2D transformations
Homogeneous coordinates

Uses of Transformations
- Modeling: position and resize parts of a complex model;
- Viewing: define and position the virtual camera
- Animation: define how objects move/change with time

Transformations
Examples of transformations:
- Translation
- Rotation
- Scaling
- Shear

More examples:
- Reflection with respect to the y-axis
- Reflection with respect to the origin

Transformations and matrices
Any affine transformation can be written as
\[
\begin{pmatrix}
    x' \\
    y'
\end{pmatrix}
= \begin{pmatrix}
    a_{11} & a_{12} \\
    a_{21} & a_{22}
\end{pmatrix}
\begin{pmatrix}
    x \\
    y
\end{pmatrix}
+ \begin{pmatrix}
    b_1 \\
    b_2
\end{pmatrix}
\]

Images of basis vectors under affine transformations:
\[
\begin{pmatrix}
    e_x \\
    e_y
\end{pmatrix}
\rightarrow
\begin{pmatrix}
    Ae_x \\
    Ae_y
\end{pmatrix}
\]

Linear transformations: take straight lines to straight lines.
All of the examples are linear.
Affine transformations: take parallel lines to parallel lines.
All of the examples are affine, an example of linear non-affine is perspective projection.
Orthogonal transformations: preserve distances, move all objects as rigid bodies.
Rotation, translation and reflections are affine.
Transformations and matrices

Matrices of some transformations:

- **Shear**: 
  \[
  \begin{pmatrix}
  1 & 1 \\
  0 & 1 \\
  \end{pmatrix}
  \]
  Scale by factor s

- **Rotation**:
  \[
  \begin{pmatrix}
  \cos \alpha & -\sin \alpha \\
  \sin \alpha & \cos \alpha \\
  \end{pmatrix}
  \]

- **Reflection with respect to the origin**:
  \[
  \begin{pmatrix}
  -1 & 0 \\
  0 & -1 \\
  \end{pmatrix}
  \]

- **Reflection with respect to the y-axis**:
  \[
  \begin{pmatrix}
  -1 & 0 \\
  0 & 1 \\
  \end{pmatrix}
  \]

Problem

Even for affine transformations we cannot write them as a single 2x2 matrix; we need an additional vector for translations.

We cannot write all linear transformations even in the form \( Ax + b \) where \( A \) is a 2x2 matrix and \( b \) is a 2d vector. Example: perspective projection

\[
\begin{align*}
x' &= 1 \\
y' &= y/x \\
x &= 1 \\
\end{align*}
\]

Equations not linear!

Homogeneous coordinates

- Replace 2d points with 3d points, last coordinate 1
- For a 3d point \((x,y,w)\) the corresponding 2d point is \((x/w,y/w)\) if \(w\) is not zero
- Each 2d point \((x,y)\) corresponds to a line in 3d; all points on this line can be written as \([kx,ky,k]\) for some \(k\).
- \((x,y,0)\) does not correspond to a 2d point, corresponds to a direction (will discuss later)
- Geometric construction: 3d points are mapped to 2d points by projection to the plane \(z=1\) from the origin

Homogeneous transformations

Any linear transformation can be written in matrix form in homogeneous coordinates.

Example 1: translations

\[ [x,y] \text{ in hom. coords is } [x,y,1] \]

Example 2: perspective projection

\[
\begin{pmatrix}
  x' \\
  y' \\
  w' \\
\end{pmatrix} =
\begin{pmatrix}
  1 & 0 & 0 \\
  0 & 1 & 0 \\
  0 & 0 & 1 \\
\end{pmatrix}
\]

Homogeneous coordinates

- From homogeneous to 2d: \([x,y,w]\) becomes \([x/w,y/w]\)
- From 2d to homogeneous: \([x,y]\) becomes \([kx,ky,k]\)
  (can pick any nonzero \(k\)!)
Matrices of basic transformations

\[
\begin{bmatrix}
  \\
  \cos \theta & -\sin \theta & 0 \\
  \sin \theta & \cos \theta & 0 \\
  0 & 0 & 1
\end{bmatrix}
\]
rotation

\[
\begin{bmatrix}
  1 & 0 & t_x \\
  0 & 1 & t_y \\
  0 & 0 & 1
\end{bmatrix}
\]
translation

\[
\begin{bmatrix}
  s_x & 0 & 0 \\
  0 & s_y & 0 \\
  0 & 0 & 1
\end{bmatrix}
\]
scaling

\[
\begin{bmatrix}
  1 & 0 & 0 \\
  0 & 1 & 0 \\
  0 & 0 & 1
\end{bmatrix}
\]
skew

\[
\begin{bmatrix}
  \alpha_{11} & \alpha_{12} & \alpha_{13} \\
  \alpha_{21} & \alpha_{22} & \alpha_{23} \\
  0 & 0 & 1
\end{bmatrix}
\]
general affine transform

Homogeneous line equation

Implicit line equation in 2D: \((n, q-p) = 0\),
\(n = 2D\) vector, \(p = 2D\) point on the line.
Goal: rewrite in homogeneous coordinates.

2D point corresponds to a 3D line through origin;
2D line corresponds to a plane through origin
In other words, the 2D line is intersection of a plane through origin with the plane \(z=1\).

Homogeneous form of the line equation:

\((N, \overline{q}) = 0\)

Intersection of two lines

Let homogeneous equations of the two lines be
\((N_1, q_1)=0\), and \((N_2, q_2)=0\).
The two lines are intersections of planes with normals \(N_1\) and \(N_2\) with the plane \(z=1\).
Intersection of lines = intersection point of the common line of the two planes with the plane \(z=1\).
The direction of the common line is the cross product:
\(N_1 \times N_2\)
This is also (one of possible) homogeneous forms of the intersection point!

How to intersect two lines

Step 1: Convert equations to homogeneous form.
Step 2: Compute the vector product \(N_1 \times N_2\)
Step 3: Convert \(N_1 \times N_2\) from homogeneous coordinates to 2D, dividing by the last component (may be not 1!)

Line through two points

Goal: given two points in homogeneous coordinates, find the homogeneous equation of the line through these points, that is, the vector \(N\) in the equation \((N, q)=0\).
Homogeneous form of a point \(= 3D\) vector from the origin to the point.
Line through 2 points = intersection of the plane spanned by two lines with the plane \(z=1\)
Normal to that plane (same as \(N\) in the line equation):
\[N = p_1 \times p_2 = [p_1, p_2, 1] \times [p_1, p_2, 1]\]
Intersection of two lines

Intersection of two lines through two pairs of points all given in homogeneous form.

Apply formulas for line through two points and for intersection of lines:

\[ q' = (p' \times p^2) \times (p' \times p^4) \]

Mapping quads to quads

A common operation for texture mapping: find a transformation mapping a rectangular image to a quadrilateral.

Goal: solve a somewhat more general problem: find a linear transform mapping one quad \((A,B,C,D)\) to another \((a,b,c,d)\). Assume all points given in homogeneous coordinates.

Idea: to define a 3x3 linear transform (=matrix) \(M\), it is sufficient to define its action on three vectors. Choose suitable points for which we know what the images should be.

Transform from 3 points

Suppose for known vectors \(H_1, H_2, H_3\) we know that the images should be \(h_1, h_2, h_3\):

\[ M H_1 = h_1, M H_2 = h_2, M H_3 = h_3. \]

Combine three vector equations into one matrix equation:

\[
\begin{bmatrix}
H_1 & H_2 & H_3 \\
H_4 & H_5 & H_6 \\
M & h & h
\end{bmatrix}
\]

solving \(MH=h\), we get \(M = hH^{-1}\)

Intersections of sides and diagonals

Intersections of sides and diagonals:

\[ H_1 = (B \times A) \times (C \times D) \]
\[ H_2 = (A \times D) \times (B \times C) \]
\[ H_3 = (A \times C) \times (B \times D) \]

Similar for \(h_1, h_2, h_3\)

To find the matrix \(M\), build matrices \(H\) and \(h\), and compute \(HH^{-1}\).

Mapping rectangles to quads

Problem: map a square \([-1, 1] \times [-1, 1]\) to an arbitrary quadrilateral with vertices \(a,b,c,d\) using a linear map \(M\).
Points at infinity

Idea: look at images of homogeneous points $H_2 = [1,0,0]$, $H_1 = [0,1,0]$, $H_3 = [0,0,1]$. In this case the matrix $H$ is identity, and $M = h$.

What are points with last component zero?

- $[1,0,0]$ “intersection” of horizontal sides (point at infinity for direction $[1,0]$)
- $[1,0,0]$ “intersection” of vertical sides (point at infinity for direction $[0,1]$)
- $[0,0,1]$ intersection of diagonals

Mapping rectangles to quads

Using formulas for intersections of lines, we get

- $h_1 = (b \times a) \times (c \times d)$
- $h_2 = (a \times d) \times (b \times c)$
- $h_3 = (a \times c) \times (b \times d)$

Finally, we obtain the transform:

$$ M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} h_{22} & h_{23} & h_{21} \\ h_{32} & h_{33} & h_{31} \\ h_{12} & h_{13} & h_{11} \end{bmatrix} $$

Note the last line is not $[0,0,1]$, this is not an affine transform!

Texture Mapping

- Put image on a quadrilateral (can be thought of as rectangle viewed in perspective)
- Map image to the square $[-1,1] \times [-1,1]$
- Recall scan conversion: we traverse pixels along scan lines intersecting the quadrilateral; for a pixel $p = [i,j]$ need to determine the pixel $[m,n]$ in the image.
- To determine color, need the inverse of $M$:
  - first, find the point $q$ in the square which is mapped to $p$ by $M$,
  - second, find the pixel in the image that maps to $q$.

Putting it all together

Given an an image of size width X height and four corners of a quadrilateral $a, b, c, d$ in homogeneous coordinates:

- compute the matrix $M$ using intersections of sides $h_1$, $h_2$ and intersection of diagonals $h_3$
- compute the inverse of $M$
- scan convert the quadrilateral; for each pixel to be painted, determine the pixel of the image corresponding to it; use its color to paint the pixel

Implementation

- Representing matrices in C: I recommend against using 2d arrays (two many pitfalls) use a 1d array with 9 elements to represent a 3 by 3 matrix, and expressions of the type $m[3*i+j]$ to address element $m[i,j]$; note that indices start from 0.
- Write a small matrix/vector library, implementing operations like add two vectors, multiply a vector by a number, multiply a matrix and a vector, dot product, vector product.
Inverse matrix

Code for inverse:

\[
D = m[0]*m[4]*m[8] - m[0]*m[5]*m[7] - \\
m[3]*m[1]*m[8] + m[3]*m[2]*m[7] + \\
m[6]*m[1]*m[5] - m[6]*m[2]*m[4];
\]

\[
\text{minv[0]} = (m[4]*m[8] - m[5]*m[7]) / D;
\]

\[
\text{minv[1]} = -(m[1]*m[8] - m[2]*m[7]) / D;
\]

\[
\text{minv[2]} = (m[1]*m[5] - m[2]*m[4]) / D;
\]

\[
\text{minv[3]} = -(m[3]*m[8] - m[5]*m[6]) / D;
\]

\[
\text{minv[4]} = (m[0]*m[8] - m[2]*m[6]) / D;
\]

\[
\text{minv[5]} = -(m[0]*m[5] - m[2]*m[3]) / D;
\]

\[
\text{minv[6]} = (m[3]*m[7] - m[4]*m[6]) / D;
\]

\[
\text{minv[7]} = -(m[0]*m[7] - m[4]*m[3]) / D;
\]

\[
\text{minv[8]} = (m[0]*m[4] - m[1]*m[3]) / D;
\]