Lecture 1: 4 September 1997

Welcome to 6.837 !

Course Staff

Please sign sheet, **with athena id**

Course info, calendar handouts; text

Class page:

`graphics.lcs.mit.edu/classes/6.837/F97/homepage.html`

Course notes

Today:

- Course overview
- Prerequisites
- Assignments and grading policy
- 2D Segment, Circle rasterization
- Course web registration (required)
- Assignment 1: Fusion system

Course Overview

What is computer graphics? Why do it?
Easy answers: "Making pictures. Because it's fun."

Better: "Application of computational processes to simulation data to simulate or predict appearance."
"To improve our understanding. Because it's fun."

Simulations require input! **Modeling, Computer vision.**

Most common simulation: **Rendering, Optics. (Haptics!)**

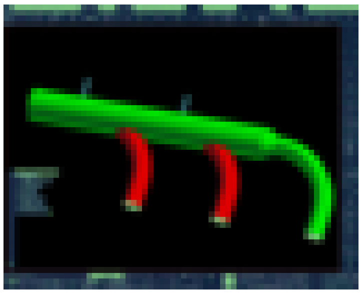
Also simulate motion, time-evolution: **Kinematics, Physics.**

All are grounded in **Geometry**, **Coordinates**, **Computation**.

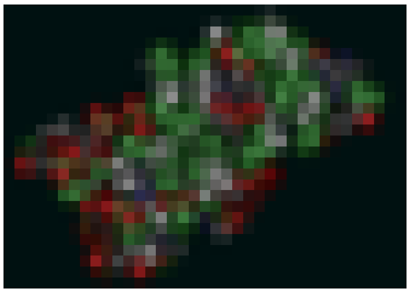
Also: interaction; theoretical concerns; systems issues.

Graphics Examples

Computer-Aided Design (CAD)

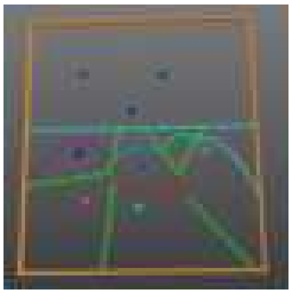


Scientific Visualization

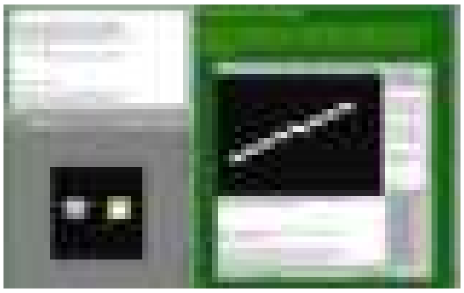


Graphics Examples

Mathematical Visualization



Education (e.g., Fusion system)



Graphics Examples

Entertainment

Military (planning, training, etc.)

Graphics Examples

Engineering Analysis

Medicine (Image-Guided Surgery)

Other examples ?

Course Overview

2D Preliminaries (framebuffer model, etc.)
Coordinate Systems and Transformations
Classical Rendering Pipeline
Surface, Volume Modeling
Ray Tracing and Global Illumination
Texture Mapping and Realism
Hardware, Algorithmic Acceleration
Careers in Graphics
Special Guest Lectures, Graphics shows

Prerequisites

18.06 (Linear Algebra):
- Vectors (dot, cross products, etc.)
- Matrices and matrix operators
- Solving linear systems of equations
- Bases: spanning, orthogonal, orthonormal

6.046J (Algorithms):
- Algorithm complexity, "big O" notation
- Upper bound (can't be harder than...)
- Lower bound (at least as hard as...)
- Sorting, convex hulls, tree traversal, etc.

Some C Programming Experience
- Assignments assume some C proficiency
- You will pick up Java, C++, OpenInventor...

Substitutions:
- Analogous courses; relevant experience; osmosis
- An enthusiastic attitude; tolerance for pain
- Evening review sessions (David, Jonathan, Luke)

Assignments and Grading Policy

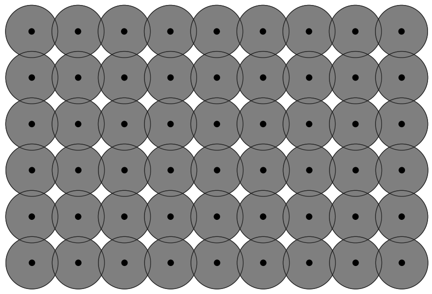
Programming assignments (six, 60%)
- Due Fridays at 5pm, weeks 2-7
- Asst 1: 2D Segment rasterization (Fusion, Java)
- Asst 2: 2D Segment clipping (Fusion, Java)
- Asst 3A,B: 3D modeling (Inventor, C, C++)
- Asst 4: Polygon Scan Conversion (Inventor, C++)
- Asst 5: Ray Casting/Tracing (Inventor, C++)

Late policy
- Assignments time-stamped by `turnin` (or equivalent)
- No late assignments accepted, period,
  except with a letter from the Dean.

Final project (40%)
- Examples, brainstorming week 7
- Team formation, written proposals week 8
- Progress meetings with TAs, weeks 8-12
- Written report week 13
- Team presentation week 14

Framebuffer Model

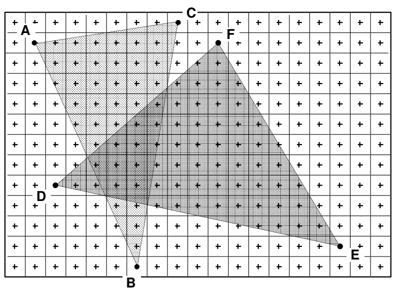
Raster Display: 2D array of picture elements (pixels)
Pixels individually set/cleared (greyscale, color)
Window coordinates: pixel centered at integers



2D Scan Conversion

Geometric "primitive:" specified or rendered object
- 2D: point, line, polygon, circle...
- (3D: point, line, polyhedron, sphere...)

Challenge: primitives are continuous; screen is discrete
- Solution: compute & display discrete approximation

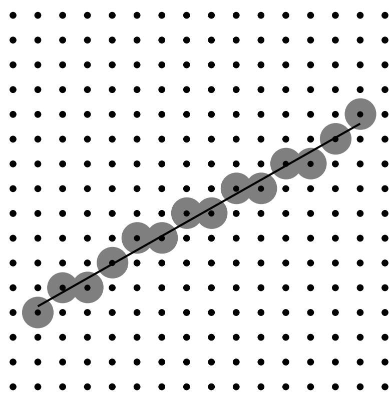


Scan Conversion: the generation of this approximation

Scan Converting 2D Line Segments

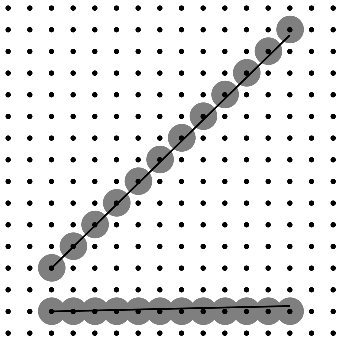
Given:
- Segment endpoints (integers x_1, y_1, x_2, y_2)
- Framebuffer access via *setPixel*(x, y)

Identify:
- Set of x, y for which to call *setPixel*(x, y)
- I.e., the set of pixels to "turn on"



Algorithm Design Choices

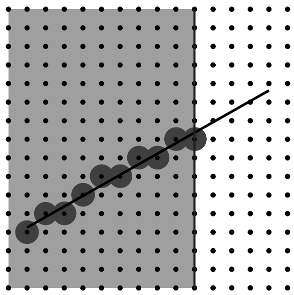
For $m \equiv dy/dx$, assume: $0 < m < 1$ (Why?)
Exactly one pixel per column (Why?)
  less $\Rightarrow$ disconnected
  more $\Rightarrow$ "too thick"
  "connectedness" with just 1 pixel per column
Note: brightness can vary with slope



How could we compensate for this?
  Answer: antialiasing

Naive algorithm

First attempt: simply compute y as a function of x
(Conceptually: move vertical scan line from x_1 to x_2)



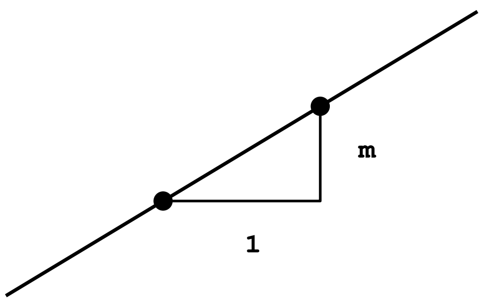
Express y as function of x: How?

Round y (Why necessary?)

Call *setPixel* $(x, \mathrm{rnd}(y(x)))$

Efficiency

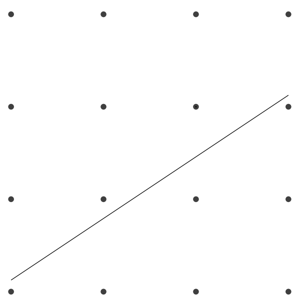
Computing line equation is expensive!
Observe: y `+=` m at each x step ($m = dy/dx$)
  First example of **coherence**!



Incremental algorithm:
  Start at (x_1, y_1)
  Thereafter, increment by slope
  (Hidden endpoint assumption?)

New Definition

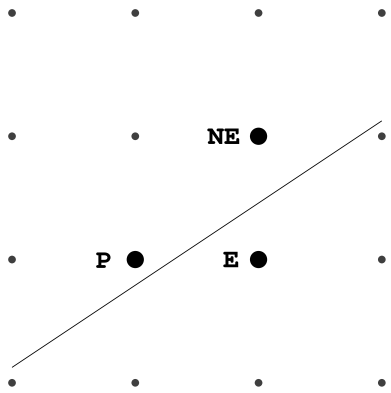
Select pixel **vertically** closest to segment



Justification: intuitive, efficient
  Also: pixel center always within 0.5 vertically
What other rules could we have used?

Bresenham Segment Algorithm

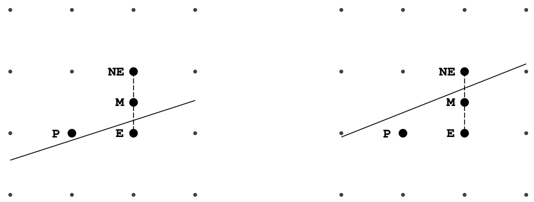
Another scan line algorithm
Same output as naive & incremental algorithms
Observation: after pixel P at (x_P, y_P)
next pixel either E or NE



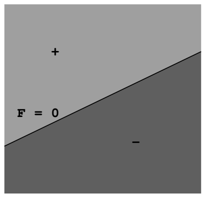
Why?

Bresenham Step

Which pixel to choose: E or NE?
Does segment pass above or below midpoint M?



Choose E if M lies "above" or "on" segment
Choose NE if M lies "below" segment
Use **implicit equation** for underlying line $\mathbf{L}$:
$F(x, y) = 0$, where $F(x, y) \equiv -mx + y + c$ (Why?)
F positive above $\mathbf{L}$, zero on $\mathbf{L}$, negative below $\mathbf{L}$



Define decision variable d by $d \equiv F(M)$
choose NE if d is negative and E otherwise.

Decision Variable

Simply compute $d = F(M)$ and check its sign:

$$\begin{aligned} d_{old} &= F(M) \\ &= F(x_P + 1, y_P + 0.5) \\ &= m(x_P + 1) + y_P + 0.5 + c \end{aligned}$$

We can do better! If E is chosen:

$$\begin{aligned} d_{new} &= F(M_{new}) \\ &= F((x_{P_{old}} + 1) + 1, y_{P_{old}} + 0.5) \\ &= m(x_{P_{old}} + 2) + y_{P_{old}} + 0.5 + c \\ &= m + m(x_{P_{old}} + 1) + y_{P_{old}} + 0.5 + c \\ &= d + m. \end{aligned}$$

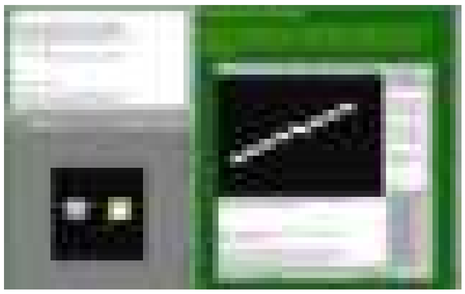
Whereas if NE is chosen:

$$\begin{aligned} d_{new} &= F(M_{new}) \\ &= F(x_{P_{new}} + 1, y_{P_{new}} + 0.5) \\ &= F((x_{P_{old}} + 1) + 1, (y_{P_{old}} + 1) + 0.5) \\ &= m(x_{P_{old}} + 2) + y_{P_{old}} + 1.5 + c \\ &= (m(x_{P_{old}} + 1) + m) + y_{P_{old}} + 0.5 + 1 + c \\ &= m(x_{P_{old}} + 1) + y_{P_{old}} + 0.5 + c + m + 1 \\ &= d + m - 1. \end{aligned}$$

Note: in both cases, increment is constant!
So: initialize x, y, d; loop over $x_1..x_2$, plot, update

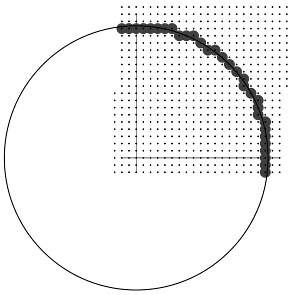
Bresenham Implementation

We've handled case in which $x_1 < x_2$, $m <= 1$
How to handle other seven octants?
Challenge: implement using integer arithmetic!
(Asst 1)
Hint: multiply increments through by $2dx$

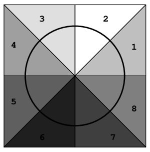


Circle Scan Conversion

Circle of radius R centered at origin



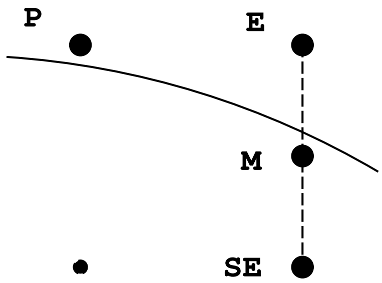
Need generate pixels for 2nd octant only



Why use the second octant?
Slope progresses from 0 to -1
  Analog of Bresenham Segment Algorithm

Bresenham Circle Algorithm

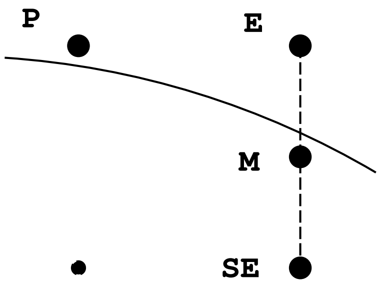
Design choice:
After pixel P, next either E or SE



Why? Analogous reasoning
  Call M midpoint between E and SE
  Choose E if M inside circle; SE otherwise

Bresenham Step

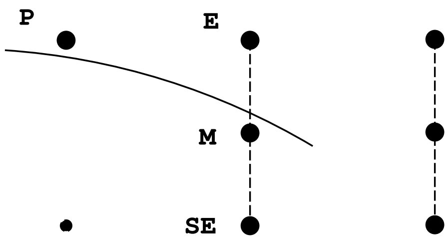
Implicit equation for circle:



  $F(x,y) = 0$ with $F(x,y) \equiv x^2 + y^2 - R^2$
  F positive outside, zero on, negative inside circle
  Define decision variable $d \equiv F(M)$
  Choose E if $d < 0$ and SE otherwise
  Note: M is at $(x_P + 1, y_P - 0.5)$

Bresenham Step E

Compute $d = F(M)$:



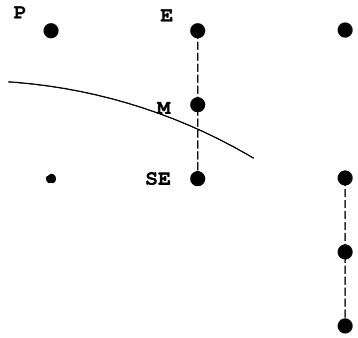
$$\begin{aligned} d &= F(M) \\ &= F(x_P + 1, y_P - 0.5) \\ &= (x_P + 1)^2 + (y_P - 0.5)^2 - R^2 \end{aligned}$$

If E is chosen, then at next step

$$\begin{aligned} d_{new} &= F(M_{new}) \\ &= F(x_{P_{new}} + 1, y_{P_{new}} - 0.5) \\ &= F((x_{P_{old}} + 1) + 1, y_{P_{old}} - 0.5) \\ &= (x_{P_{old}} + 2)^2 + (y_{P_{old}} - 0.5)^2 - R^2 \\ &= x_{P_{old}}^2 + 4x_{P_{old}} + 4 + (y_{P_{old}} - 0.5)^2 - R^2 \\ &= x_{P_{old}}^2 + 2x_{P_{old}} + 2x_{P_{old}} + 1 + 3 + (y_{P_{old}} - 0.5)^2 - R^2 \\ &= x_{P_{old}}^2 + 2x_{P_{old}} + 1 + (y_{P_{old}} - 0.5)^2 - R^2 + 2x_{P_{old}} + 3 \\ &= (x_{P_{old}} + 1)^2 + (y_{P_{old}} - 0.5)^2 - R^2 + 2x_{P_{old}} + 3 \\ &= d + (2x_{P_{old}} + 3). \end{aligned}$$

Thus $\Delta_E = 2x_{P_{old}} + 3$.

Bresenham Step SE



If SE is chosen, then at the next step

$$\begin{aligned}
d_{new} &= F(M_{new}) \\
&= F(x_{P_{new}}+1, y_{P_{new}}-0.5) \\
&= F((x_{P_{old}}+1)+1, (y_{P_{old}}-1)-0.5) \\
&= (x_{P_{old}}+2)^2 + (y_{P_{old}}-1.5)^2 - R^2 \\
&= x_{P_{old}}^2 + 4x_{P_{old}} + 4 + y_{P_{old}}^2 - 3y_{P_{old}} + 2.25 - R^2 \\
&= x_{P_{old}}^2 + 2x_{P_{old}} + 1 + y_{P_{old}}^2 - y_{P_{old}} + 0.25 - R^2 + 2x_{P_{old}} - 2y_{P_{old}} + 5 \\
&= (x_{P_{old}}+1)^2 + (y_{P_{old}}-0.5)^2 - R^2 + 2x_{P_{old}} - 2y_{P_{old}} + 5 \\
&= d_{old} + (2x_{P_{old}} - 2y_{P_{old}} + 5).
\end{aligned}$$

Thus $\Delta_{SE} = 2x_{P_{old}} - 2y_{P_{old}} + 5$.

Bresenham Optimization

Note: Δ's are integers; use integer arithmetic
Problem: neither Δ is a constant.
How do we do avoid recomputation at each step?
Coherence!
If we choose E:

$$\Delta_{E_{old}} = 2x_{P_{old}} + 3.$$

$$\begin{aligned}
\Delta_{E_{new}} &= 2x_{P_{new}} + 3 \\
&= 2(x_{P_{old}}+1) + 3 \\
&= \Delta_{E_{old}} + 2.
\end{aligned}$$

$$\Delta_{SE_{old}} = 2x_{P_{old}} - 2y_{P_{old}} + 5.$$

$$\begin{aligned}
\Delta_{SE_{new}} &= 2x_{P_{new}} - 2y_{P_{new}} + 5 \\
&= 2(x_{P_{old}}+1) - 2y_{P_{new}} + 5 \\
&= \Delta_{SE_{old}} + 2.
\end{aligned}$$

Bresenham Optimization

If we choose SE,

$$\Delta_{E_{old}} = 2x_{P_{old}} + 3.$$

$$\begin{aligned}
\Delta_{E_{new}} &= 2x_{P_{new}} + 3 \\
&= 2(x_{P_{old}}+1) + 3 \\
&= 2x_{P_{old}} + 3 + 2 \\
&= \Delta_{E_{old}} + 2.
\end{aligned}$$

$$\Delta_{SE_{old}} = 2x_{P_{old}} - 2y_{P_{old}} + 5.$$

$$\begin{aligned}
\Delta_{SE_{new}} &= 2x_{P_{new}} - 2y_{P_{new}} + 5 \\
&= 2(x_{P_{old}}+1) - 2(y_{P_{old}}-1) + 5 \\
&= 2x_{P_{old}} - 2y_{P_{old}} + 5 + 2 + 2 \\
&= \Delta_{SE_{old}} + 4.
\end{aligned}$$

Increments now *are* constant (d, Δ)
How do we apply them to the decision variable d?
  Note boundary conditions
  Technique of "successive differencing"
Why must both Δ's be updated at every step?

Bresenham Generalizations

Circular arcs
Ellipses
General conics
Filled ellipses

Registration and Assignment 1

Course registration
- Online signup, survey, picture, demos
  - Due 5pm next Friday, 12 September
  - **Required** for course registration

Assignment 1
- Segment scan conversion