1. Area Fill

- Boundary/Flood Fill
- Scanline Polygon Fill
- Scanline Boundary Fill
- Pattern Fill

2. Transformations

1. Set up edge table from vertex list; determine range of scanlines spanning polygon (miny, maxy)
2. Preprocess edges with nonlocal max/min endpoints
3. Set up activation table (bin sort)
4. For each scanline spanned by polygon:
   - Add new active edges to AEL using activation table
   - Sort active edge list on x
   - Fill between alternate pairs of points (x,y) in order of sorted active edges
   - For each edge e in active edge list:
     - If (y != ymax[e]) Compute & store new x (x+=1/m)
     - Else, Delete edge e from the active edge list

Scanline Polygon Fill Algorithm Example

Scanline Fill Algorithms can be fast if sorting is done efficiently

Video of BALSA Scanline Poly Fill Algorithm Animator

Brown University ALgorithm Simulator and Animator
Mark Brown and Bob Sedgewick

Scanline Fill Algorithms can be fast if sorting is done efficiently
Demo of Scanline Polygon Fill Algorithm vs. Boundary Fill Algorithm

Polyfill Program
- Does:
  - Boundary Fill
  - Scanline Polygon Fill
  - Scanline Circle with a Pattern
  - Scanline Boundary Fill (Dino Demo)

Adapting Scanline Polygon Fill to other primitives
- Example: a circle or an ellipse
  - Use midpoint algorithm to obtain intersection points with the next scanline
  - Draw horizontal lines between intersection points
  - Only need to traverse part of the circle or ellipse

Scanline Circle Fill Algorithm

Modify midpoint circle algorithm for each step draw 4 horizontal lines

The Scanline Boundary Fill Algorithm (General)
Select a Seed Point (x, y)
Push (x, y) onto Stack
While Stack is not empty:
  Pop Stack (retrieve x, y)
  Fill current run y (iterate on x until borders are hit)
  To left: Push unfilled pixels before above/below border pixels --> new above/below seeds
  To right: Push unfilled pixels after above/below border pixels --> new above/below seeds

The Scanline Boundary Fill Algorithm for Convex Polygons
Select a Seed Point (x, y)
Push (x, y) onto Stack
While Stack is not empty:
  Pop Stack (retrieve x, y)
  Fill current run y (iterate on x until borders are hit)
  Push left-most unfilled, nonborder pixel above --> new "above" seed
  Push left-most unfilled, nonborder pixel below --> new "below" seed

Dino Demo of Scanline Boundary Fill Algorithm
**Pattern Filling**

- Represent fill pattern with a Pattern Matrix
- Replicate it across the area until covered by non-overlapping copies of the matrix
  - Called Tiling

**Pattern Filling--Pattern Matrix**

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

In general, pos in matrix: 
\(xpos = x \mod W, ypos = y \mod H\)

**Using the Pattern Matrix**

- Modify fill algorithm
- As \((x,y)\) pixel in area is examined:
  
  ```python
  if(pat_mat[x%W][y%H] == 1):
      SetPixel(x, y);
  ```

**A More Efficient Way**

Store pat_matrix as a 1-D array of bytes or words, e.g., \(W \times H\)

\(y \% H \) \(\rightarrow\) byte or word in pat_matrix

Shift a mask by \(x \% W\)

- e.g. 10000000 for \(8 \times 8\) pat_matrix
- \(\rightarrow\) position of bit in byte/word of pat_matrix
  
  “AND” byte/word with shifted mask
  
  if result \(!= 0\), Set the pixel

**Color Patterns**

- Pattern Matrix contains color values
- So read color value of pixel directly from the Pattern Matrix:
  
  ```python
  SetPixel(x, y, pat_mat[x%W][y%H])
  ```

**Moving the Filled Polygon**

- As done above, pattern doesn’t move with polygon
- Need to “anchor” pattern to polygon
- Fix a polygon vertex as “pattern reference point”, e.g., \((x_0, y_0)\)
  
  ```python
  if (pat_matrix[(x-x0)%W][(y-y0)%H]==1):
      SetPixel(x, y)
  ```

Now pattern moves with polygon
**Pattern Filling--Pattern Matrix**

- Pattern Matrix
- In general, posn in matrix: 
  - Xpos = X = row
  - Ypos = Y = column

**Geometric Transformations**

- Moving objects relative to a stationary coordinate system
- Common transformations:
  - Translation
  - Rotation
  - Scaling
- Implemented using vectors and matrices

**Quick Review of Matrix Algebra**

- Matrix--a rectangular array of numbers
- $a_{ij}$: element at row i and column j
- Dimension: $m \times n$
  - $m$ = number of rows
  - $n$ = number of columns

**A Matrix**

A $m \times n$ matrix

\[
A = \begin{bmatrix}
    a_{11} & a_{12} & \cdots & a_{1n} \\
    a_{21} & a_{22} & \cdots & a_{2n} \\
    \vdots  & \vdots  & \ddots & \vdots  \\
    a_{m1} & a_{m2} & \cdots & a_{mn}
\end{bmatrix}
\]

Degenerate case: $m = 1$ (a row vector)

- $V = [a_{11} \ a_{12} \ a_{13} \ \cdots \ a_{1n}]$ or:
- $V = \begin{bmatrix}
    a_{11} \\
    a_{12} \\
    \vdots  \\
    a_{1n}
\end{bmatrix}$

**Vectors and Scalars**

- Degenerate Case ($n=1$): a column vector--
  - $V = \begin{bmatrix}
      a_{11} \\
      a_{21} \\
      \vdots  \\
      a_{m1}
    \end{bmatrix}$
  - $P = [2D \ 3D]$ or $P = \begin{bmatrix}
      2D \\
      3D
    \end{bmatrix}$

- Transpose of a Matrix $A^T$

**Matrix Operations--Multiplication by a Scalar**

- $C = k \cdot A$
  - $c_{ij} = k \cdot a_{ij}$, $1 \leq i \leq m$, $1 \leq j \leq n$
- Example: multiplying position vector by a constant:
  - Multiplies each component by the constant
  - Gives a scaled position vector ($k$ times as long)
Example of Multiplying a Position Vector by a Scalar

Adding two Matrices
- Must have the same dimension
- \( C = A + B \)
- \( c_{ij} = a_{ij} + b_{ij}, \ 1 \leq i \leq m, \ 1 \leq j \leq n \)
- Example: adding two position vectors
  - Add the components
  - Gives a vector equal to the net displacement

Adding two Position Vectors: Result is the Net Displacement

Multiplying Two Matrices
- \( m \times n = (m \times p) \times (p \times n) \)
- \( C = A \times B \)
- \( c_{ij} = \sum a_{ik} b_{kj}, \ 1 \leq k \leq p \)
- In other words:
  - To get element in row i, column j
    - Multiply each element in row i by each corresponding element in column j
    - Add the partial products

Matrix Multiplication
An Example

Multiply a Vector by a Matrix
- \( V_1 = A \times V \)
- If V is a m-dimensional column vector, A must be an \( m \times m \) matrix
- \( v_{1i} = \sum a_{ik} v_k, \ 1 \leq k \leq m \)
  - So to get element i of product vector:
    - Multiply each row i matrix element by each corresponding element of the vector
    - Add the partial products
An Example
Multiplying a 2-D Vector by a Matrix

\[
\begin{bmatrix}
3 & 0 \\
1 & 4
\end{bmatrix}
\begin{bmatrix}
5 \\
2
\end{bmatrix} =
\begin{bmatrix}
15 \\
10
\end{bmatrix}
\]

Geometrical Transformations
- Alter or move objects on screen
- Affine Transformations:
  - Preserve straight lines
  - Transform points in the object
- Translation:
  - A Vector Sum
- Rotation and Scaling:
  - Matrix Multiplies

Translation: Moving Objects

Scaling: Sizing Objects

Scaling, continued

\[
P' = S \cdot P
\]
P, P' are 2D vectors, so S must be 2x2 matrix

Component equations:

\[
x' = sx \cdot x, \quad y' = sy \cdot y
\]

Rotation about Origin

\[
R = \begin{bmatrix}
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta
\end{bmatrix}
\]

\[
P' = R \cdot P
\]

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} = \begin{bmatrix}
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix}
\]

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} = \begin{bmatrix}
sx \\
sy
\end{bmatrix}
\]

Therefore: \( S = \begin{bmatrix}
sx & 0 \\
0 & sy
\end{bmatrix} \) (The scaling matrix)
Rotation: X Component

P' = R*P
R must be a 2x2 matrix
Component equations:

\[ x' = x \cos(\theta) - y \sin(\theta) \]
\[ y' = x \sin(\theta) + y \cos(\theta) \]

Rotation: Y Component

P' = R*P
R must be a 2x2 matrix
Component equations:

\[ x' = A \cdot x \]
\[ y' = B \cdot x + C \cdot y \]

Rotation: Result

For example, lines
1. Transform every point & plot (too slow)
2. Transform endpoints, draw the line
   • Since these transformations are affine, result is the transformed line

Transforming Objects

Composite Transformations

- Successive transformations
- e.g., scale then rotate an n-point object:
  1. Scale points: \( P' = S*P \) (n matrix multiplies)
  2. Rotate pts: \( P'' = R*P' \) (n matrix multiplies)
  But:
      \[ P'' = R'(SP) \] & matrix multiplication is associative
      \[ P'' = (R'S)*P = M_{comp}P \]
  So Compute \( M_{comp} = R'S \) (1 matrix mult.)
  \[ P'' = M_{comp}P \] (n matrix multiplies)
  \( n+1 \) multiplies vs. \( 2n \) multiplies

Another example: Rotate in place center at (a,b)
1. Translate to origin: \( T(-a,-b) \)
2. Rotate: \( R(\theta) \)
3. Translate back: \( T(a,b) \)
Rotation in place:
1. $P' = P + T_1$
2. $P'' = R^*P' = R^*(P + T_1)$
3. $P''' = P'' + T_3 = R^*(P + T_1) + T_3$

Can't be put into single matrix mult. form:
i.e., $P''' \neq T_{comp} \times P$

But we want to be able to do that!!

Problem is: translation--vector add
rotation/scaling--matrix multiply

---

**Homogeneous Coordinates**

- Redefine transformations so each is a matrix multiply
- Express each 2-D Cartesian point as a triple:
  - A 3-D vector in a “homogeneous” coordinate system

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} xh \\ yh \\ 1 \end{bmatrix} \text{ where we define:}$$

$$xh = w*x, \quad yh = w*y$$

---

**Homogeneous Translations**

$$P' = P + T \quad \text{(Cartesian 2-D coordinates)}$$

$$P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$

$$P' = T \times P$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

What matrix is $T$?

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$

$$T = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix}$$

So:

$$T = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

---

**Homogeneous Scaling (wrt origin)**

$$P' = S \times P$$

Component Equations:
$$x' = s1x + s2y + s3$$
$$y' = s1x + s2y + s3$$
$$1 = s1x + s2y + s3$$

Comparing with component eqns:
$$s1 = 1, \quad s2 = 1, \quad s3 = 0$$

So:
$$S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

---

**Homogeneous Rotation (about origin)**

$$P' = R \times P$$

Component Equations:
$$x' = \cos(o) \times x - \sin(o) \times y$$
$$y' = \sin(o) \times x + \cos(o) \times y$$

$$R = \begin{bmatrix} \cos(o) & -\sin(o) \\ \sin(o) & \cos(o) \end{bmatrix}$$

Comparing with component eqns:
$$\cos(o) \times x + \sin(o) \times y$$

So:
$$R = \begin{bmatrix} \cos(o) & -\sin(o) \\ \sin(o) & \cos(o) \end{bmatrix}$$
Composite Transformations with Homogeneous Coordinates

- All transformations implemented as homogeneous matrix multiplies
- Assume transformations $T_1$, then $T_2$, then $T_3$:
  - Homogeneous matrices are $T_1$, $T_2$, $T_3$
  - $P' = T_1 P$
  - $P'' = T_2' P' = (T_2' T_1') P$
  - $P''' = T_3' P'' = (T_3' T_2' T_1') P$
  - Composite transformation: $T = T_3' T_2' T_1$
  - Compute $T$ just once!

Example

Rotate line from $(5,5)$ to $(10,5)$ by $90$ about $(5,5)$

$T_1 = T(-5,-5)$, $T_2 = R(90)$, $T_3 = T(5,5)$

$T = T_3' T_2' T_1$

$\begin{bmatrix}
1 & 0 & 5 \\
0 & 1 & 5 \\
0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
\cos 90 & -\sin 90 & 0 \\
\sin 90 & \cos 90 & 0 \\
0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
1 & 0 & -5 \\
0 & 1 & -5 \\
0 & 0 & 1 \\
\end{bmatrix}$

$T = \begin{bmatrix}1 & 0 & 5 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix}$

Example, continued

$P_1' = T P_1$

$\begin{bmatrix}
0 & -1 & 10 \\
1 & 0 & 0 \\
0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}5 \\ 5 \\ 1 \end{bmatrix}$

$P_1' = \begin{bmatrix}5 \\ 5 \\ 1 \end{bmatrix}$

$\begin{bmatrix}
0 & -1 & 10 \\
1 & 0 & 0 \\
0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}10 \\ 5 \\ 1 \end{bmatrix}$

$P_2' = \begin{bmatrix}10 \\ 5 \\ 1 \end{bmatrix}$

i.e., $P_1' = (5,5)$, $P_2' = (5,10)$