Important Graphics Output Primitives

- in 2D
  - points, lines
  - polygons, circles, ellipses & other curves (also filled)
  - pixel array operations
  - characters
- in 3D
  - triangles & other polygons
  - free form surfaces
- + commands for properties: color, texture, ...

### Points and Lines

- point plotting
  - instruction in display list (random scan)
  - entry in frame buffer (raster scan)
- line drawing
  - instruction in display list (random scan)
  - intermediate discrete pixel positions calculated (raster scan)
  - "jaggies", aliasing

### Lines: Staircase Effect

Stairstep effect (jaggies) produced when a line is generated as a series of pixel positions.

### Line-Drawing Algorithms

#### line equation: \( y = mx + b \)

- line path between two points:
  \[
  m = \frac{y_{\text{end}} - y_0}{x_{\text{end}} - x_0} \\
  b = y_0 - mx_0
  \]

#### DDA Line-Drawing Algorithm

#### line equation: \( y = mx + b \)

- \( \delta y = m \cdot \delta x \) for \( |m| < 1 \)
- \( \delta x = \frac{\delta y}{m} \) for \( |m| > 1 \)

- DDA (digital differential analyzer)

  \[
  \text{for } \delta x = 1, \ |m| < 1 : \ y_{k+1} = y_k + m
  \]
**DDA – Algorithm Principle**

\[
\begin{align*}
    dx &= x_{\text{End}} - x_0; \\
    dy &= y_{\text{End}} - y_0; \\
    m &= \frac{dy}{dx}; \\
    x &= x_0; \\
    y &= y_0; \\
    \text{setPixel} \left( \text{round}(x), \text{round}(y) \right); \\
    \text{for} \ (k = 0; \ k < dx; \ k++) \\
    &\quad \{ \ x += 1; \ y += m; \\
    &\quad \text{setPixel} \left( \text{round}(x), \text{round}(y) \right) \}
\end{align*}
\]

extension to other cases simple

---

**Bresenham’s Line Algorithm**

- faster than simple DDA
- incremental integer calculations
- adaptable to circles, other curves

\[
y = m(x_k + 1) + b
\]

Section of a display screen where a straight line segment is to be plotted, starting from the pixel at column 10 on scan line 11.

---

**Bresenham’s Line Algorithm (1/4)**

Section of the screen grid showing a pixel in column \( x_k \) on scan line \( y_k \) that is to be plotted along the path of a line segment with slope \( 0 < m < 1 \).

\[
y_k = mx_k + b
\]

\[
\begin{align*}
    d_{upper} &= y - y_k = \frac{m(x_k + 1) + b - y_k}{2}; \\
    d_{lower} &= (y_k + 1) - y = y_k + 1 - m(x_k + 1) - b; \\
    d_{lower} - d_{upper} &= 2m(x_k + 1) - 2y_k + 2b - 1
\end{align*}
\]

decision parameter:

\[
p_k = \Delta x (d_{lower} - d_{upper}) = 2\Delta y x_k - 2\Delta x y_k + c
\]

\( \rightarrow \) same sign as \( d_{lower} - d_{upper} \)
Bresenham’s Line Algorithm (3/4)

current decision value:

\[ p_k = \Delta x(d_{\text{lower}} - d_{\text{upper}}) = 2\Delta yx_k - 2\Delta xy_k + c \]

next decision value:

\[ p_{k+1} = 2\Delta yx_{k+1} - 2\Delta xy_{k+1} + c + 0 \]
\[ + p_k - 2\Delta yx_k + 2\Delta xy_k - c = p_k + 2\Delta y - 2\Delta x(y_{k+1} - y_k) \]

starting decision value:

\[ p_0 = 2\Delta y - \Delta x \]

Bresenham: Example

<table>
<thead>
<tr>
<th>k</th>
<th>( p_k )</th>
<th>((x_{k+1}, y_{k+1}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-4</td>
<td>(20,41)</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>(21,41)</td>
</tr>
<tr>
<td>2</td>
<td>-12</td>
<td>(22,42)</td>
</tr>
<tr>
<td>3</td>
<td>-6</td>
<td>(23,42)</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>(24,42)</td>
</tr>
<tr>
<td>5</td>
<td>-14</td>
<td>(25,43)</td>
</tr>
<tr>
<td>6</td>
<td>-8</td>
<td>(26,43)</td>
</tr>
<tr>
<td>7</td>
<td>-2</td>
<td>(27,43)</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>(28,44)</td>
</tr>
<tr>
<td>9</td>
<td>-10</td>
<td>(30,44)</td>
</tr>
</tbody>
</table>

Bresenham’s Line Algorithm (4/4)

1. store left line endpoint in \((x_0, y_0)\)
2. plot pixel \((x_0, y_0)\)
3. calculate constants \(\Delta x, \Delta y, 2\Delta y, 2\Delta x - 2\Delta x\), and obtain \(p_0 = 2\Delta y - \Delta x\)
4. At each \(x_k\) along the line, perform test:
   - if \(p_k < 0\) then plot pixel \((x_k+1, y_k)\); \(p_{k+1} = p_k + 2\Delta y\)
   - else plot pixel \((x_k+1, y_k+1)\); \(p_{k+1} = p_k + 2\Delta y - 2\Delta x\)
5. perform step 4 \((\Delta x - 1)\) times.

Important Graphics Output Primitives

- In 2D
  - points, lines
  - polygons, circles, ellipses & other curves (also filled)
  - pixel array operations
  - characters
- In 3D
  - triangles & other polygons
  - free form surfaces
- + commands for properties: color, texture, …

What is Inside a Polygon?

- Odd Even rule
- Nonzero Winding
- Number rule

Fill-Area Primitives

- for polygon area (solid-color, patterned)
  - scan-line polygon fill algorithm
    - intersection points located and sorted
    - consecutive pairs define interior span
    - attention with vertex intersections
    - exploit coherence (incremental calculations)
  - flood fill algorithm
Polygon Fill Areas

- polygon classifications
  - convex: no interior angle > 180°
  - concave: not convex
- splitting concave polygons
  - vector method
    - all vector cross products have the same sign
      ⇒ convex
  - rotational method
    - rotate polygon-edges onto x-axis, always same direction ⇒ convex

Important Graphics Output Primitives

- In 2D
  - points, lines
  - polygons, circles, ellipses & other curves (also filled)
  - pixel array operations
  - characters
- In 3D
  - triangles & other polygons
  - free form surfaces
  - + commands for properties: color, texture, …

Character Primitives

- font (typeface)
  - design style for (family of) characters
  - Courier, Times, Arial, …
    - serif (better readable),
    - sans serif (better legible)
- definition model
  - bitmap font (simple to define and display), needs more space (font cache)
  - outline font (more costly, less space, geometric transformations)

Example: Newspaper

Panorama
Nach 28 Jahren jetzt ein Führer
Neuer Direktor
Werkenschulheim I

Character Generation Examples

the letter B represented with an 8x8 bilevel bitmap pattern and with an outline shape defined with straight line and curve segments
Polygon Surfaces (1)

- set of surface polygons enclose object interior
  **= Boundary Representation** (*B-rep*)

Example of a triangle mesh representing a dolphin.

Polygon Surfaces (2)

- polygon tables (B-rep lists)
  - geometric and attribute tables
  - vertex, edge, polygon tables
  - consistency, completeness checks

Polygon Surfaces: Data Structure

- VERTEX TABLE
- EDGE TABLE
- POLYGON-SURFACE TABLE

Lists for B-Reps

- surface list
- edge list
- vertex list

Polygon Surfaces: Plane Equation

\[ Ax + By + Cz + D = 0 \]

- plane parameters \( A, B, C, D \)
- normal \((A, B, C)\)

example: \( x-1=0 \)

Front and Back Polygon Faces

- **back face** = polygon side that faces into the object interior
- **front face** = polygon side that faces outward
- **behind** a polygon plane = visible to the polygon back face
- **in front of** a polygon plane = visible to the polygon front face
Front and Back Polygon Faces

Ax + By + Cz + D = 0 for points on the surface
< 0 for points behind
> 0 for points in front

if (1) right-handed coordinate system
(2) polygon points ordered
   counterclockwise

V₁, V₂, V₃ counterclockwise ⇒ normal vector

N = (V₂ - V₁) x (V₃ - V₁)

Reminder: Product of Vectors

V₁ = \begin{pmatrix} a_1 \\ b_1 \\ c_1 \end{pmatrix} \quad V₂ = \begin{pmatrix} a_2 \\ b_2 \\ c_2 \end{pmatrix}

- **scalar product:**
  \( V₁ \cdot V₂ = a_1 a_2 + b_1 b_2 + c_1 c_2 \)

- **cross product (vector product):**
  \( V₁ \times V₂ = \begin{pmatrix} a_1 \\ b_1 \\ c_1 \end{pmatrix} \times \begin{pmatrix} a_2 \\ b_2 \\ c_2 \end{pmatrix} = \begin{pmatrix} a_1 c_2 - c_1 b_2 \\ a_2 b_1 - b_2 a_1 \\ a_1 b_2 - b_1 a_2 \end{pmatrix} \)