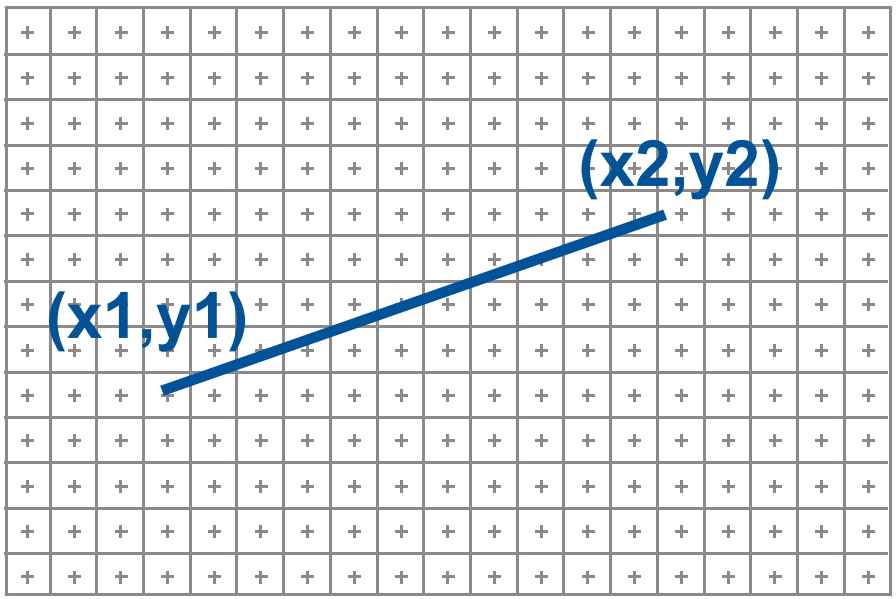
Framebuffer Model

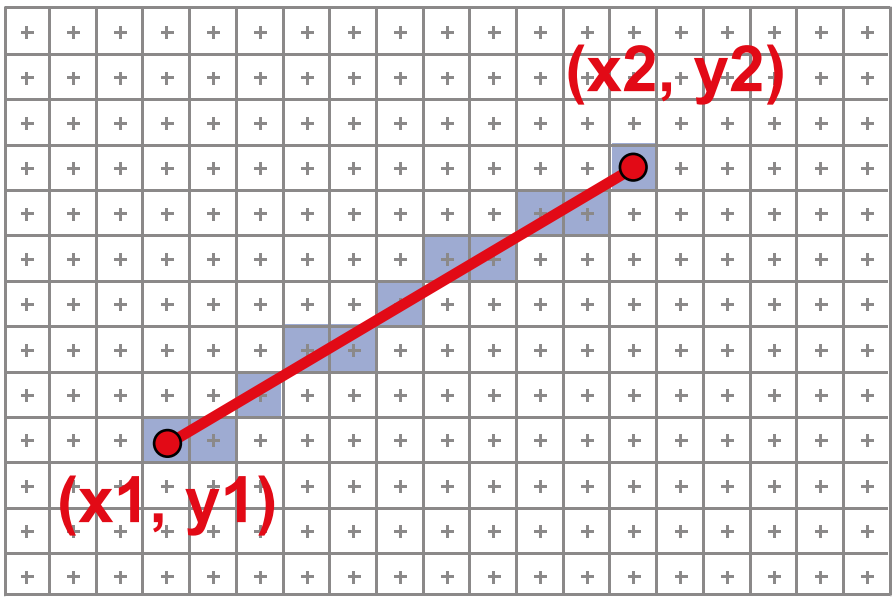
- Raster Display: 2D array of picture elements (pixels)
- Pixels individually set/cleared (greyscale, color)
- Window coordinates: pixels centered at integers




```
glBegin(GL_LINES)
glVertex3f(...)
glVertex3f(...)
glEnd();
```

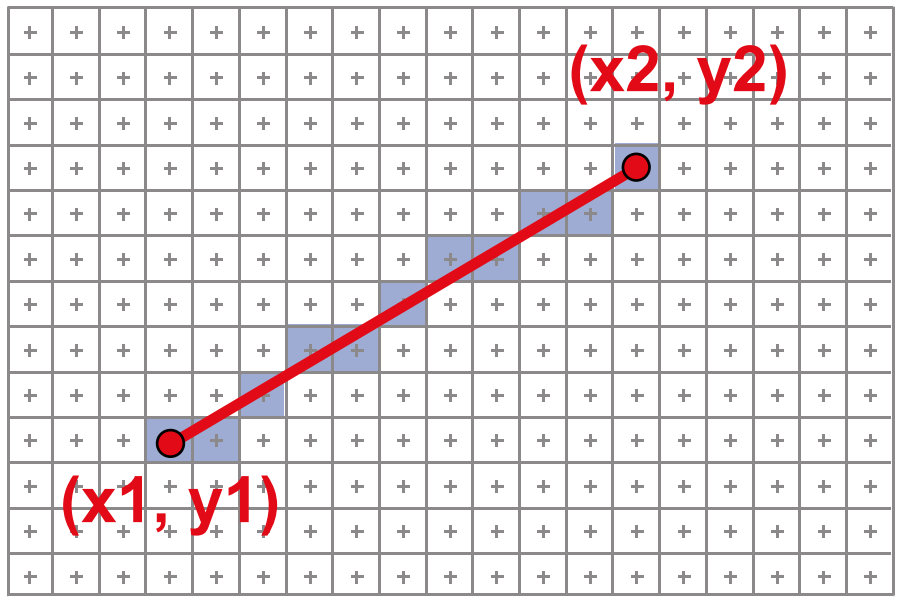
Scan Converting 2D Line Segments

- Given:
  - Segment endpoints (integers x1, y1; x2, y2)
- Identify:
  - Set of pixels (x, y) to display for segment



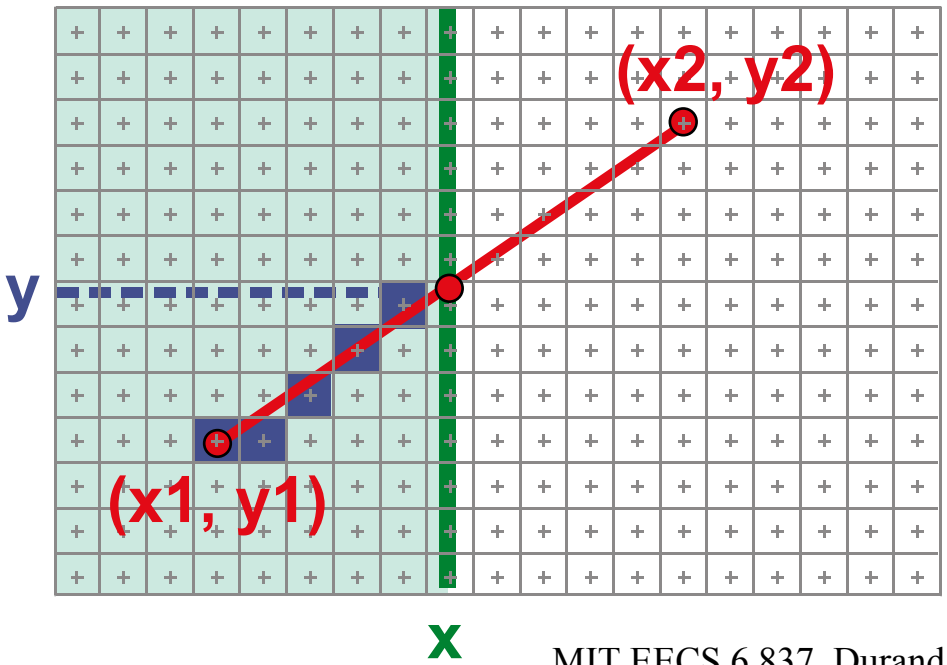
Line Rasterization Requirements

- Transform continuous primitive into discrete samples
- Uniform thickness & brightness
- Continuous appearance
- No gaps
- Accuracy
- Speed



Naive Line Rasterization Algorithm

- Simply compute y as a function of x
  - Conceptually: move vertical scan line from x1 to x2
  - What is the expression of y as function of x?
  - Set pixel (x, round (y(x)))



$$y = y1 + \frac{x - x1}{x2 - x1}(y2 - y1)$$

$$= y1 + m(x - x1)$$

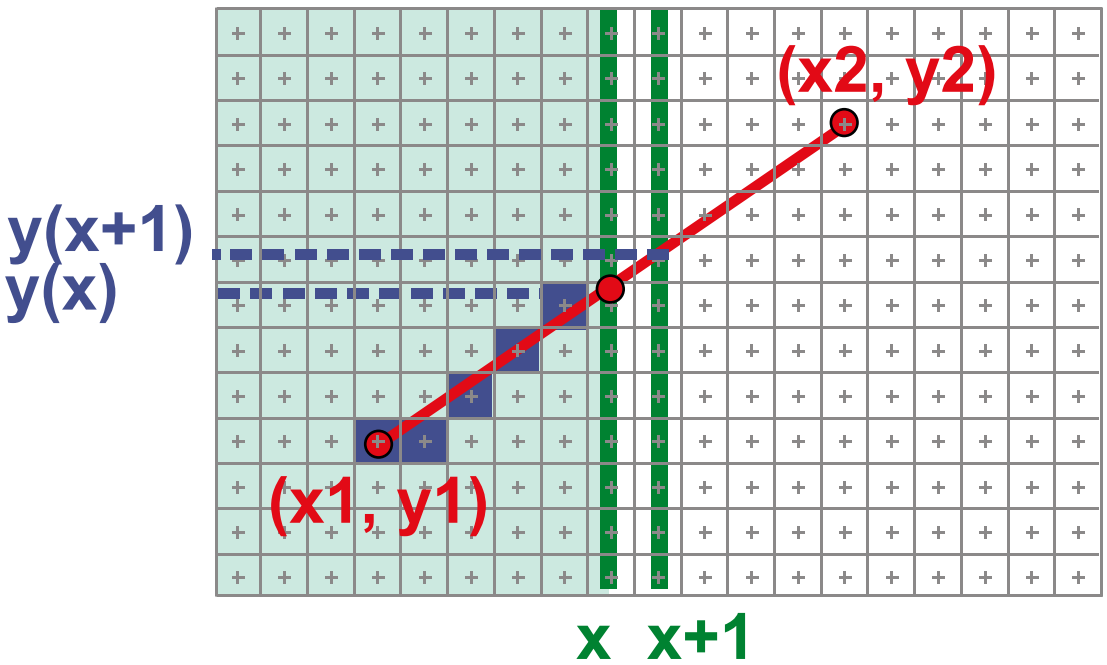
$$m = \frac{dy}{dx}$$

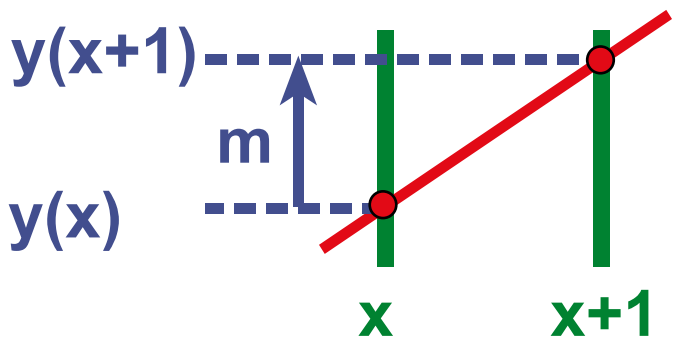
Efficiency

- Computing y value is expensive

$$y = y1 + m(x - x1)$$

- Observe: $y \mathrel{+}= m$ at each x step ($m = dy/dx$)





Bresenham's Algorithm (DDA)

- Select pixel vertically closest to line segment
  - intuitive, efficient,
    pixel center always within 0.5 vertically
- Same answer as naive approach