Today

- 2D Transformations
  - “Primitive” Operations
    - Scale, Rotate, Shear, Flip, Translate
  - Homogenous Coordinates
  - SVD
  - Start thinking about rotations...
Introduction

- Transformation: An operation that changes one configuration into another
- For images, shapes, etc.
  A geometric transformation maps positions that define the object to other positions
  Linear transformation means the transformation is defined by a linear function... which is what matrices are good for.

Some Examples

Original
Rotation
Shear
Uniform Scale
Nonuniform Scale

Images from Conan The Destroyer, 1984
### Mapping Function

\[
f(x) = x \text{ in old image}
\]

\[c(x) = [195, 120, 58] \quad c'(x) = c(f(x))\]

### Linear -vs- Nonlinear

- Linear (shear)
- Nonlinear (swirl)
Geometric -vs- Color Space

Color Space Transform
(edge finding)

Linear Geometric
(flip)

Instancing

M.C. Escher, from Ghostscript 8.0 Distribution
Instancing

- Reuse geometric descriptions
- Saves memory

Linear is Linear

- Polygons defined by points
- Edges defined by interpolation between two points
- Interior defined by interpolation between all points
- Linear interpolation
Linear is Linear

- Composing two linear function is still linear
- Transform polygon by transforming vertices

\[ f(x) = a + bx \quad g(f) = c + df \]

\[ g(x) = c + df(x) = c + ad + bdx \]

\[ g(x) = a' + b'x \]
Points in Space

- Represent point in space by vector in $\mathbb{R}^n$
  - Relative to some origin!
  - Relative to some coordinate axes!
- Later we'll add something extra...

$\mathbf{p} = [4, 2]^T$

Basic Transformations

- Basic transforms are: rotate, scale, and translate
- Shear is a composite transformation!
Linear Functions in 2D

\[ x' = f(x, y) = c_1 + c_2x + c_3y \]
\[ y' = f(x, y) = d_1 + d_2x + d_3y \]

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix}
= \begin{bmatrix} t_x \\ t_y \end{bmatrix} +
\begin{bmatrix}
  M_{xx} & M_{xy} \\
  M_{yx} & M_{yy}
\end{bmatrix}
\cdot
\begin{bmatrix} x \\ y \end{bmatrix}
\]

\[ x' = t + M \cdot x \]

Rotations

\[ p' = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\
\sin(\theta) & \cos(\theta) \end{bmatrix} p \]

45 degree rotation
Rotations

- Rotations are positive counter-clockwise
- Consistent w/ right-hand rule
- Don’t be different...
- Note:
  - rotate by zero degrees give identity
  - rotations are modulo 360 (or $2\pi$)

Rotations

- Preserve lengths and distance to origin
- Rotation matrices are orthonormal
- $\text{Det}(R) = 1 \neq -1$
- In 2D rotations commute...
  - But in 3D they won’t!
Scales

\[ p' = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} p \]

- **Uniform/isotropic**
- **Non-uniform/anisotropic**

**Diagonal matrices**
- Diagonal parts are scale in \( X \) and scale in \( Y \) directions
- Negative values flip
- Two negatives make a positive (180 deg. rotation)
- Really, axis-aligned scales
Shears

\[ p' = \begin{bmatrix} 1 & H_{yx} \\ H_{xy} & 1 \end{bmatrix} p \]

- Shears are not really primitive transforms
- Related to non-axis-aligned scales
- More shortly.....
### Translation

- This is the not-so-useful way:

\[
p' = p + \begin{bmatrix} t_x \\ t_y \end{bmatrix}
\]

Note that its not like the others.

### Arbitrary Matrices

- For everything but translations we have:

\[
x' = A \cdot x
\]

- Soon, translations will be assimilated as well

- What does an arbitrary matrix mean?
Singular Value Decomposition

- For any matrix, $A$, we can write SVD:
  $$A = QSR^T$$
  where $Q$ and $R$ are orthonormal and $S$ is diagonal

- Can also write Polar Decomposition:
  $$A = QRSR^T$$
  where $Q$ is still orthonormal

Decomposing Matrices

- We can force $Q$ and $R$ to have $\text{Det}=1$ so they are rotations
- Any matrix is now:
  - Rotation:Rotation:Scale:Rotation
  - See, shear is just a mix of rotations and scales
Composition

- Matrix multiplication composites matrices
  \[ p' = BAp \]
  "Apply A to p and then apply B to that result."
  \[ p' = B(Ap) = (BA)p = Cp \]
- Several translations composites to one
- Translations still left out...

\[ p' = B(Ap + t) = BAp + Bt = Cp + u \]
### Composition

Transformations built up from others

SVD builds from scale and rotations

Also build other ways

**i.e.** 45 deg rotation built from shears

---

### Homogeneous Coordinates

- Move to one higher dimensional space
  - Append a 1 at the end of the vectors

<table>
<thead>
<tr>
<th>(p = \begin{bmatrix} p_x \ p_y \end{bmatrix} )</th>
<th>( \tilde{p} = \begin{bmatrix} p_x \ p_y \ 1 \end{bmatrix} )</th>
</tr>
</thead>
</table>

- For *directions* the extra coordinate is a zero
Homogeneous Translation

\[
\tilde{p}' = \begin{bmatrix}
1 & 0 & t_x \\
0 & 1 & t_y \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
p_x \\
p_y \\
1
\end{bmatrix}
\]

\[
\tilde{p}' = \tilde{A}\tilde{p}
\]

The tildes are for clarity to distinguish homogenized from non-homogenized vectors.

Homogeneous Others

\[
\tilde{A} = \begin{bmatrix}
A & 0 \\
0 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

Now everything looks the same...
Hence the term “homogenized!”
Compositing Matrices

- Rotations and scales always about the origin
- How to rotate/scale about another point?

Rotate About Arb. Point

- Step 1: Translate point to origin

Translate (-C)
Rotate About Arb. Point

- Step 1: Translate point to origin
- Step 2: Rotate as desired

Translate (-C)
Rotate (θ)

Translate (C)
• Step 1: Translate point to origin
• Step 2: Rotate as desired
• Step 3: Put back where it was

\[ \hat{p}' = (-T)RT\hat{p} = A\hat{p} \]

Don’t negate the 1...
Scale About Arb. Axis

• Diagonal matrices scale about coordinate axes only:

Not axis-aligned

Scale About Arb. Axis

• Step 1: Translate axis to origin
Scale About Arb. Axis

- Step 1: Translate axis to origin
- Step 2: Rotate axis to align with one of the coordinate axes
- Step 3: Scale as desired
Scale About Arb. Axis

- Step 1: Translate axis to origin
- Step 2: Rotate axis to align with one of the coordinate axes
- Step 3: Scale as desired
- Steps 4&5: Undo 2 and 1 (reverse order)

Order Matters!

- The order that matrices appear in matters
  \[ A \cdot B \neq BA \]
- Some special cases work, but they are special
- But matrices are associative
  \[ (A \cdot B) \cdot C = A \cdot (B \cdot C) \]
- Think about efficiency when you have many points to transform...
Matrix Inverses

• In general: $A^{-1}$ undoes effect of $A$

• Special cases:
  • Translation: negate $t_x$ and $t_y$
  • Rotation: transpose
  • Scale: invert diagonal (axis-aligned scales)

• Others:
  • Invert matrix
  • Invert SVD matrices

Point Vectors / Direction

• Points in space have a 1 for the “w” coordinate

• What should we have for $a - b$?
  • $w = 0$
    • Directions not the same as positions
    • Difference of positions is a direction
    • Position + direction is a position
    • Direction + direction is a direction
    • Position + position is nonsense
Some things require care.

For example, normals do not transform normally.

\[ \mathbf{M}(\mathbf{a} \times \mathbf{b}) \neq (\mathbf{M}a) \times (\mathbf{M}b) \]

Use inverse transpose of the matrix for normals.
See text book.

Suggested Reading

- Fundamentals of Computer Graphics by Pete Shirley
  - Chapter 5
  - And re-read chapter 4 if your linear algebra is rusty!