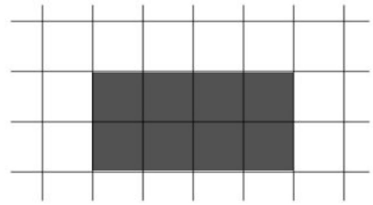
Transformations in 2D
Short version

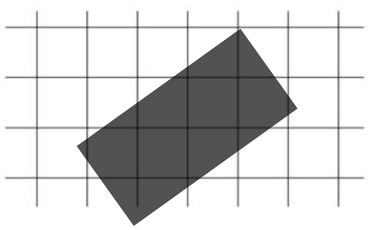
We will discuss transformation in 3D, and with full details, later in the course
(will need Matrix Multiplication and Homogenous coordinates)

About hw1
Aliasing and Anti-Aliasing



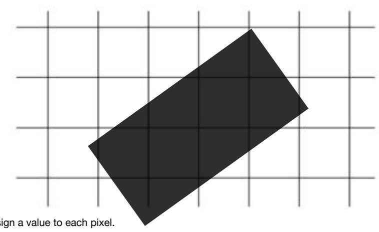
- This about an image where each pixels is fully black or fully white

What if we rotate the rectangle



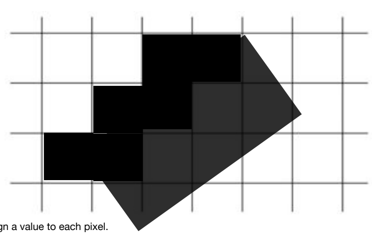
- Some pixels are **partially** covered by the rectangle. Show they be rendered as black, white, or some shade of grey ?

What if we rotate the rectangle



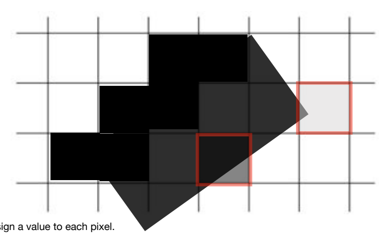
- IWe still need to assign a value to each pixel.
- If we draw each **partially covered** pixel as black, we will obtain a very pixelated shape. This is an example of **aliasing.**
- A possible solution is to render some pixels as gray. For example, based on the portion of its area which is covered. This technique is call antialiasing. Essentially, the color of a pixel might be determined using input from several neighboring pixels.
- We will study much much more about it. Do not worry about it in hw1.
- In hw1, each rendered pixel has the (rgb) value of one (single) input pixel. No averaging or mixing.

What if we rotate the rectangle



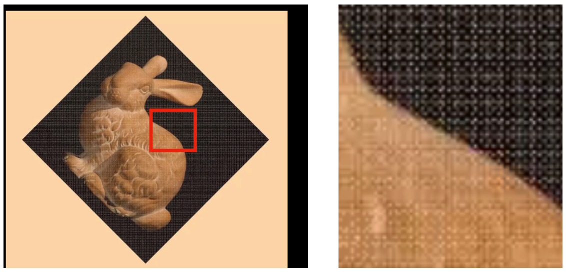
- IWe still need to assign a value to each pixel.
- If we draw each **partially covered** pixel as black, we will obtain a very pixelated shape. This is an example of **aliasing.**
- A possible solution is to render some pixels as gray. For example, based on the portion of its area which is covered. This technique is call antialiasing. Essentially, the color of a pixel might be determined using input from several neighboring pixels.
- We will study much much more about it. Do not worry about it in hw1.
- In hw1, each rendered pixel has the (rgb) value of one (single) input pixel. No averaging or mixing.

What if we rotate the rectangle

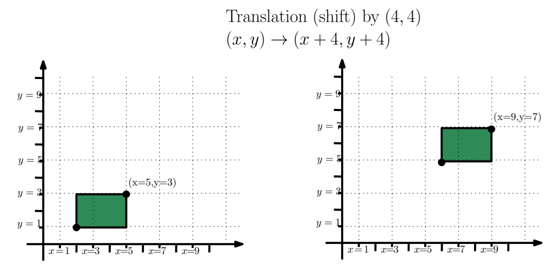


- IWe still need to assign a value to each pixel.
- If we draw each **partially covered** pixel as black, we will obtain a very pixelated shape. This is an example of **aliasing.**
- A possible solution is to render some pixels as gray. For example, based on the portion of its area which is covered. This technique is call antialiasing. Essentially, the color of a pixel might be determined using input from several neighboring pixels.
- We will study much much more about it. Do not worry about it in hw1.
- In hw1, each rendered pixel has the (rgb) value of one (single) input pixel. No averaging or mixing.

Something to be careful about with hw1



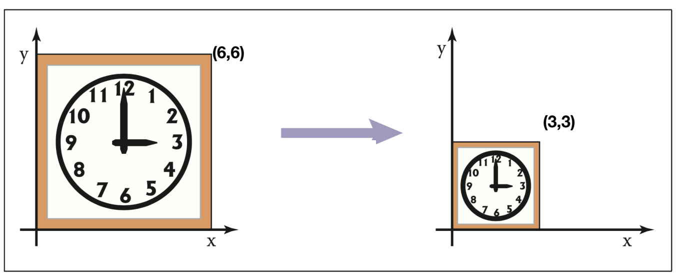
Translations (shift) by(α, β)



- Adding a constant α to the x-coordinate of every point
- Adding a constant β to the y-coordinate of every point
- $(x, y) \rightarrow (x + \alpha, y + \beta)$

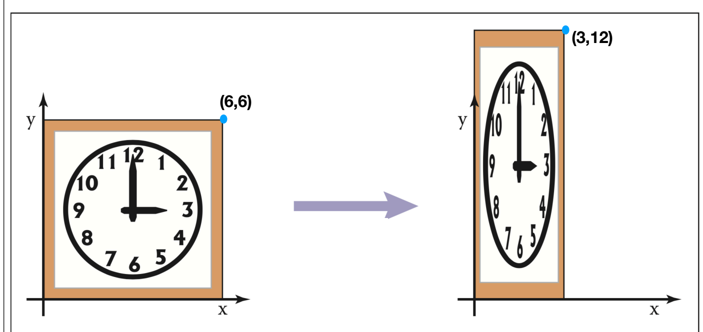
Scaling

- We can use two constants *(s_x,s_y)* for the x-axis and the y-axis. Then we shift each point *(x,y)* into the point $(s_x \cdot x, \;\; s_y \cdot y)$
- $(x, y) \rightarrow (s_x \cdot x, \;\; s_y \cdot y)$
- Example $(x, y) \rightarrow (x/2, \;\; y/2)$

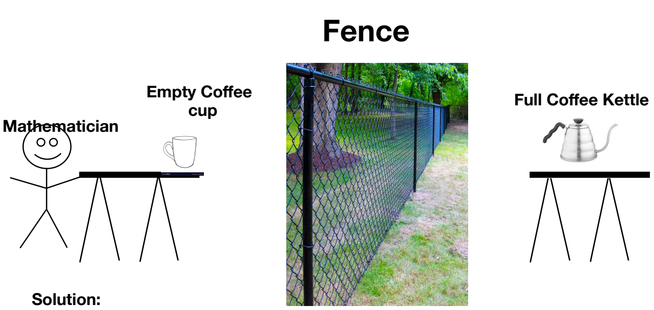


Scaling

- Example: $(x, y) \rightarrow (0.5x, \;\; 2y)$

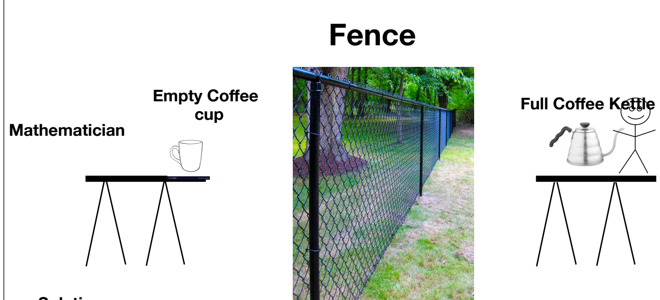


The mathematician and coffee cup non-funny joke Part 1



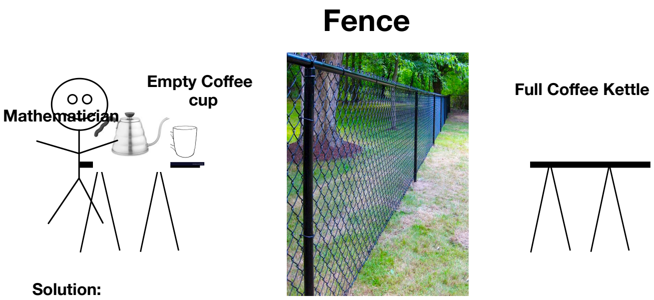
Solution:
1. **Walk around the fence,**
2. **fetch coffee kettle,**
3. **walk back pure coffee,**
4. **drink**

The mathematician and coffee cup non-funny joke Part 1



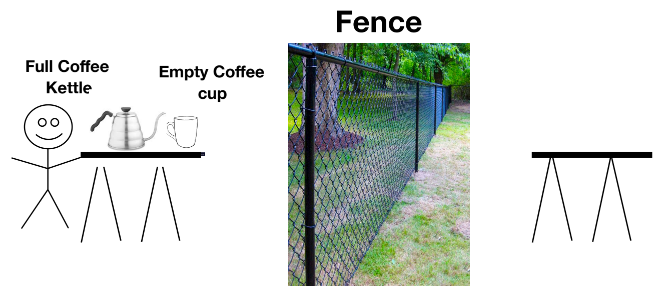
Solution:
1. **Walk around the fence,**
2. **fetch coffee kettle,**
3. **walk back pure coffee,**
4. **drink**

The mathematician and coffee cup non-funny joke
Part 1



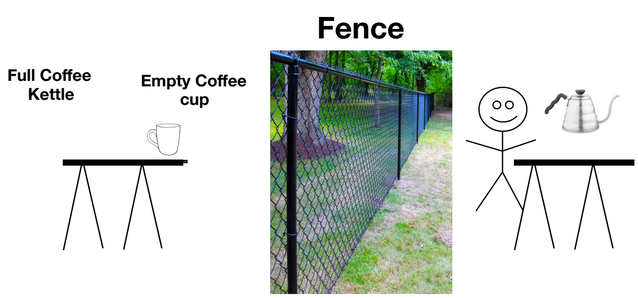
Solution:
1. **Walk around the fence,**
2. **fetch coffee kettle,**
3. **walk back pure coffee,**
4. **drink**

The mathematician and coffee cup non-funny joke
Part 2



Solution:
1. **Bring the coffee Kettle to the other table, and walk to the left table**
2. **Apply the solution from the previous slide**

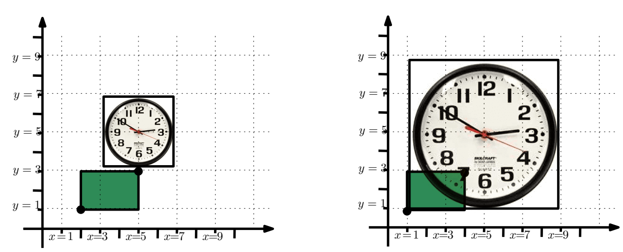
The mathematician and coffee cup non-funny joke
Part 2



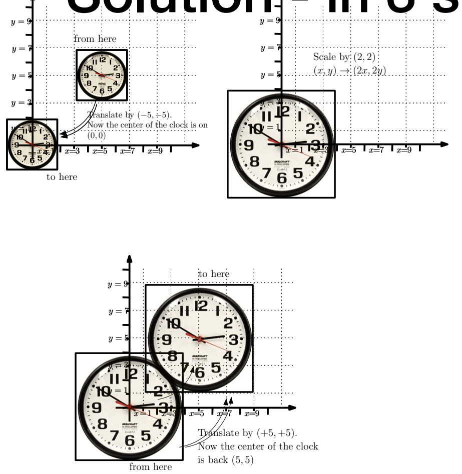
Solution:
1. **Bring the coffee Kettle to the other table, and walk to the left table**
2. **Apply the solution from the previous slide**

Resize the clock, without changing its center

Problem: scale the clock, but without changing its center and without effecting the green rectangle

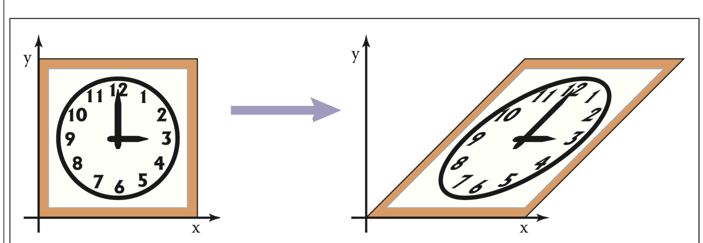


Solution - in 3 steps



Shearing

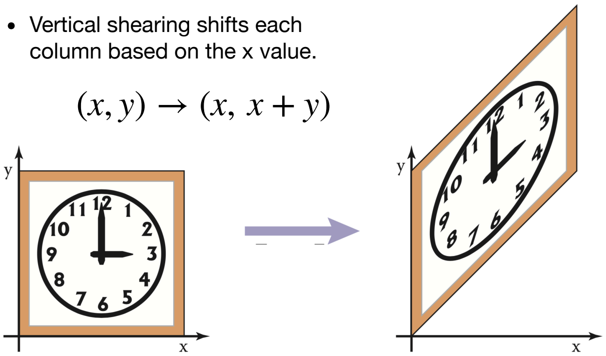
- If we move each point (x,y) into the point $(x, y) \rightarrow (x + y,\ y)$



Shearing

- Vertical shearing shifts each column based on the x value.

$$(x, y) \rightarrow (x,\ x + y)$$

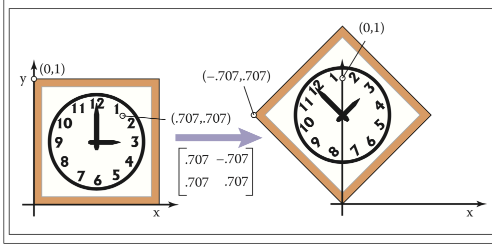


Rotation

- Rotate counterclockwise by an angle θ about the origin.

$$(x, y) \rightarrow (x\cos\theta - y\sin\theta,\ \ x\sin\theta + y\cos\theta)$$

New *x* **New *y***

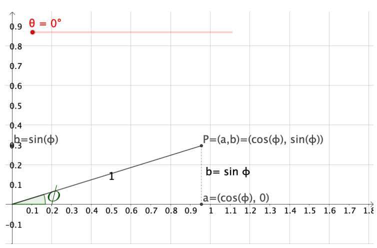


Assume we rotate *p* by an angle θ CCW

Starting from a point $P = (a, b)$, where will this point find itself after rotation by θ in the CounterClockwise direction ?
Let $p' = (x', y')$ denote the new location of this point. Lets compute this location:
For simplicity, assume $a^2 + b^2 = 1$

$$x' = \cos(\phi + \theta) = \underbrace{\cos(\phi)}_{=a}\cos(\theta) - \underbrace{\sin(\phi)}_{=b}\sin(\theta) = a\cos(\theta) - b\sin(\theta)$$

$$y' = \sin(\phi + \theta) = \underbrace{\sin(\phi)}_{=b}\cos(\theta) + \underbrace{\cos(\phi)}_{=a}\sin(\theta)\ a\sin(\theta) + y\sin(\theta)$$

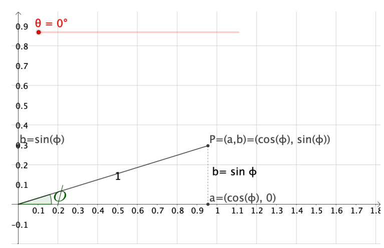


Assume we rotate *p* by an angle θ CCW

Starting from a point $P = (a, b)$, where will this point find itself after rotation by θ in the CounterClockwise direction ?
Let $p' = (x', y')$ denote the new location of this point. Lets compute this location:
For simplicity, assume $a^2 + b^2 = 1$

$$x' = \cos(\phi + \theta) = \underbrace{\cos(\phi)}_{=a}\cos(\theta) - \underbrace{\sin(\phi)}_{=b}\sin(\theta) = a\cos(\theta) - b\sin(\theta)$$

$$y' = \sin(\phi + \theta) = \underbrace{\sin(\phi)}_{=b}\cos(\theta) + \underbrace{\cos(\phi)}_{=a}\sin(\theta)\ a\sin(\theta) + y\sin(\theta)$$

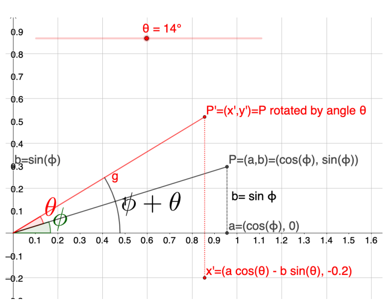


Assume we rotate *p* by an angle θ CCW

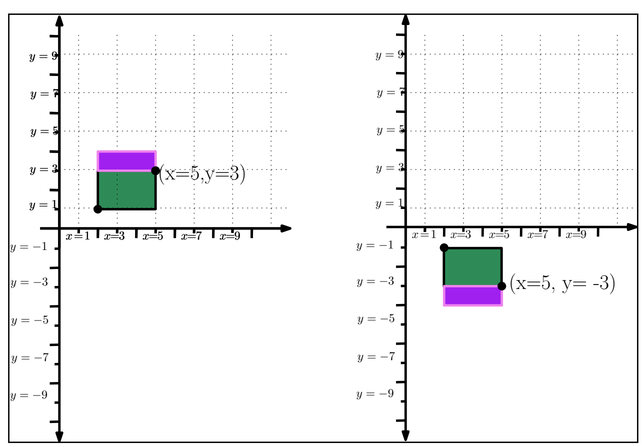
Starting from a point $P = (a, b)$, where will this point find itself after rotation by θ in the CounterClockwise direction ?
Let $p' = (x', y')$ denote the new location of this point. Lets compute this location:
For simplicity, assume $a^2 + b^2 = 1$

$$x' = \cos(\phi + \theta) = \underbrace{\cos(\phi)}_{=a}\cos(\theta) - \underbrace{\sin(\phi)}_{=b}\sin(\theta) = a\cos(\theta) - b\sin(\theta)$$

$$y' = \sin(\phi + \theta) = \underbrace{\sin(\phi)}_{=b}\cos(\theta) + \underbrace{\cos(\phi)}_{=a}\sin(\theta)\ a\sin(\theta) + y\sin(\theta)$$



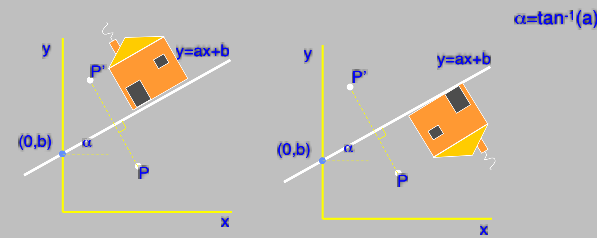
Reflection on the x-axes: $(x, y) \rightarrow (x,\ -y)$



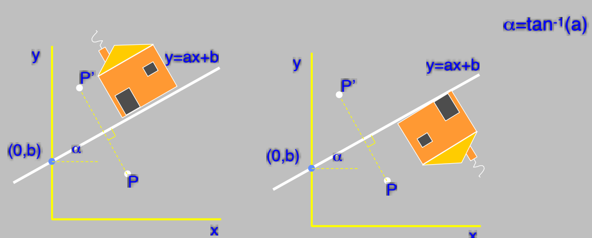
Transformation Composition

- What operation rotates by θ around $P = (p_x, p_y)$?
  - Translate P to origin
  - Rotate around origin by θ
  - Translate back

Arbitrary Reflection - promo
We will get back to it later in the semester

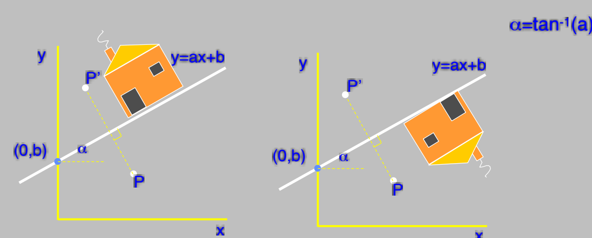


Arbitrary Reflection - promo
We will get back to it later in the semester



1. Compute b.
2. Shift by (0,-b)
3. Rotate by $-\alpha$ CCW
4. Reflect through x
5. Rotate by α
6. Shift by (0,b)

Arbitrary Reflection - promo
We will get back to it later in the semester



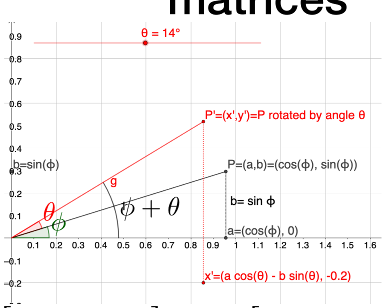
1. Compute b.
2. Shift by (0,-b)
3. Rotate by $-\alpha$ CCW
4. Reflect through x
5. Rotate by α
6. Shift by (0,b)

Very scarrrrry….

Unless we represent transformation by matrices

And then it is trivial

Expressing rotations with matrices



- $$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a\cos\theta - b\sin\theta \\ a\sin\theta + b\cos\theta \end{bmatrix} = \begin{bmatrix} a' \\ y' \end{bmatrix}$$

What about other operaions

- Scaling ?
- Reflection by the x-axis ?
- Sheering ?
- Translation is problematic ?

Homogeneous coordinates

- We represent a point $p = (x, y)$ using 3 numbers $p = (x, y, w)_h$
- What are the coordinates of this point in Euclidean Cartesian representation ? $p = (x/w,\ y/w)$
- So $(1,2,1) = (2,4,2) = (0.5,1,0.5)$