we go

- Back to some basic math:

$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

$$x = x_c + r \cos \theta$$

$$y = y_c + r \sin \theta$$



- In some demos we will look at the following classes:
 - Dimension
 - Graphics (setColor, setFont, drawImage, drawLine, drawRect, fillRect, drawArc)
 - Color
 - Font
 - Image
 - ImageIcon
 - BufferedImage