Lecture 3: Transformations
Who am I?

Pontus Olsson, Centre for Image Analysis (CBA)

Whole Hand Haptics project, lots of transformations!

pontus@cb.uu.se
• Object representations in CG
• Review of linear algebra
• Linear transformations (rotation, scaling) with matrices.
• Translation and homogeneous coordinates.
• Main point: A lot of interesting transforms can be expressed with matrix multiplication.
Transformations?

Some of you may remember the transformers toys...

Images: Hasbro

[ Demo Time! ]
Todays lecture in the graphics pipeline
3D Object representation for CG

- An object is described by its surface (i.e. objects are hollow).
- An object is specified by a set of 3D points (vertices).
- An object is composed of (or can be approximated by) flat convex polygons.
- Transform an object by transforming its vertices
Smooth objects?

- May be approximated by polygons
- If the (screen)size of each polygon is smaller than the size of a pixel, there is no visual difference.
- Modern graphics cards can render LOTS of triangles very fast.
- We will talk more about smooth surfaces later in the course.

CG-dinosaur in Jurassic Park
Other object representations

Solid/Volumetric modelling

Objects are composed of a set of Voxels = 3D pixels
Volumetric Modelling

Volumetric rendering of a foot. (Project at the end of the course)
Implicit surfaces, $F(x,y,z)=C$

Sphere defined implicitly by $x^2 + y^2 + z^2 = 1$
Scalars points and vectors

- **Scalars**
  - Real (or complex) numbers

- **Points**
  - Locations in space (no size or shape)

- **Vectors**
  - Directions in space (magnitude but no position)

1.2

-2.3

7.3i
Vector operations

Review of vector operations:
• Adding/Subtracting two vectors
• Multiplication by scalars
• Magnitude of a vector
• Normalizing a vector
Vector Addition

\[ \mathbf{a} = (2, 4) \]
\[ \mathbf{b} = (2, 2) \]
Vector Addition

\[ \mathbf{a} = (2, 4) \]
\[ \mathbf{b} = (2, 2) \]
\[ \mathbf{a} + \mathbf{b} = \mathbf{c} = (4, 6) \]
\[ \mathbf{a} = (2, 4) \quad \mathbf{b} = (2, 2) \]

\[ \mathbf{b} + \mathbf{a} = \mathbf{c} = (4, 6) \]
Vector Subtraction

\[ \mathbf{a} = (2, 4) \]
\[ \mathbf{b} = (2, 2) \]
\[ \mathbf{a} - \mathbf{b} = (2, 4) - (2, 2) = (0, 2) = \mathbf{d} \]
Vector Subtraction

\[ \textbf{a} = (2, 4) \]
\[ \textbf{b} = (2, 2) \]

\[ \textbf{b} - \textbf{a} = (2, 2) - (2, 4) = (0, -2) = \textbf{e} \neq \textbf{d} \]

Order is important when subtracting vectors!
a = (4, 2)
Vector Multiplied by Scalar

\[ \mathbf{b} = 2 \times \mathbf{a} = 2 \times (4, 2) = (8, 4) \]
Vector Multiplied by Scalar

\[ \mathbf{b} = -0.5 \times \mathbf{a} = -0.5 \times (4, 2) = (-2, -1) \]
Length of Vector

\[ \mathbf{a} = (a_x, a_y) \]

\[ \|\mathbf{a}\| = \sqrt{a_x^2 + a_y^2} \]

Length of vector \( \mathbf{a} \) (Euclidian norm)
Length of Vector

\[ \mathbf{a} = (a_x, a_y) \quad \Rightarrow \quad ||\mathbf{a}|| = \sqrt{a_x^2 + a_y^2} \]

Length of vector \( \mathbf{a} \) (Euclidian norm)

\[ \mathbf{b} = (b_x, b_y, b_z) \quad ||\mathbf{b}|| = \sqrt{b_x^2 + b_y^2 + b_z^2} \]
Length of Vector

\[ \mathbf{a} = (a_x, a_y) \quad \Rightarrow \quad ||\mathbf{a}|| = \sqrt{a_x^2 + a_y^2} \]

Length of vector \( \mathbf{a} \) (Euclidean norm)

\[ \mathbf{b} = (b_x, b_y, b_z) \quad ||\mathbf{b}|| = \sqrt{b_x^2 + b_y^2 + b_z^2} \]

\[ ||\mathbf{a}|| = \sqrt{\mathbf{a} \cdot \mathbf{a}} \]

Dot product
Normalize a vector = set its length to 1.
Divide the vector by its Euclidean norm (length)

\[ \hat{a} = \frac{a}{||a||} \]
Matrix multiplication

\[
\begin{pmatrix}
a & b & c \\
d & e & f \\
g & h & i
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z
\end{pmatrix}
= 
\begin{pmatrix}
ax + by + cz \\
dx + ey + fz \\
gx + hy + iz
\end{pmatrix}
\]

\[
\begin{pmatrix}
a & b & c \\
d & e & f \\
g & h & i
\end{pmatrix}
\begin{pmatrix}
j & k & l \\
m & n & o \\
p & q & r
\end{pmatrix}
= 
\begin{pmatrix}
aj + bm + cp \\
dj + em + fp \\
gj + hm + ip
ak + bn + cq \\
dk + en + fq \\
gk + hn + iq
al + bo + cr \\
dl + eo + fr \\
gl + ho + ir
\end{pmatrix}
\]
• Convex combinations (coefficients are all positive and add up to 1)

• E.g., linear interpolation: \( p(t) = (t)a + (1-t)b \) where \( a \) and \( b \) are points and \( 0 \leq t \leq 1 \).
Linear combinations of vectors

\[ a = (2, 6) \]
\[ b = (10, 4) \]
\[ c = t^*b + (1-t)^*a \]

- \( t = 0.0 \)
- \( t = 0.25 \)
- \( t = 0.50 \)
- \( t = 0.75 \)
- \( t = 1.0 \)
Dot products

\[ a \cdot b = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \]

\[ a \cdot b = \|a\| \|b\| \cos \theta \]

- Vectors are orthogonal if dot product is 0.
- Vectors are less than 90° apart if dot product is positive.
• Perpendicular projection of $\mathbf{a}$ onto $\mathbf{b}$ is $(\mathbf{a} \cdot \mathbf{b})\mathbf{b}$
Dot products

- Perpendicular projection of $\mathbf{a}$ onto $\mathbf{b}$ is $(\mathbf{a} \cdot \hat{\mathbf{b}}) \hat{\mathbf{b}}$

$c = \text{projection of } \mathbf{a} \text{ onto } \mathbf{b}$
• How do we get the reflection $\mathbf{b}$ of a vector $\mathbf{a}$?
• Reflection $b$ of $a$ is $-a + 2(a \cdot \hat{n})\hat{n}$ where $\hat{n}$ is the unit normal vector to the reflection surface.
Cross products

\[ \mathbf{u} = (u_x, u_y, u_z), \quad \mathbf{v} = (v_x, v_y, v_z) \]

\[ \mathbf{u} \times \mathbf{v} = \begin{vmatrix} x & y & z \\ u_x & u_y & u_z \\ v_x & v_y & v_z \end{vmatrix} \]

• \( \mathbf{u} \times \mathbf{v} \) is perpendicular to both \( \mathbf{u} \) and \( \mathbf{v} \).
• Orientation follows the right hand rule.
• \( |\mathbf{u} \times \mathbf{v}| = |\mathbf{u}| |\mathbf{v}| \sin \Theta \)
Cross products

\[ \mathbf{u} \times \mathbf{v} = \begin{vmatrix} x & y & z \\ u_x & u_y & u_z \\ v_x & v_y & v_z \end{vmatrix} \]

\[ = (u_yv_z - u_zv_y)x - (u_xv_z - u_zv_x)y + (u_xv_y - u_yv_x)z \]

Determinant

Right hand rule
Transformations

- To create and move objects we need to be able to transform objects in different ways.
- There are many classes of transformations.
  - Translation (move around)
  - Rotation
  - Scaling

- Can you think of more?
Transformations

• To create and move objects we need to be able to transform objects in different ways.

• There are many classes of transformations.
  – Translation (move around)
  – Rotation
  – Scaling

• Can you think of more?
  - Shear
  - Mirroring/Flipping
Translation

Simply add a translation vector

\[ x' = x + dx \]
\[ y' = y + dy \]
Scaling (about the origin)

Multiply by a scale factor

\[ x' = s_x x \]
\[ y' = s_y y \]
Rotation (about the origin)

\[ P(x, y) \text{ in polar coordinates} \]
\[ x = r \cos(\phi) \quad y = r \sin(\phi) \]

\[ P'(x', y') \text{ in polar coordinates} \]
\[ x' = r \cos(\theta + \phi) = r \cos(\theta) \cos(\phi) - r \sin(\phi) \sin(\theta) \]
\[ y' = r \sin(\theta + \phi) = r \cos(\phi) \sin(\theta) + r \sin(\phi) \cos(\theta) \]
Rotation (about the origin)

\[ P(x,y) \text{ in polar coordinates} \]
\[ x = r \cos(\phi) \quad y = r \sin(\phi) \]

\[ P'(x',y') \text{ in polar coordinates} \]
\[ x' = r \cos(\theta + \phi) = r \cos(\theta) \cos(\phi) - r \sin(\phi) \sin(\theta) \]
\[ y' = r \sin(\theta + \phi) = r \cos(\phi) \sin(\theta) + r \sin(\phi) \cos(\theta) \]

Substitute \( x \) and \( y \)
\[ x' = x \cos(\theta) - y \sin(\theta) \]
\[ y' = x \sin(\theta) + y \cos(\theta) \]
Vectors and matrices (2D)

\[ x' = x \cos(\theta) - y \sin(\theta) \]
\[ y' = x \sin(\theta) + y \cos(\theta) \]

Rotation on matrix form:

\[
\begin{pmatrix}
  x' \\
  y'
\end{pmatrix}
= 
\begin{pmatrix}
  \cos(\theta) & -\sin(\theta) \\
  \sin(\theta) & \cos(\theta)
\end{pmatrix}
\begin{pmatrix}
  x \\
  y
\end{pmatrix}
\]
Arbitrary rotation center

• Translate the rotation center to the origin
• Rotate
• Translate back
Vectors and matrices (2D)

Translation

\[
\begin{pmatrix}
  x' \\
  y'
\end{pmatrix} = \begin{pmatrix}
  x \\
  y
\end{pmatrix} + \begin{pmatrix}
  dx \\
  dy
\end{pmatrix}
\]

Rotation

\[
\begin{pmatrix}
  x' \\
  y'
\end{pmatrix} = \begin{pmatrix}
  \cos(\theta) & -\sin(\theta) \\
  \sin(\theta) & \cos(\theta)
\end{pmatrix} \begin{pmatrix}
  x \\
  y
\end{pmatrix}
\]

Scaling

\[
\begin{pmatrix}
  x' \\
  y'
\end{pmatrix} = \begin{pmatrix}
  s_x & 0 \\
  0 & s_y
\end{pmatrix} \begin{pmatrix}
  x \\
  y
\end{pmatrix}
\]
Translation is different!

"Stepping up one dimension":

\[
\begin{pmatrix}
x' \\
y' \\
W
\end{pmatrix} = \begin{pmatrix}
1 & 0 & dx \\
0 & 1 & dy \\
0 & 0 & 1
\end{pmatrix} \begin{pmatrix}
x \\
y \\
W
\end{pmatrix}
\]

If \( W=1 \), this is called a *Homogenous coordinate*

\[
\begin{pmatrix}
x' \\
y' \\
1
\end{pmatrix} = \begin{pmatrix}
1 & 0 & dx \\
0 & 1 & dy \\
0 & 0 & 1
\end{pmatrix} \begin{pmatrix}
x \\
y \\
1
\end{pmatrix}
\]
Example

\[
\begin{pmatrix}
1 & 0 & -2 \\
0 & 1 & 3 \\
0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
4 \\
6 \\
1
\end{pmatrix}
= \begin{pmatrix}
2 \\
9 \\
1
\end{pmatrix}
\]

Matrix multiplication can be used also for translation!
Why bother?

• Compact notation for concatenated transforms.
• Translation followed by Rotation $\neq$ Rotation followed by Translation
Using homogenous coordinates

Translation

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

Rotation

\[
\begin{pmatrix}
x' \\
y' \\
1
\end{pmatrix} = \begin{pmatrix}
\cos(\theta) & -\sin(\theta) & 0 \\
\sin(\theta) & \cos(\theta) & 0 \\
0 & 0 & 1
\end{pmatrix} \begin{pmatrix}
x \\
y \\
1
\end{pmatrix}
\]

Scaling

\[
\begin{pmatrix}
x' \\
y' \\
1
\end{pmatrix} = \begin{pmatrix}
s_x & 0 & 0 \\
0 & s_y & 0 \\
0 & 0 & 1
\end{pmatrix} \begin{pmatrix}
x \\
y \\
1
\end{pmatrix}
\]
What about the last element $W$?

- If $W=0$, then $P$ is not affected by translations.
- Be aware of this during the assignments!
The order of matrices (right to left) is important!

\[ P' = T^{-1}(S(R(T(P)))) = (T^{-1}SRT)P \]

Example of concatenated transforms:

\[
P' = \begin{pmatrix} 1 & 0 & -4 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 4 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}
\]

Translation  Scaling  Rotation  Translation  \( P \)
Moving to 3D

- Translation and scaling are the same
- Rotation is a bit more complicated, and becomes $R_x$, $R_y$ and $R_z$.
- Observe that $R_xR_y \neq R_yR_x$
3D transformations

Translation

\[
\begin{pmatrix}
  x' \\
  y' \\
  z' \\
  1
\end{pmatrix} =
\begin{pmatrix}
  1 & 0 & 0 & dx \\
  0 & 1 & 0 & dy \\
  0 & 0 & 1 & dz \\
  0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
  x \\
  y \\
  z \\
  1
\end{pmatrix}
\]

Scaling

\[
\begin{pmatrix}
  x' \\
  y' \\
  z' \\
  1
\end{pmatrix} =
\begin{pmatrix}
  s_x & 0 & 0 & 0 \\
  0 & s_y & 0 & 0 \\
  0 & 0 & s_z & 0 \\
  0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
  x \\
  y \\
  z \\
  1
\end{pmatrix}
\]
3D transformation 3

X Rotation
\[
\begin{pmatrix}
x' \\
y' \\
z' \\
1
\end{pmatrix} =
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & \cos(\theta) & -\sin(\theta) & 0 \\
0 & \sin(\theta) & \cos(\theta) & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix}
\]

Y Rotation
\[
\begin{pmatrix}
x' \\
y' \\
z' \\
1
\end{pmatrix} =
\begin{pmatrix}
\cos(\theta) & 0 & \sin(\theta) & 0 \\
0 & 1 & 0 & 0 \\
-\sin(\theta) & 0 & \cos(\theta) & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix}
\]

Z Rotation
\[
\begin{pmatrix}
x' \\
y' \\
z' \\
1
\end{pmatrix} =
\begin{pmatrix}
\cos(\theta) & -\sin(\theta) & 0 & 0 \\
\sin(\theta) & \cos(\theta) & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix}
\]
Rotation about an arbitrary axis $\mathbf{a}$:

- Find a matrix $\mathbf{M}$ that aligns e.g. $(1,0,0)$ with $\mathbf{a}$.
- Apply $\mathbf{M}^{-1}$.
- Rotate around $(1,0,0)$
- Apply $\mathbf{M}$. 
How do we find $\mathbf{M}$?

• The matrix $\mathbf{M}$ represents a *change of coordinate system*.

• This (among other things) is the topic of the next lecture.
Lab 2

- 10 april
- Transformations
- Positioning of a camera in a graphical scene