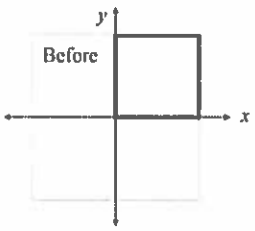
Solutions

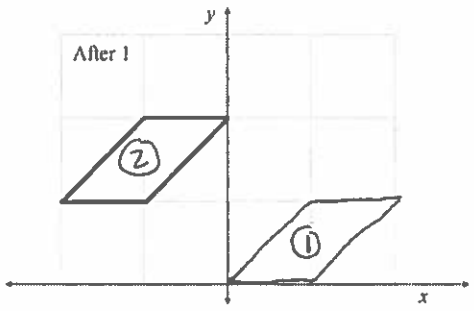
Midterm Practice Problems

1. *Transformations:* the images below show a square before two sets transformations. "After 1" is a composition of two transformations (denote AB in matrix multiplication form). "After 2" is a composition of the same two transformations, but in reverse order (BA in matrix multiplication). Find A and B that satisfy these conditions, and perform matrix multiplication on the "Before" square to demonstrate your answer.

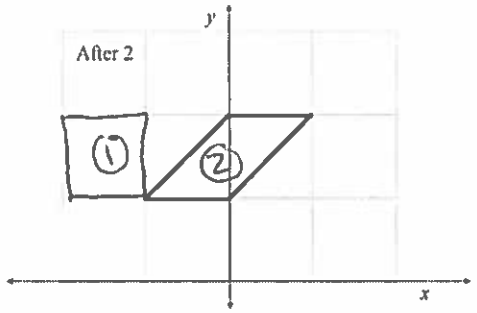
$$S = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Square





$T \cdot S_x \cdot S$



$S_x \cdot T \cdot S$

Let:

$$T = \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \quad \& \quad S_x = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

After 1

$$\underbrace{\begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}}_{T}\underbrace{\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{S_x}\underbrace{\begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}}_{S} = \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & 1 & 2 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -2 & -1 & 0 & -1 \\ 1 & 1 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

After 2

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} -2 & -1 & -1 & -2 \\ 1 & 1 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 & 0 \\ 1 & 1 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

2. *Recursion:* write a recursive function that will return $n!$ for any integer $n \geq 0$. In general, $n! = n \cdot (n-1) \cdot (n-2) \cdots 3 \cdot 2 \cdot 1$, so this starts $1! = 1$, $2! = 2$, $3! = 6$, etc, and $0!$ is defined to be 1. In short, implement `function factorial(n) {...}`

```
function factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)
```

Solutions

3. *Lines:* In HW1 we saw how to implement an algorithm for drawing a line between two points: $p_0 = (x_0, y_0)$ and $p_1 = (x_1, y_1)$, which relied on the slope of the line. For this question, write pseudocode for a line algorithm that would achieve the same goal, but this time using a *parametric Bézier* approach (i.e. implement: `function line(p0,p1) {...}`). Your algorithm should:

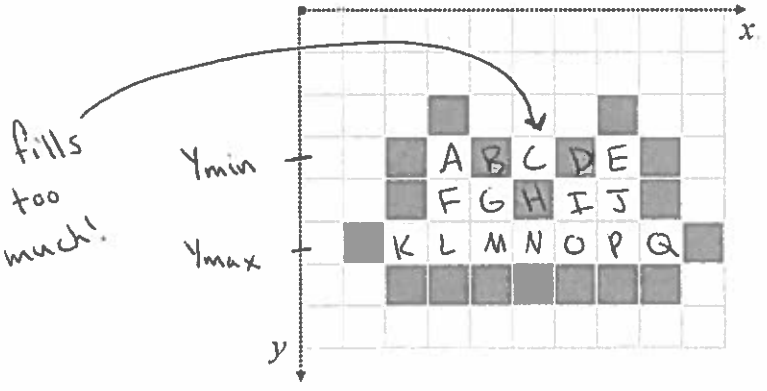
   (a) Make the line look "connected" (no gaps). Diagonal pixels are considered connected.

   (b) No pixel should be colored more than once.

```
function line(p0, p1) {
    numPoints = max{ abs(x1-x0), abs(y1-y2) }

    for i in range(numPoints) {
        t = i/numPoints
        x = (1-t)·x0 + t·x1
        y = (1-t)·y0 + t·y1
        fillRect(x, y, 1, 1)
    }
}
```

4. *Sweep fill:* as presented in class, what order will the pixels below be filled? Use "A" for the first filled pixel, "B" for the second, etc. The outer loop over the y values will start at y_{min} and go to y_{max}. For each y value, the x's will go loop from x_{min} to x_{max}. Assume y increases going down.



$y = 3$: $x_{min} = 3$, $x_{max} = 7$

$y = 4$: $x_{min} = 3$, $x_{max} = 7$

$y = 5$: $x_{min} = 2$, $x_{max} = 8$