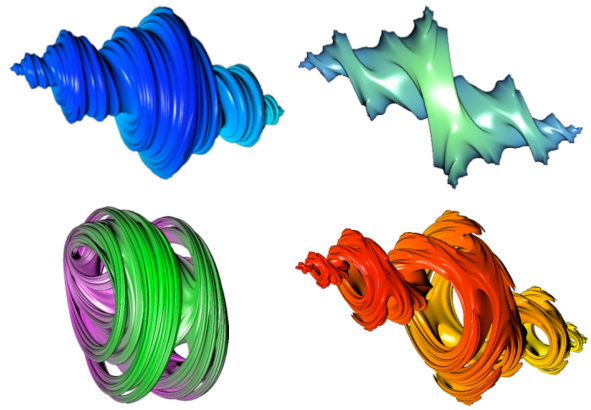
Geometry

CS148, Summer 2010

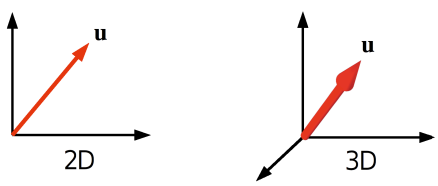
Siddhartha Chaudhuri



Ödegaard and Wennergren, 2D projections of 4D "Julia sets"

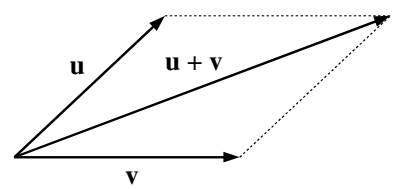
Vectors: the "Physical" View

- Object $\mathbf{u}$ with direction and length/magnitude
- $-\mathbf{u}$ is vector in opposite direction with same length
- $\| \mathbf{u} \|$ denotes length of $\mathbf{u}$
  - *Unit vector* has length 1, is typically written with hat: $\hat{\mathbf{u}}$
- Multiplying a vector by a scalar changes its length



Addition of Vectors

- "Parallelogram Rule"

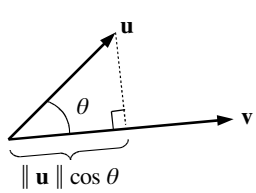


- $\mathbf{u} + (-\mathbf{v}) \equiv \mathbf{u} - \mathbf{v}$

Dot Product

- If θ is the angle between $\mathbf{u}$ and $\mathbf{v}$, then

$$\mathbf{u}\cdot\mathbf{v} = \| \mathbf{u} \| \; \| \mathbf{v} \| \cos\theta$$



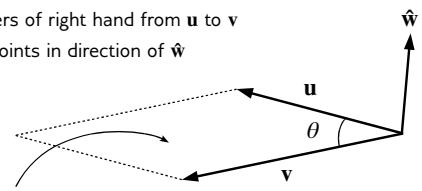
- This is also the length of the orthogonal projection of $\mathbf{u}$ onto $\mathbf{v}$, times the length of $\mathbf{v}$
- **Note:** $\mathbf{u}\cdot\mathbf{u} = \| \mathbf{u} \|^2$

Orthogonal and Orthonormal Vectors

- *Orthogonal* vectors are at right angles to each other
  - $\mathbf{u}\cdot\mathbf{v} = 0$
- *Orthonormal* vectors are at right angles to each other *and* each has unit length
- A set $\{\mathbf{u}_i\}$ of orthonormal vectors has
  - $\mathbf{u}_i\cdot\mathbf{u}_j = 1$ if $i = j$
  - $\mathbf{u}_i\cdot\mathbf{u}_j = 0$ if $i \neq j$

Cross Product (only 3D)

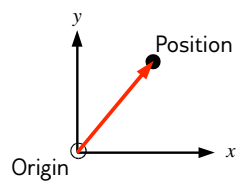
- If θ is the angle between $\mathbf{u}$ and $\mathbf{v}$, then $\mathbf{u} \times \mathbf{v} = (\| \mathbf{u} \| \ \| \mathbf{v} \| \sin \theta)\ \hat{\mathbf{w}}$
  - $\hat{\mathbf{w}}$ is a unit vector orthogonal to both $\mathbf{u}$ and $\mathbf{v}$
  - "Right-Hand Rule":
    - Curl fingers of right hand from $\mathbf{u}$ to $\mathbf{v}$
    - Thumb points in direction of $\hat{\mathbf{w}}$



Area of parallelogram = $\| \mathbf{u} \times \mathbf{v} \|$

Position Vector

- Identifies point in space
  - Interpreted as: tip of vector "arrow" when the other end is placed at a fixed point called the *origin* of the space

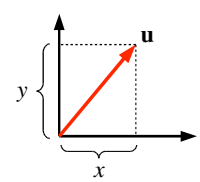


Vectors: the "Physical" View

- Directions and positions are ***different***!
- "Legal" Operations:
  - Direction = Scalar * Direction
  - Direction = Direction + Direction
  - Position = Position + Direction
  - Direction = Position – Position
- "Illegal" Operations:
  - Position = Scalar * Position
  - Position = Position + Position
  - Direction = Position + Direction
  - Position = Position - Position

Cartesian/Euclidean n-space $\mathbb{R}^n$

- Vectors represented by real n-tuples of *coordinates* $(u_1, u_2, \ldots, u_n)$
  - 2D: (x, y)
  - 3D: (x, y, z)
  - Represents extent along orthogonal *coordinate axes*
    - Right-handed system: Curl right hand fingers from x axis to y axis, thumb points along z axis



- Length/magnitude: $\| \mathbf{u} \| = (u_1^2 + u_2^2 + \ldots + u_n^2)^{1/2}$
  - From Pythagoras' Theorem

Cartesian/Euclidean n-space $\mathbb{R}^n$

- Let $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ and $\mathbf{v} = (v_1, v_2, \ldots, v_n)$
- Addition of vectors:

$$\mathbf{u} + \mathbf{v} \equiv (u_1 + v_1, u_2 + v_2, \ldots, u_n + v_n)$$

- Multiplication of vector by scalar:

$$s\mathbf{u} \equiv (su_1, su_2, \ldots, su_n)$$

Cartesian/Euclidean n-space $\mathbb{R}^n$

- Dot Product:

$$\mathbf{u} \cdot \mathbf{v} \equiv (u_1 v_1 + u_2 v_2 + \ldots + u_n v_n)$$

- Cross Product (remember, 3D only!):

$$\mathbf{u} \times \mathbf{v} \equiv (u_2 v_3 - u_3 v_2, u_3 v_1 - u_1 v_3, u_1 v_2 - u_2 v_1)$$

Another Way to Remember the Cross Product in $\mathbb{R}^3$

- Let $\mathbf{u} = (x_1, y_1, z_1)$ and $\mathbf{v} = (x_2, y_2, z_2)$
- $\mathbf{u} \times \mathbf{v} \equiv (y_1 z_2 - z_1 y_2, z_1 x_2 - x_1 z_2, \underbrace{x_1 y_2 - y_1 x_2})$

If $\mathbf{u}$ and $\mathbf{v}$ are 2D instead, this is the area of the parallelogram between them

Vectors: the "Mathematical" View

- *Vector space*: Set of objects (*vectors*) closed under
  - Addition of two vectors
  - Multiplication of a vector by a scalar (from field F)
- Necessary properties of vector spaces
  - Commutative addition: $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$
  - Associative addition: $(\mathbf{x} + \mathbf{y}) + \mathbf{z} = \mathbf{x} + (\mathbf{y} + \mathbf{z})$
  - Existence of additive identity $\mathbf{0}$: $\mathbf{x} + \mathbf{0} = \mathbf{0} + \mathbf{x}$
    - Can show: $0.\mathbf{x} = \mathbf{0}$ (prove!)
  - Existence of additive inverse $-\mathbf{x}$: $\mathbf{x} + (-\mathbf{x}) = \mathbf{0}$
    - Can show: $-\mathbf{x} = -1.\mathbf{x}$ (prove!)

Vectors: the "Mathematical" View

- Properties of vector spaces (contd.)
  - Associative scalar multiplication: $r(s\mathbf{x}) = (rs)\mathbf{x}$
  - Distributive scalar sum: $(r + s)\mathbf{x} = r\mathbf{x} + s\mathbf{x}$
  - Distributive scalar multiplication: $r(\mathbf{x} + \mathbf{y}) = r\mathbf{x} + r\mathbf{y}$
  - Scalar multiplication identity: $1.\mathbf{x} = \mathbf{x}$
    - **Note:** This says that the identity for multiplication of two scalars is also the identity for multiplication of a vector by a scalar

Basis and Dimension

- *Linear combination* of vectors $\{\mathbf{u}_i\}$: $\sum c_i \mathbf{u}_i$
  - $\{c_i\}$: Scalar coefficients
- $\{\mathbf{u}_i\}$ is *linearly independent* if no $\mathbf{u}_i$ can be expressed as a linear combination of the others
- *Span* of $\{\mathbf{u}_i\}$: Set of all linear combinations
- *Basis* of vector space: Set of linearly independent vectors whose span is the entire space
- *Dimension* of vector space: Cardinality of basis
  - **Note:** All bases have same cardinality

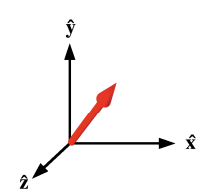
Basis

- Let a vector space have basis $\mathrm{B} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n]$
- Any vector $\mathbf{u}$ can be written as $\sum_{i=1\ldots n} u_i \mathbf{u}_i$
  - Or compactly, $(u_1, u_2, \ldots, u_n)$
  - These are the coordinates of $\mathbf{u}$ in basis B
- **Note:**
  - We've generalized our original notion of coordinates: they're now relative to the selected basis (axes)
  - A point has different coordinates in different bases

Back to $\mathbb{R}^n$

- Has a basis $[\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, \ldots, \hat{\mathbf{u}}_n]$ along coordinate axes

$$\hat{\mathbf{u}}_1 = (1, 0, \ldots, 0)$$
$$\hat{\mathbf{u}}_2 = (0, 1, \ldots, 0)$$
$$\vdots$$
$$\hat{\mathbf{u}}_n = (0, 0, \ldots, 1)$$



  - 2D: $\hat{\mathbf{x}}, \hat{\mathbf{y}}$
  - 3D: $\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}$
- Any n linearly independent vectors form a basis
  - Dot and cross products have same formula in right-handed orthonormal bases (but ***not*** in other bases!)

Change of Basis (example in 3D)

- Let $[\mathbf{u}, \mathbf{v}, \mathbf{n}]$ be a basis B for a vector space
- Let coordinates of $\mathbf{u}$, $\mathbf{v}$, $\mathbf{n}$ in another basis B_0 be

  $\mathbf{u} = (u_1, u_2, u_3)$
  $\mathbf{v} = (v_1, v_2, v_3)$
  $\mathbf{n} = (n_1, n_2, n_3)$
- Let coordinates of $\mathbf{a}$ in B be (a_1, a_2, a_3)
- Let coordinates of $\mathbf{a}$ in B_0 be $(a^0{}_1, a^0{}_2, a^0{}_3)$

Change of Basis (example in 3D)

$$\begin{bmatrix} a_1^0 \\ a_2^0 \\ a_3^0 \end{bmatrix} = \begin{bmatrix} u_1 & v_1 & n_1 \\ u_2 & v_2 & n_2 \\ u_3 & v_3 & n_3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Change of Basis (example in 3D)

- If B and B_0 are orthonormal, inverse of matrix is its transpose (*orthogonal matrix*)

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ n_1 & n_2 & n_3 \end{bmatrix} \begin{bmatrix} a_1^0 \\ a_2^0 \\ a_3^0 \end{bmatrix}$$

(only if bases are orthonormal)

Metric

- Assign non-negative real number $\mathrm{d}(\mathbf{u}, \mathbf{v})$ to every pair of vectors $\mathbf{u}$, $\mathbf{v}$
  - Interpret as distance between $\mathbf{u}$ and $\mathbf{v}$
- Necessary properties of metric
  - $\mathrm{d}(\mathbf{u}, \mathbf{v}) \geq 0$
  - $\mathrm{d}(\mathbf{u}, \mathbf{v}) = 0$ if and only if $\mathbf{u} = \mathbf{v}$
  - $\mathrm{d}(\mathbf{u}, \mathbf{v}) = \mathrm{d}(\mathbf{v}, \mathbf{u})$
  - $\mathrm{d}(\mathbf{u}, \mathbf{v}) \leq \mathrm{d}(\mathbf{u}, \mathbf{w}) + \mathrm{d}(\mathbf{w}, \mathbf{v})$ (*Triangle Inequality*)

$\mathbb{R}^n$ as Metric Space

- Magnitude: L_2 *norm*: $\| \mathbf{u} \| = (u_1{}^2 + u_2{}^2 + \ldots + u_n{}^2)^{1/2}$
- Metric $\mathrm{d}(\mathbf{u}, \mathbf{v})$ defined as $\| \mathbf{u} - \mathbf{v} \|$
- Other common magnitude functions: L_p *norms*

  $\| \mathbf{u} \| = (|u_1|^p + |u_2|^p + \ldots + |u_n|^p)^{1/p}$
  - L_1 is "Manhattan" norm
  - L_∞ is "max" norm

Inner Product

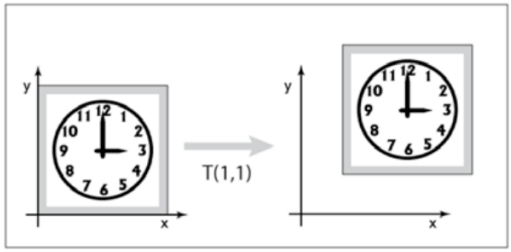
- Assign scalar $\langle \mathbf{u}, \mathbf{v} \rangle \in \mathrm{F}$ to pair of vectors $\mathbf{u}$, $\mathbf{v}$
  - F is assumed to be set of real or complex numbers
- Necessary properties of inner product
  - $\langle \mathbf{u}, \mathbf{v} \rangle = \overline{\langle \mathbf{v}, \mathbf{u} \rangle}$ (overline denotes complex conjugate)
  - $\langle s\mathbf{u}, \mathbf{v} \rangle = s\langle \mathbf{u}, \mathbf{v} \rangle$
  - $\langle \mathbf{u} + \mathbf{v}, \mathbf{w} \rangle = \langle \mathbf{u}, \mathbf{w} \rangle + \langle \mathbf{v}, \mathbf{w} \rangle$
- The dot product is a valid inner product for $\mathbb{R}^n$

Transformations of Vectors

(Thanks to Pat Hanrahan for this section)

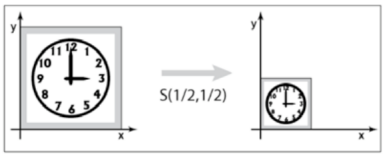
- $\mathbf{u}' = \mathrm{T}(\mathbf{u})$
- Why?
  - Modeling:
    - Create objects in convenient coordinates
    - Multiple instances of prototype shape
    - Kinematics of linked structures
  - Viewing:
    - Map between window and device coordinates
    - Virtual camera projections: parallel/perspective
- We'll stick to $\mathbb{R}^2$ for now

Translation

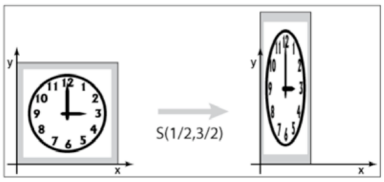


Scaling

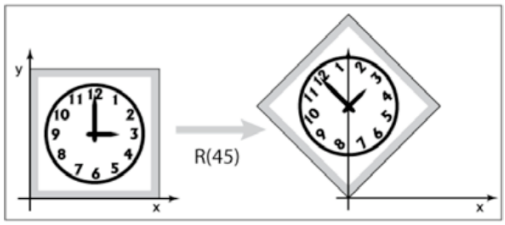
Uniform



Non-uniform

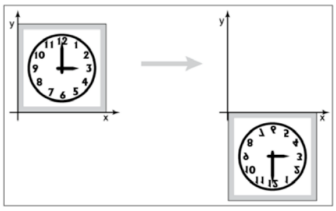


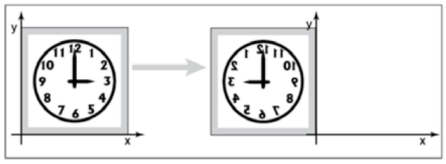
Rotation



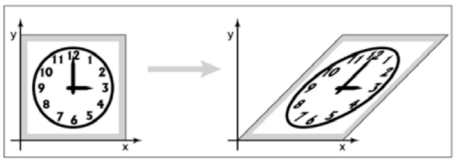
(counter-clockwise about origin)

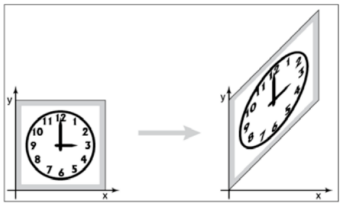
Reflection



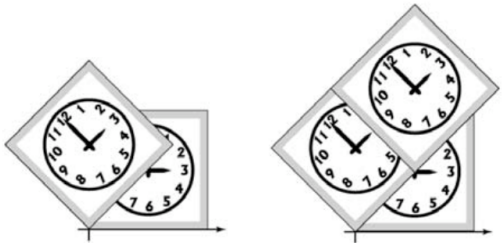


Shear





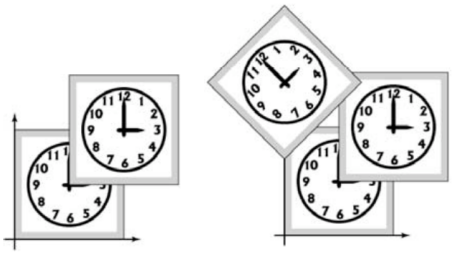
Composing Transformations



R(45) T(1, 1) R(45)

Rotate, then Translate

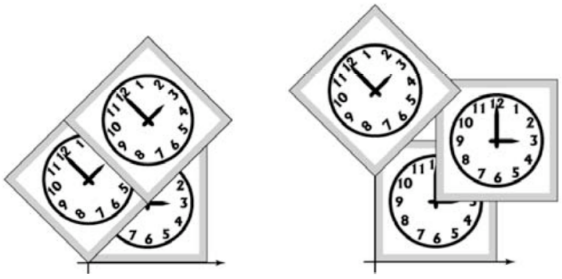
Composing Transformations



T(1, 1) R(45) T(1, 1)

Translate, then Rotate

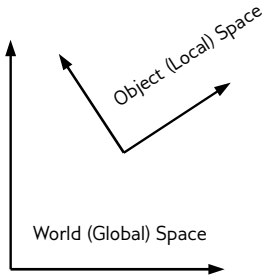
Order Matters!



T(1, 1) R(45) $\neq$ R(45) T(1, 1)

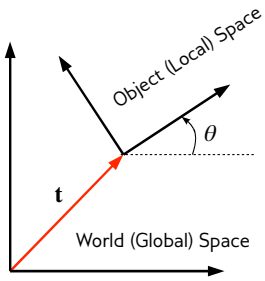
World Space and Object Space

- Transformation maps from one to the other
- Construct by composing sequence of basic transforms
  - Remember: Transforms apply ***Right-to-Left*** in our notation!



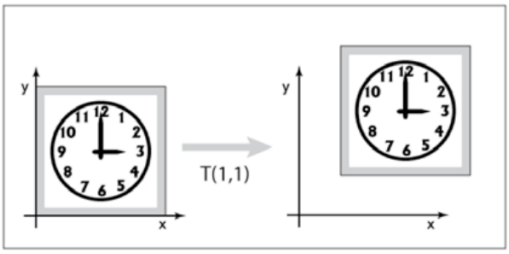
World Space and Object Space

- Let's look at the example on the right
- Object → World:
  - Rotate by θ (ccw), then translate by **t**
- World → Object:
  - Translate by -**t**, then rotate by -θ



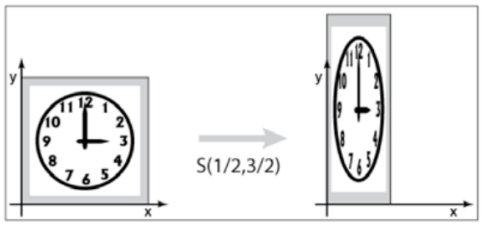
Make sure you understand this!

Translation



$$x' = x + t_x$$
$$y' = y + t_y$$

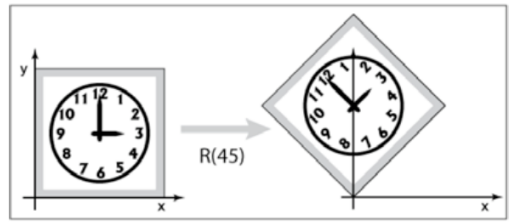
Scaling (and Reflection)



$x' = s_x x$
$y' = s_y y$

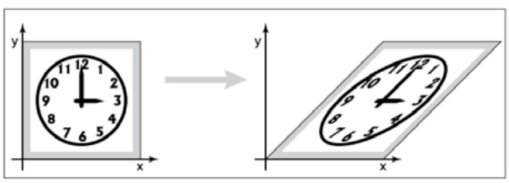
(Negative scaling coefficients give reflection)

CCW Rotation By θ About Origin



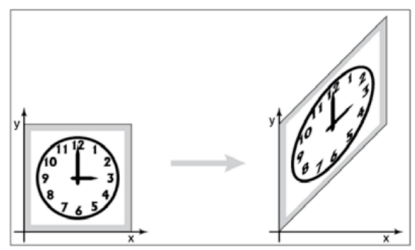
$x' = x \cos\theta - y \sin\theta$
$y' = x \sin\theta + y \cos\theta$

Horizontal Shear



$x' = x + sy$
$y' = y$

Vertical Shear



$x' = x$
$y' = sx + y$

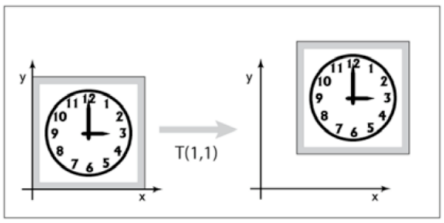
Types of 2D Transformations

- *Linear Transforms*: $\mathrm{T}(\mathbf{u} + \mathbf{v}) = \mathrm{T}(\mathbf{u}) + \mathrm{T}(\mathbf{v})$
  - Scaling
  - Rotation
  - Shear
  - Reflection
- *Affine Transforms*: $\mathrm{T}(\mathbf{u}) = \mathrm{L}(\mathbf{u}) + \mathbf{a}$, where L is linear and $\mathbf{a}$ is a fixed vector
  - Translation
- Other, e.g. perspective projection
- How do we represent these in a common format?

Homogenous Coordinates (2D)

- Point $(x, y) \rightarrow (x, y, 1)$
- Direction $(x, y) \rightarrow (x, y, 0)$
- For any scalar c, $(cx, cy, ch) \equiv (x, y, h)$
- To convert back:
  - If h is 0: $(x, y, 0) \rightarrow (x, y)$
  - If h is non-zero: $(x, y, h) \rightarrow (x / h, y / h)$
- **Note:**
  - ***Not*** 3D vector space, just a new representation for 2D
  - Legal/illegal operations for directions & positions automatically distinguished!

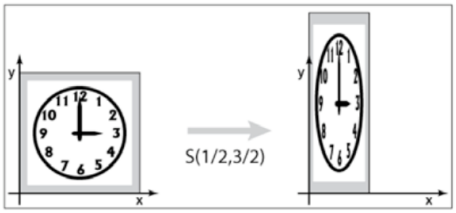
Translation



$$x' = x + t_x \qquad y' = y + t_y$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

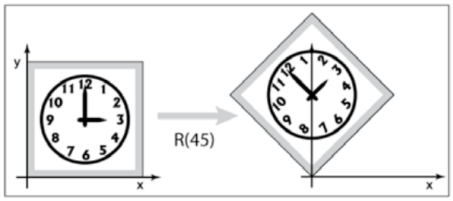
Scaling (and Reflection)



$$x' = s_x x \qquad y' = s_y y$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

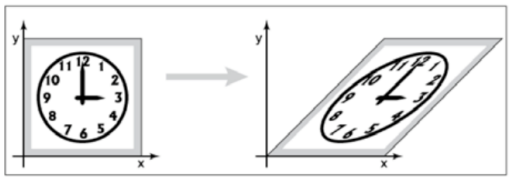
CCW Rotation By θ About Origin



$$x' = x \cos\theta - y \sin\theta$$
$$y' = x \sin\theta + y \cos\theta$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

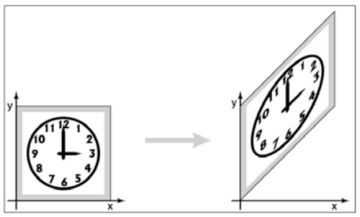
Horizontal Shear



$$x' = x + sy \qquad y' = y$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & s & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Vertical Shear



$$x' = x \qquad y' = sx + y$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ s & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

What about 3D?

- Very similar: $(x, y, z) \rightarrow (x, y, z, h)$
- Look up the formulæ!
- Rotation is a mess
  - Common method:
    - Map axis of rotation to a coordinate axis (similar to change of basis)
    - Rotate around the coordinate axis
    - Map back
  - Other approaches based on Euler angles and quaternions

Why Use Matrices?

- Compute the matrix once

$$x' = x \cos\theta - y \sin\theta$$

$$y' = x \sin\theta + y \cos\theta$$

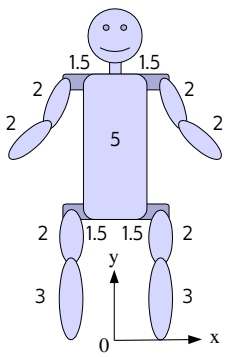
  - Don't repeatedly evaluate sines and cosines

- Combine sequence of transforms into a single transform
  - Store $M = ABCD$, apply $M(\mathbf{u})$ instead of $A(B(C(D(\mathbf{u}))))$

Hierarchical Modeling

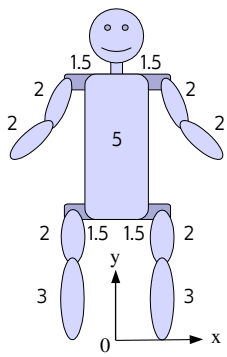
- Graphics systems maintain a *current transformation matrix* (CTM)
  - All geometry is transformed by the CTM
  - CTM defines object space in which geometry is specified
  - Transformation commands are concatenated onto the CTM. The last one added is applied first:
    - CTM = CTM * T
- Graphics systems also maintain a *transformation stack*
  - The CTM can be pushed onto the stack
  - The CTM can be restored from the stack

Example: Articulated Robot




```
body
    torso
        head
        shoulder
            leftArm
                upperArm
                lowerArm
                hand
            rightArm
                upperArm
                lowerArm
                hand
        hips
            leftLeg
                upperLeg
                lowerLeg
                foot
            rightLeg
                upperLeg
                lowerLeg
                foot
```


Example: Articulated Robot




```
translate(0, 5, 0);
torso();
pushMatrix();
translate(0, 5, 0);
    shoulder();
    pushMatrix();
        rotateY(neck_y);
        rotateX(neck_x);
        head();
    popMatrix();
    pushMatrix();
        translate(1.5, 0, 0);
        rotateX(l_shoulder_x);
        upperArm();
        pushMatrix();
            translate(0,-2,0);
            rotateX(l_elbow_x);
            lowerArm();
            ...
        popMatrix();
    popMatrix();
...
```