Computer Graphics

10 – Animation

Yoonsang Lee
Spring 2019

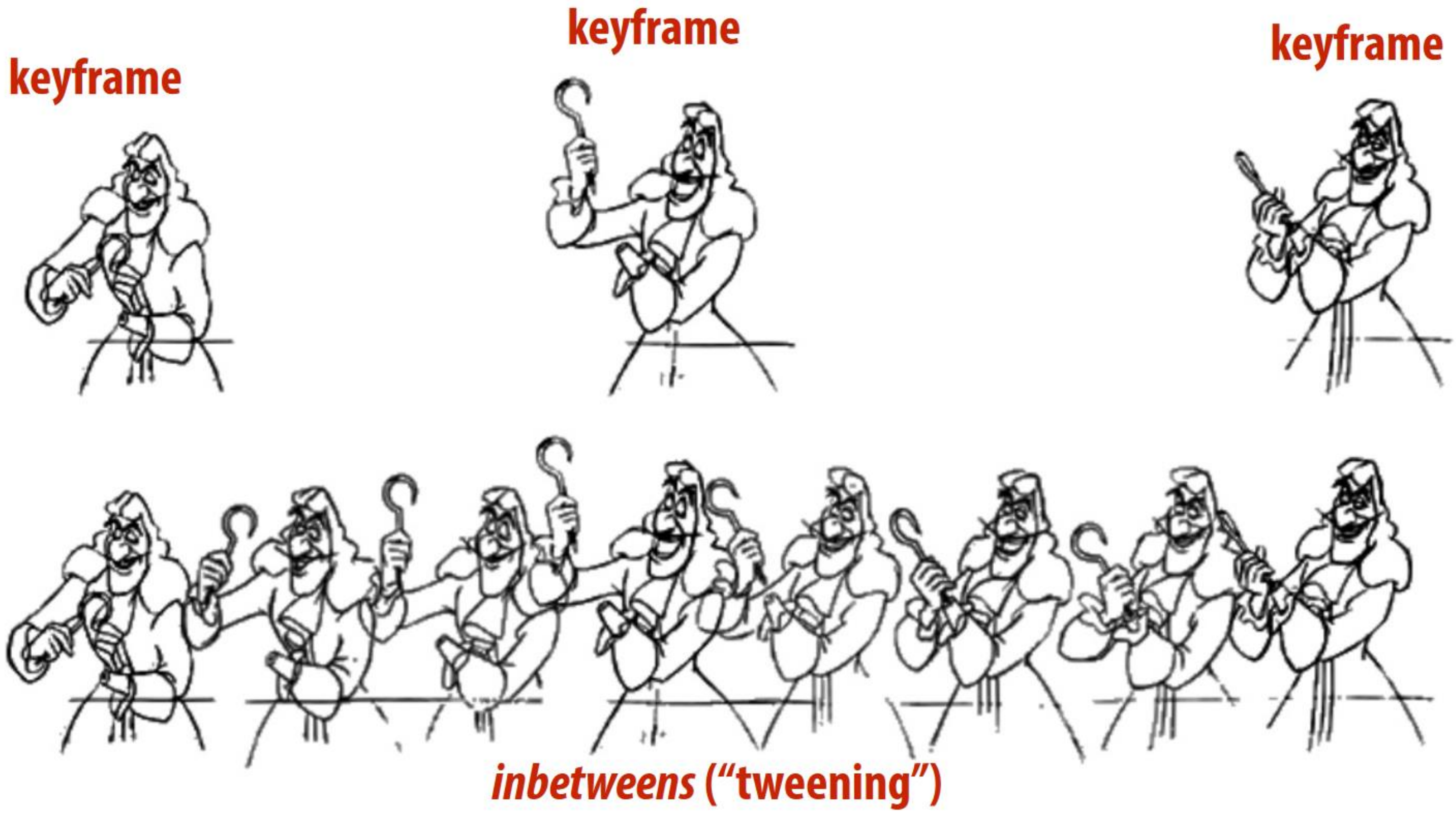
Topics Covered

- Introduction to Computer Animation
- Interpolation
  - Linear Interpolation for Rotation
- Kinematics
  - Forward Kinematics
- BVH File Format (Motion capture data)

Introduction to Computer Animation

Traditional Hand-drawn Cel Animation

- Senior artist draws *keyframes*
- Assistant draws *inbetweens*
- Tedious / labor intensive (opportunity for technology!)



Animation by Milt Kahl (Walt Disney Studios)

Animation by Marc Davis (Walt Disney Studios)

Animation by Mark Henn (Walt Disney Studios)

Animation by Milt Kahl (Walt Disney Studios)

https://www.wnyc.org/story/sideshow-classic-disney-pencil-animations-come-life-gifs/

Computer Animation

- Computers are now widely replacing labor-intensive animation processes.
  - More controllable than drawing images by hands or constructing miniatures.

Computer Animation: State-of-the-art

- Fluid

http://cvc.ucsb.edu/graphics/Papers/SIGGRAPH2018_EigenFluid/

- Face

https://vml.kaist.ac.kr/main/international/individual/133

- Cloth

http://www.cs.columbia.edu/cg/wetcloth/

- Character

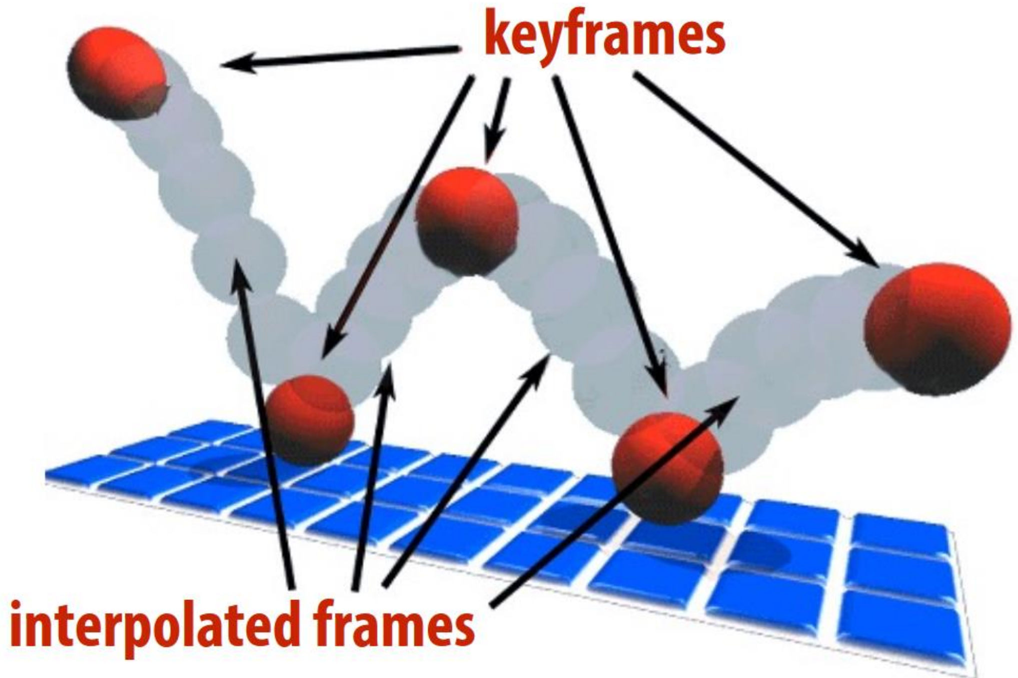
https://xbpeng.github.io/projects/DeepMimic/index.htm

Creating Computer Animation

- Keyframe Animation
- Motion Capture
- Physically-Based Simulation
- (mainly focusing on character animation)

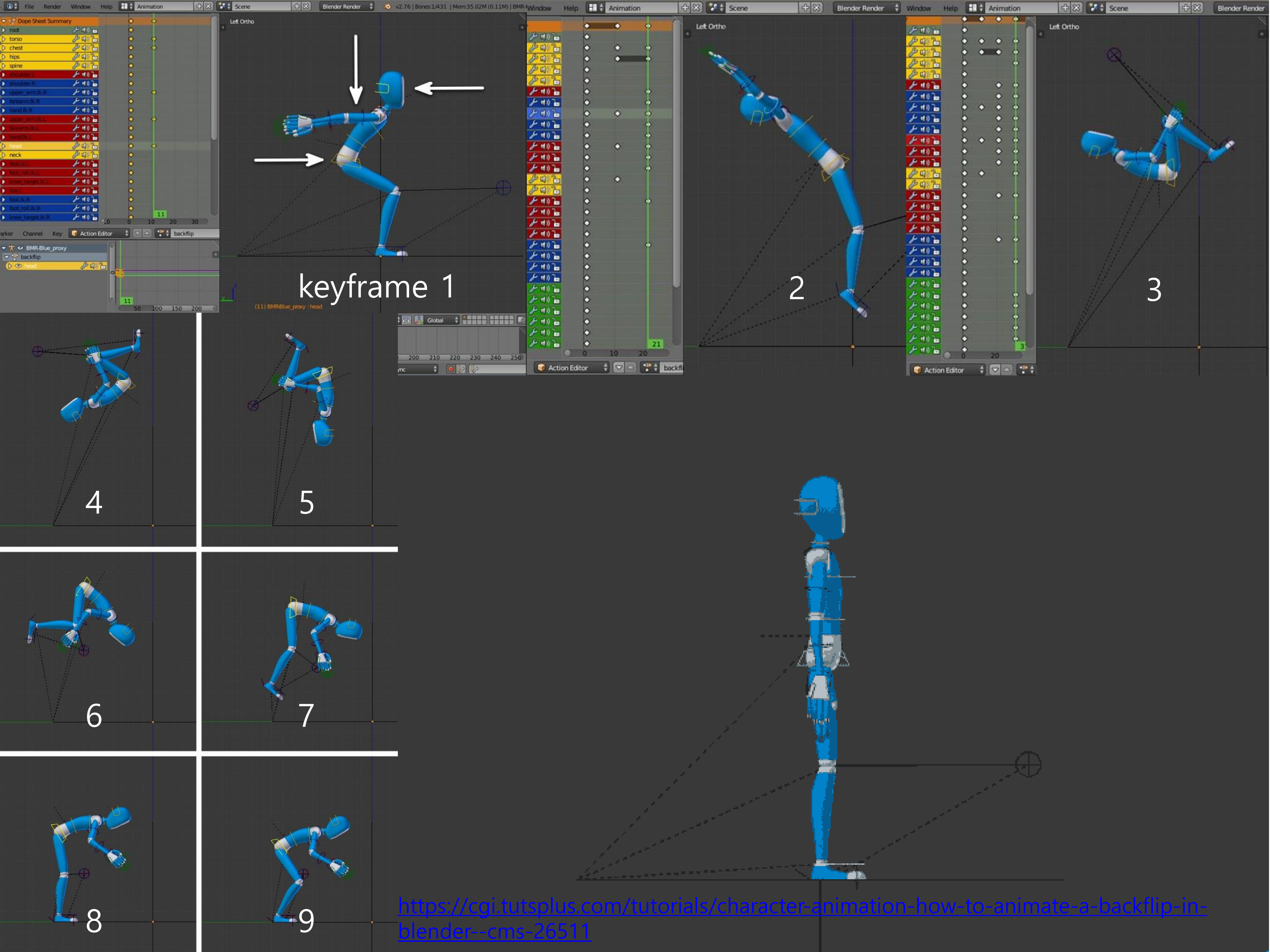
Keyframe Animation

- **Basic idea:**
  - specify important events only
  - computer fills in the rest via interpolation/approximation
- **"Events" don't have to be position**
- **Could be color, light intensity, camera zoom, ...**



Keyframe Animation

- For example, for positions and orientations:
- Affine transformations place things *at keyframes*.
- Time-varying affine transformations move things around by **interpolation** *at in-between frames*.
- How to interpolate affine transformations?
  - We'll address this issue later in the lecture today.



https://cgi.tutsplus.com/tutorials/character-animation-how-to-animate-a-backflip-in-blender--cms-26511

Keyframe Animation

- One of the earliest methods used to produce computer animation.

- Difficult to create “realistic” and “physically plausible” motions.
  - The quality of the output largely depends on the skill of each artist.

- Still used a lot.

Motion Capture

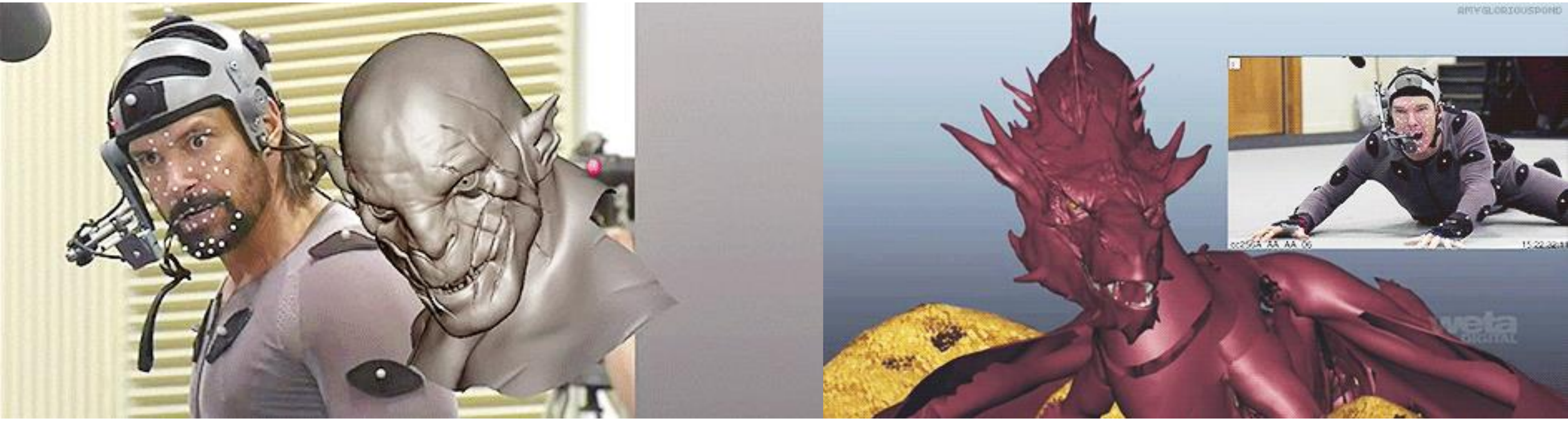
- Idea: Use “real” human motion to create realistic animation.

- Motion capture system “captures” movement of people or objects.
  - Position of each marker on the skin
  - Position and orientation of each body part (or joint)

An Example of Optical Passive Motion Capture System

Dawn of the Planet of the Apes

The Last Of Us



The Hobbit

Motion Capture

- Currently, widely used in movies & games
  - by major companies
- Very costly
  - Expensive devices
  - High operating cost
- Motion captured data is very realistic **only** in the same virtual environment as capture environment.
  - What if a character is affected by unexpected external force? Capture again?

Physically-Based Simulation

- Idea: Use physics simulation to generate motion
  - Because physical reality plays a key role in creating high-quality motion.
  - Resulting motion would be always physically plausible.

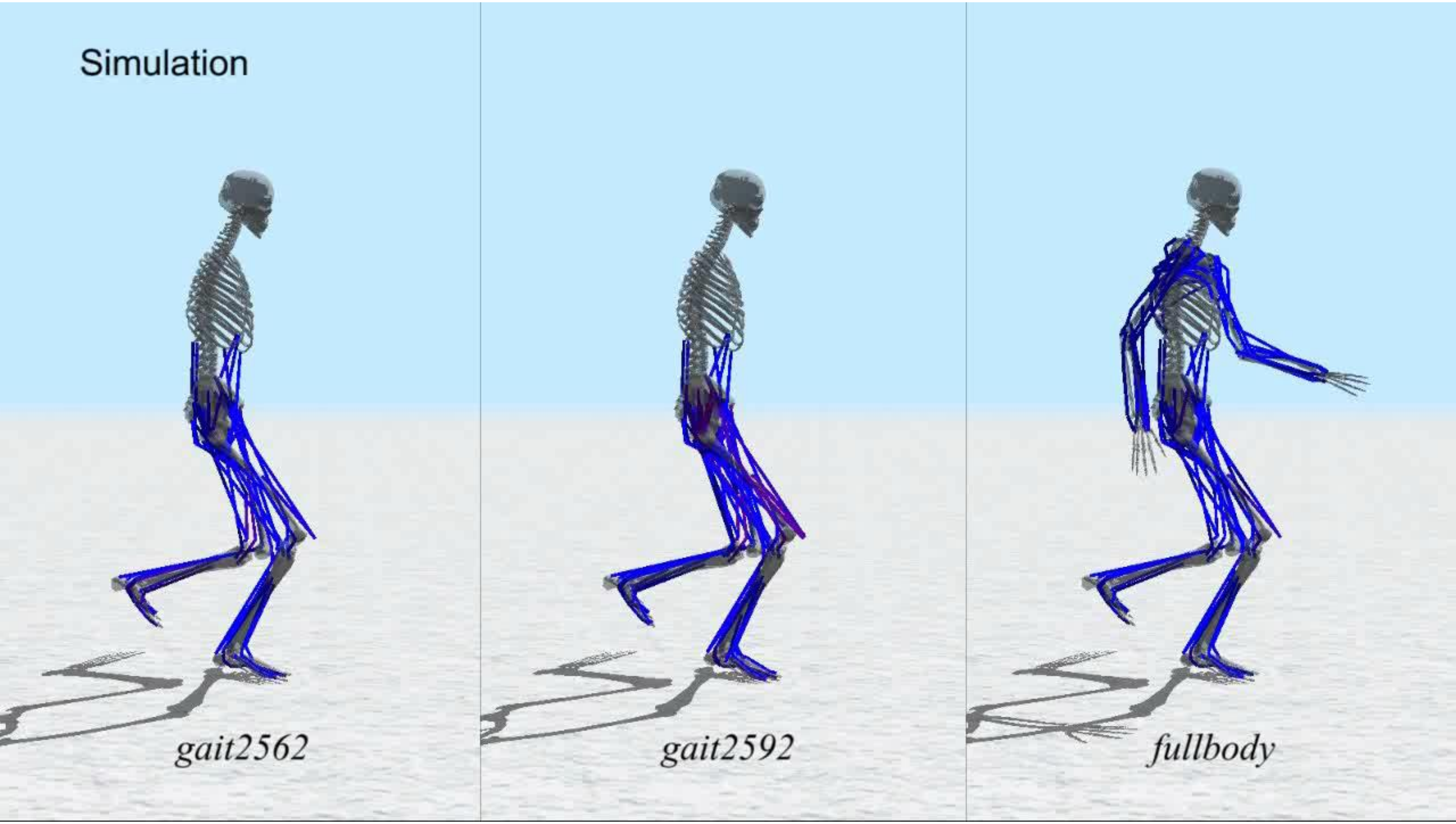
- For active character motion, it’s currently being very actively studied by researchers.

- Not easy to find what joint torques are required to make desired output motion.
  - Maintaining balance is challenging.
  - This problem is similar to that of robotics.

[Lee et al. 2010]

https://youtu.be/hpeqxc_1vwo

[Lee et al. 2014]



https://youtu.be/O0zxRbZ_kNM

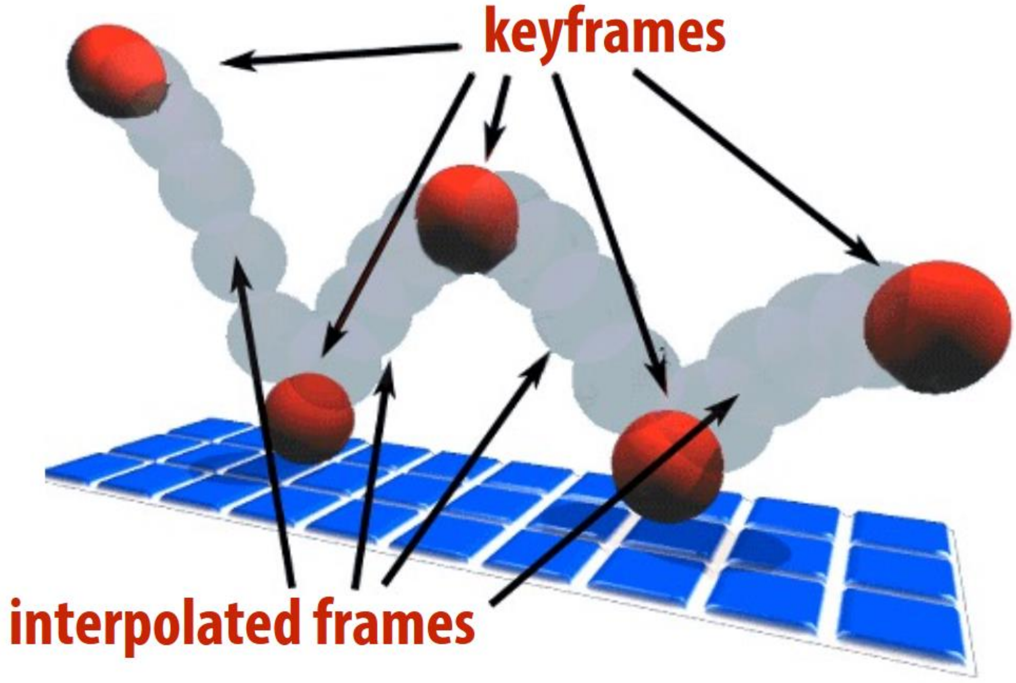
Physically-Based Simulation

- Promising approaches:
  - Combining with machine learning techniques (such as deep reinforcement learning)
  - Biomechanical simulation of musculoskeletal models
  - Control real-world robots

- A notable achievement can have a significant impact not only on computer graphics but also on related areas such as robotics and biomechanics!

Interpolation

Recall: Keyframe Animation



CMU 15-462/662, Fall 2015

- How to “interpolate” keyframes?

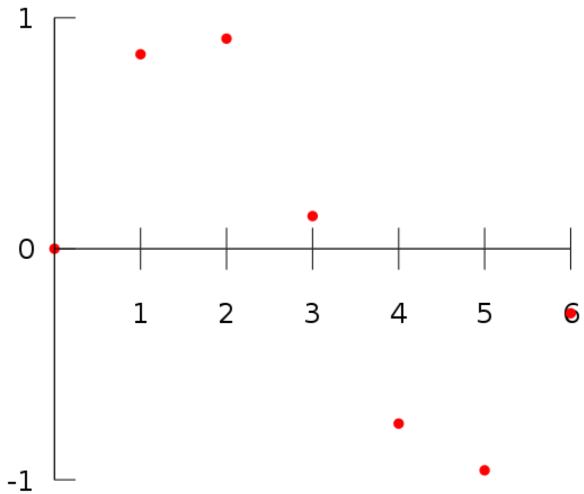
Interpolation

- A method of constructing new data points within the range of a discrete set of known data points.

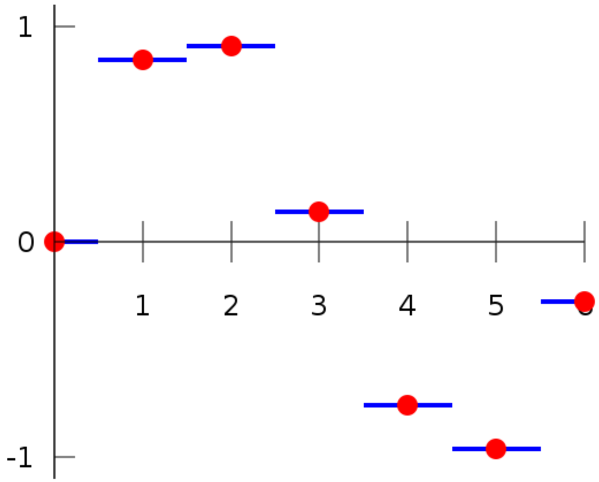
- In other words, guessing unknown function f(x) from known data points $(x_i, f(x_i))$.

Ex) Known data points

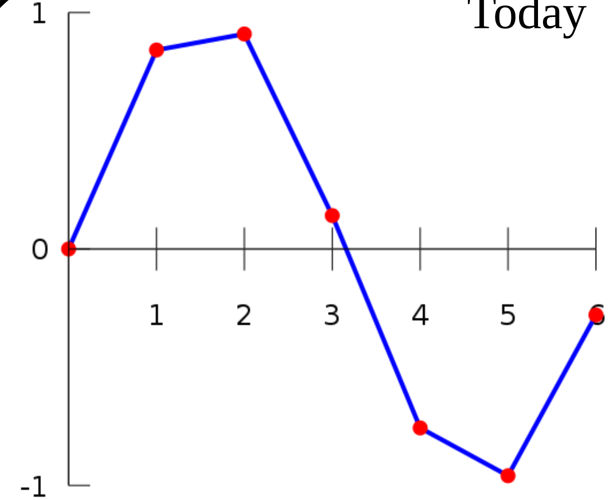
x	f(x)
0	0
1	0.8415
2	0.9093
3	0.1411
4	-0.7568
5	-0.9589
6	-0.2794



Today

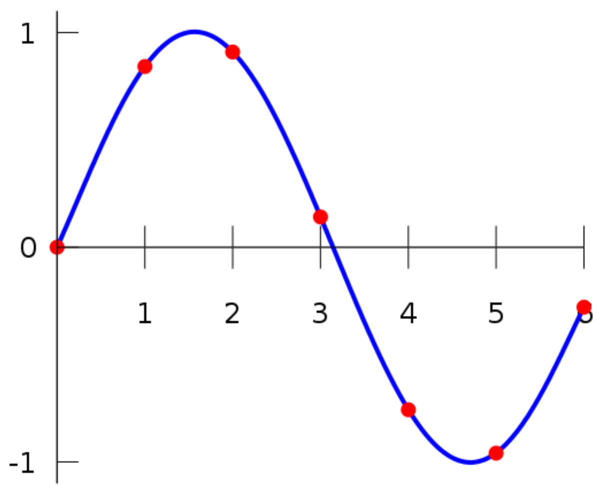


nearest-neighbor interpolation

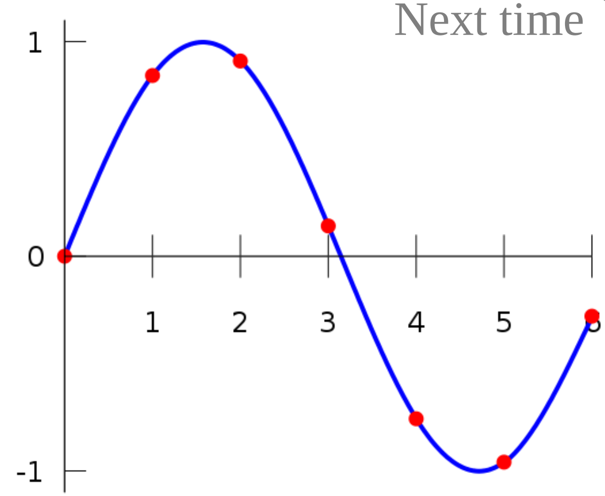


linear interpolation

Next time

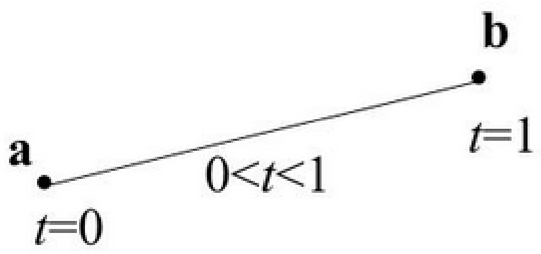


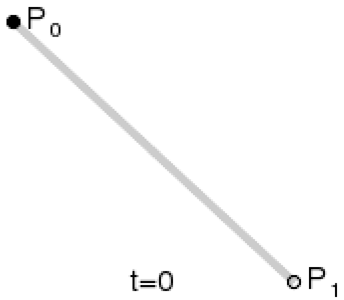
polynomial interpolation



spline interpolation

Linear Interpolation





https://upload.wikimedia.org/wikipedia/commons/0/00/B%C3%A9zier_1_big.gif

$$\operatorname{lerp}(\mathbf{a}, \mathbf{b}, t) = (1 - t)\mathbf{a} + t\mathbf{b}$$

```
float lerp(float v0, float v1, float t)
{
  return (1 - t) * v0 + t * v1;
}
```

- A straight line between two points
- This is fine for translations

Linear Interpolation for 3D Orientations?

- Recall: 3D orientation & rotation representation
  - Euler angles
  - Axis-angle (Rotation vector)
  - Rotation matrices
  - Unit quaternions

- How to linearly interpolate **orientations** in these representations?

Interpolating Each Element of Rotation Matrix?

- Let's try to interpolate $\mathbf{R}_0$(identity) and $\mathbf{R}_1$(rotation by 90° about x-axis)

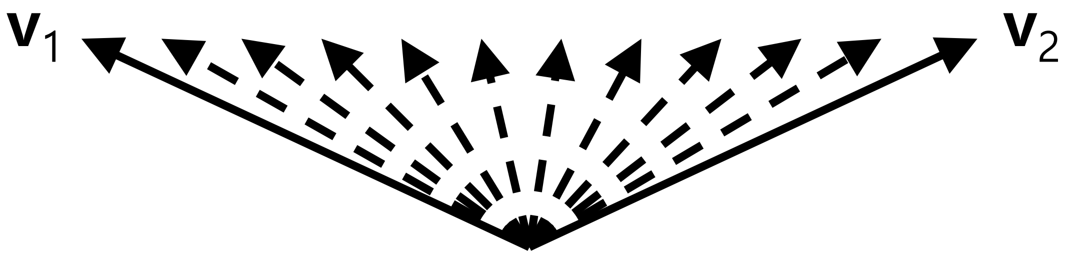
$$\mathrm{lerp}(\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}, \begin{bmatrix}1 & 0 & 0\\0 & 0 & -1\\0 & 1 & 0\end{bmatrix}, 0.5) = \begin{bmatrix}1 & 0 & 0\\0 & 0.5 & -0.5\\0 & 0.5 & 0.5\end{bmatrix}$$

is not a rotation matrix!
does not make sense at all!

- Similarly, interpolating each number (w, x, y, z) in ⊏ unit quaternions does not make sense.

Interpolating Axis-Angle (Rotation Vector)?

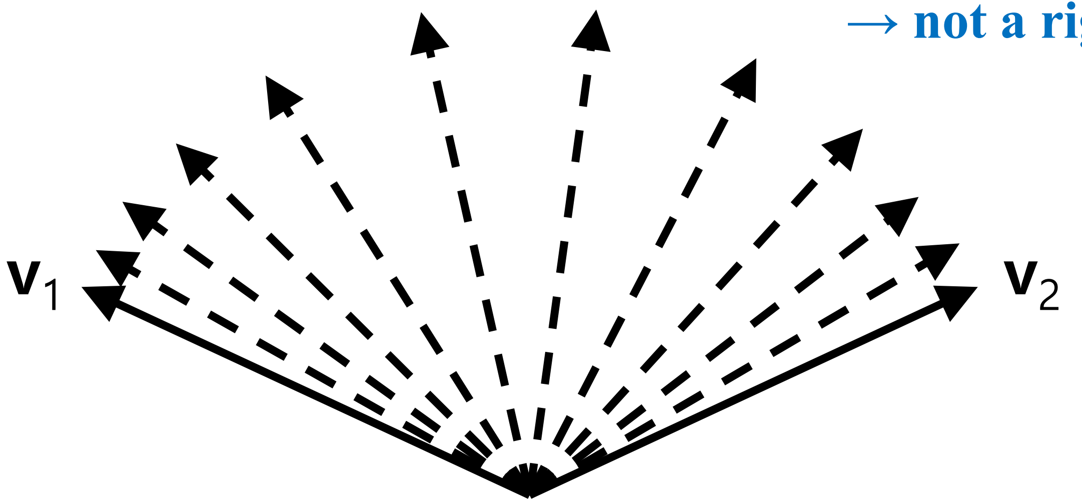
- Let's say we have two rotation vectors $\mathbf{v}_1$ & $\mathbf{v}_2$ of the same length
- Linear interpolation of $\mathbf{v}_1$ & $\mathbf{v}_2$ produces even spacing



Interpolating Axis-Angle (Rotation Vector)?

- Let's say we have two rotation vectors $\mathbf{v}_1$ & $\mathbf{v}_2$ of the same length
- Linear interpolation of $\mathbf{v}_1$ & $\mathbf{v}_2$ produces even spacing
- But it's not evenly spaced in terms of orientation!

→ **not a right method!**

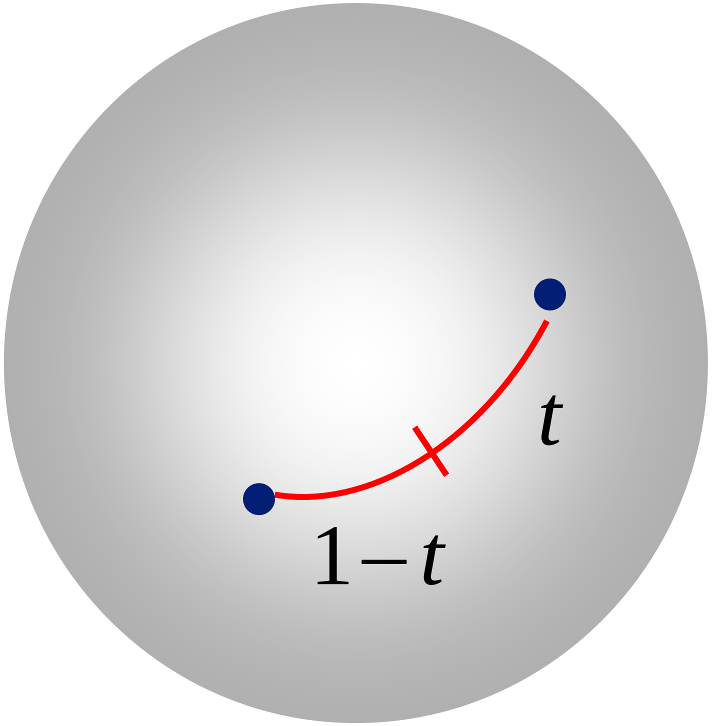


Interpolating Euler Angles?

- Interpolating two tuples of Euler angles does not make correct result
  - + angular velocity is not constant
  - + still suffer from gimbal lock: jerky movement occurs near gimbal lock configuration

Slerp

- The right answer: **Slerp** [Shoemake 1985]
  - **S**pherical **l**inear int**erp**olation
  - Linear interpolation of two orientations



"t" refers power, not transpose

$$\mathrm{slerp}(\mathbf{R}_1, \mathbf{R}_2, t) = \mathbf{R}_1(\mathbf{R}_1^T \mathbf{R}_2)^t$$
$$= \mathbf{R}_1 \exp(\mathrm{t} \cdot \log(\mathbf{R}_1^T \mathbf{R}_2))$$

Slerp

$$\mathrm{slerp}(\mathbf{R}_1, \mathbf{R}_2, t) = \mathbf{R}_1(\mathbf{R}_1^T \mathbf{R}_2)^t$$
$$= \mathbf{R}_1 \exp(\mathrm{t} \cdot \log(\mathbf{R}_1^T \mathbf{R}_2))$$

- exp(): axis-angle (rotation vector) to rotation matrix
- log(): rotation matrix to axis-angle (rotation vector)

- Implication
  - $\mathbf{R}_1^{T}\mathbf{R}_2$: difference between orientation $\mathbf{R}_1$ and $\mathbf{R}_2$ ($\mathbf{R}_2(-)\mathbf{R}_1$)
  - $\mathbf{R}^{t}$: scaling rotation (scaling rotation angle)
  - $\mathbf{R}_a\mathbf{R}_b$: add rotation $\mathbf{R}_b$ to orientation $\mathbf{R}_a$ ($\mathbf{R}_a(+)\mathbf{R}_b$)

Exp & Log

- **Exp** (exponential): axis-angle(rotation vector) to rotation matrix
  - Given normalized rotation axis u=(u_x,u_y,u_z), rotation angle θ

$$R = \begin{bmatrix} \cos\theta + u_x^2\left(1-\cos\theta\right) & u_x u_y\left(1-\cos\theta\right) - u_z\sin\theta & u_x u_z\left(1-\cos\theta\right) + u_y\sin\theta \\ u_y u_x\left(1-\cos\theta\right) + u_z\sin\theta & \cos\theta + u_y^2\left(1-\cos\theta\right) & u_y u_z\left(1-\cos\theta\right) - u_x\sin\theta \\ u_z u_x\left(1-\cos\theta\right) - u_y\sin\theta & u_z u_y\left(1-\cos\theta\right) + u_x\sin\theta & \cos\theta + u_z^2\left(1-\cos\theta\right) \end{bmatrix}$$

(Rodrigues' rotation formula)

- **Log** (logarithm): rotation matrix to axis-angle(rotation vector)

Given rotation matrix **R**, compute axis **v** and angle θ

$$\theta = \cos^{-1}((R_{11} + R_{22} + R_{33} - 1)/2)$$

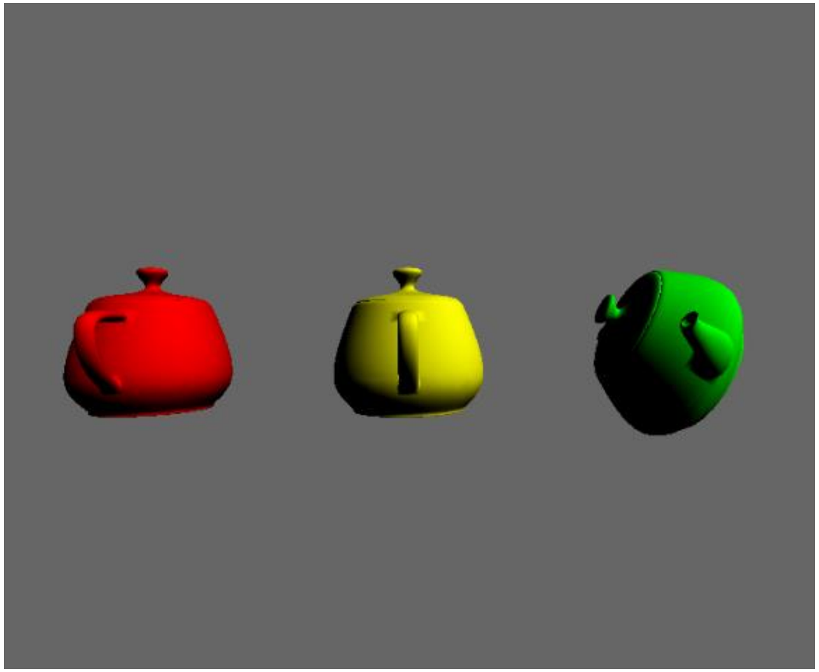
$$v_1 = (R_{32} - R_{23})/(2\sin\theta)$$

$$v_2 = (R_{13} - R_{31})/(2\sin\theta)$$

$$v_3 = (R_{21} - R_{12})/(2\sin\theta)$$

See section 3.1.3 of INTRODUCTION TO ROBOTICS for more info about matrix exp & log:
http://robotics.snu.ac.kr/fcp/files/_pdf_files_publications/a_first_coruse_in_robot_mechanics.pdf

[Practice] Slerp Online Demo



https://nccastaff.bournemouth.ac.uk/jmacey/WebGL/QuatSlerp/

- Change “Start Rotation” & “End Rotation”
- Move “Interpolate” slider

Quiz #1

- Go to https://www.slido.com/
- Join #cg-hyu
- Click “Polls”

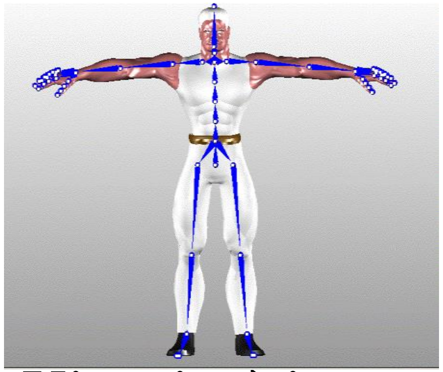
- Submit your answer in the following format:
  - **Student ID: Your answer**
  - **e.g. 2017123456: 4)**

- Note that you must submit all quiz answers in the above format to be checked for “attendance”.

Kinematics

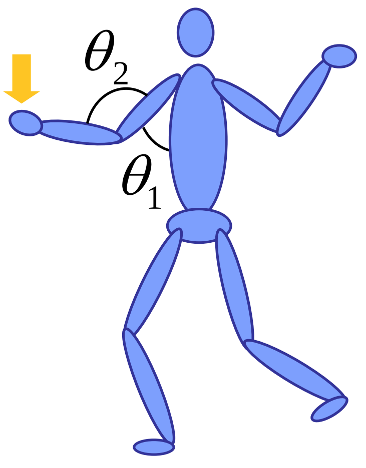
Kinematics

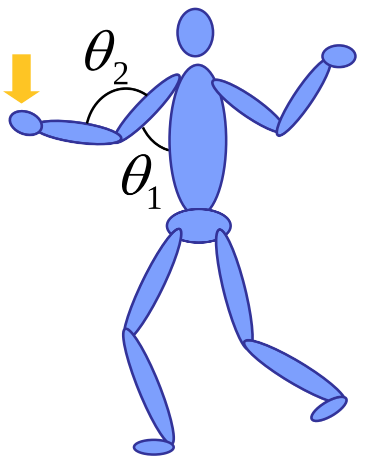
- *Kinematics* is
  - Study of **motion** of objects (or groups of objects), **without considering mass or forces**
  - In computer graphics, it’s about how to move skeletons
    - Forward kinematics
    - Inverse kinematics



- By contrast, *Dynamics (or Kinetics)* is
  - Study of the **relationship between motion and its causes, specifically, forces and mass**

Kinematics





$$(\mathbf{p}, \mathbf{q}) = \mathrm{F}(\theta_i)$$

Forward Kinematics

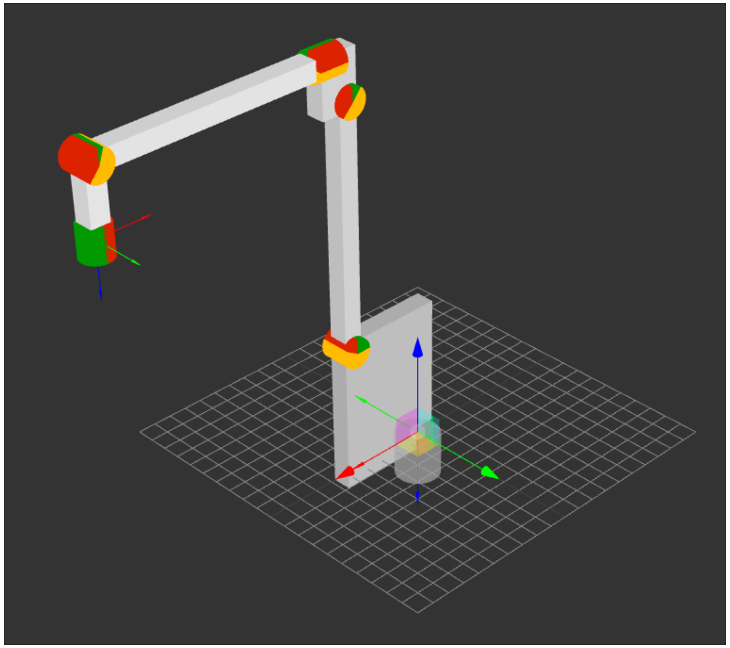
: Given joint angles,
compute the position &
orientation of end-effector

$$\theta_i = \mathrm{F}^{-1}(\mathbf{p}, \mathbf{q})$$

Inverse Kinematics

: Given the position &
orientation of end-effector,
compute joint angles

[Practice] FK / IK Online Demo

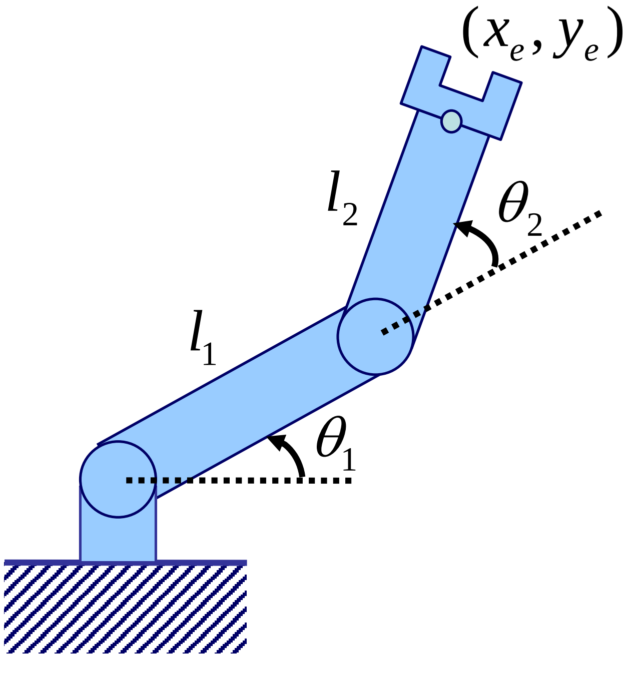


http://robot.glumb.de/

- Forward kinematics : Open “angles” menu and change values
- Inverse kinematics : Move the end-effector position by mouse dragging

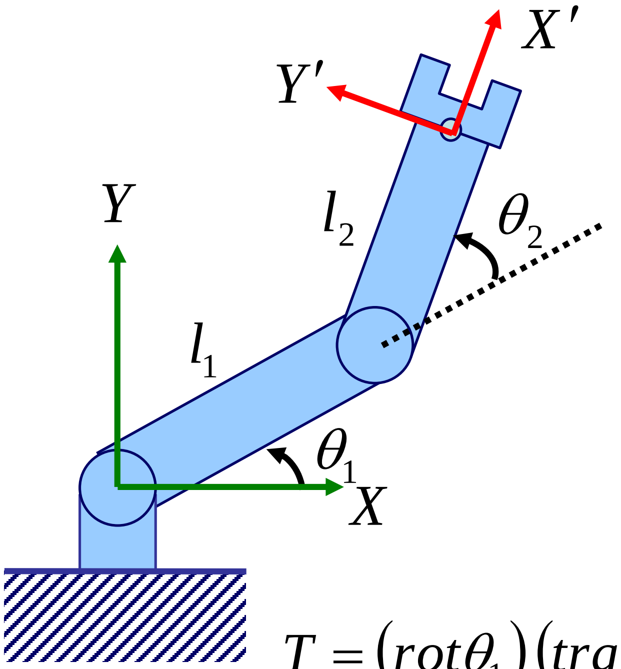
Forward Kinematics: A Simple Example

- A simple robot arm in 2-dimensional space
  - 2 revolute joints
  - Joint angles are known
  - Compute the position of the end-effector



$$x_e = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2)$$
$$y_e = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2)$$

A Chain of Transformations



$$\begin{pmatrix} x_e \\ y_e \\ 1 \end{pmatrix} = \begin{pmatrix} & & \\ & T & \\ & & \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

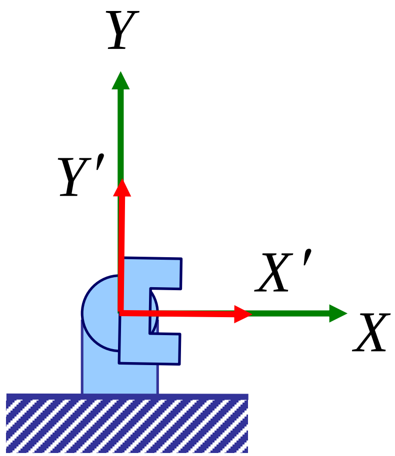
$$T = (rot\theta_1)(transl_1)(rot\theta_2)(transl_2)$$

$$= \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Thinking of Transformations

- In a view of body-attached coordinate system

(=local coordinate system of the end-effector body)

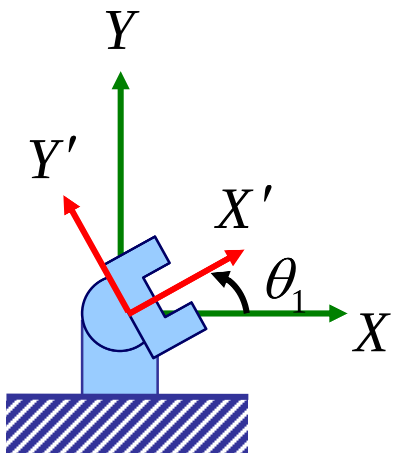


$$T = (rot\theta_1)(transl_1)(rot\theta_2)(transl_2)$$

$$= \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Thinking of Transformations

- In a view of body-attached coordinate system

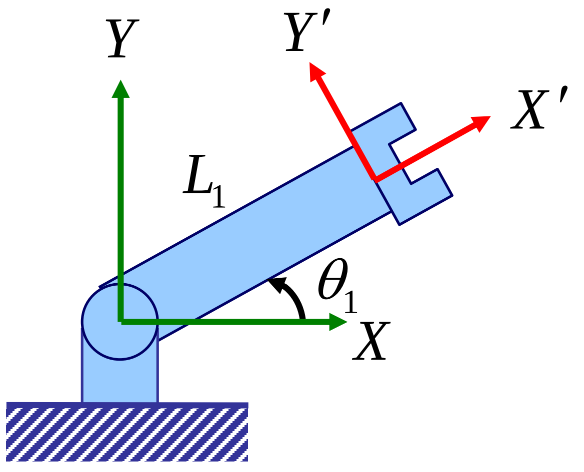


$$T = \boxed{(rot\theta_1)}(transl_1)(rot\theta_2)(transl_2)$$

$$= \boxed{\begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix}} \begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Thinking of Transformations

- In a view of body-attached coordinate system

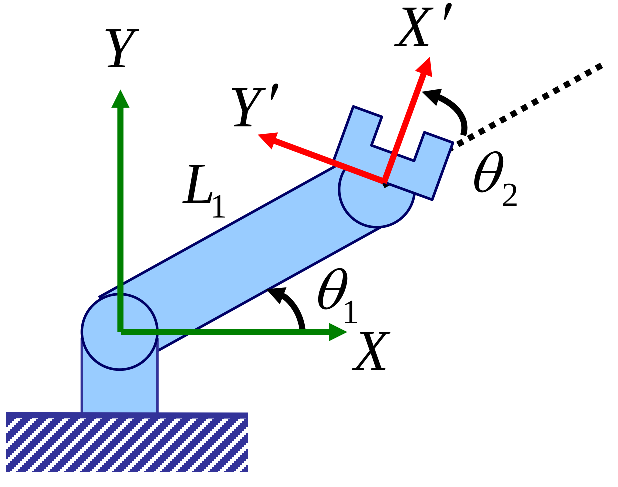


$$T = \boxed{(rot\theta_1)(transl_1)}(rot\theta_2)(transl_2)$$

$$= \boxed{\begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}\begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Thinking of Transformations

- In a view of body-attached coordinate system

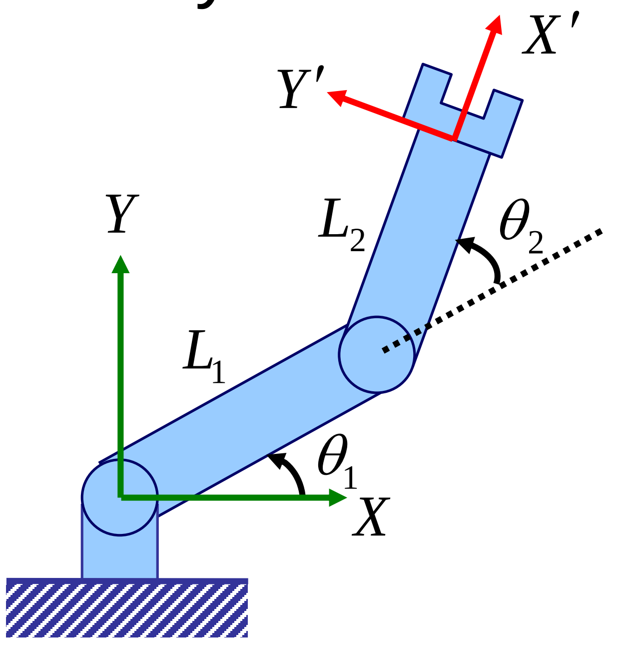


$$T = \boxed{(rot\theta_1)(transl_1)(rot\theta_2)}(transl_2)$$

$$= \boxed{\begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix}} \begin{pmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Thinking of Transformations

- In a view of body-attached coordinate system

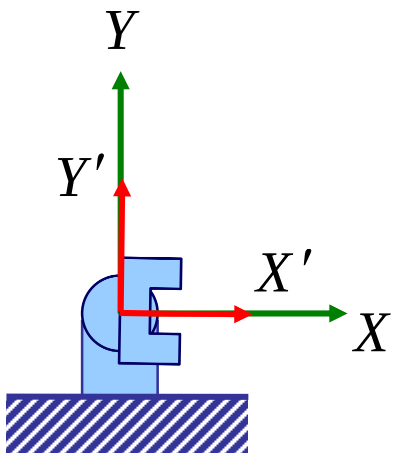


$$T = \boxed{(rot\theta_1)(transl_1)(rot\theta_2)(transl_2)}$$

$$= \boxed{\begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}$$

Thinking of Transformations

- In a view of global coordinate system

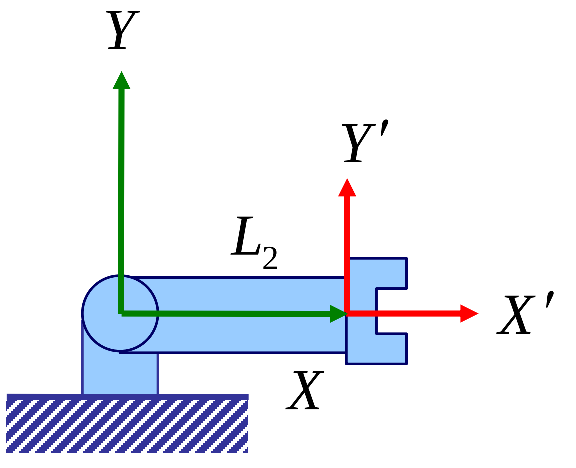


$$T = (rot\theta_1)(transl_1)(rot\theta_2)(transl_2)$$

$$= \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Thinking of Transformations

- In a view of global coordinate system

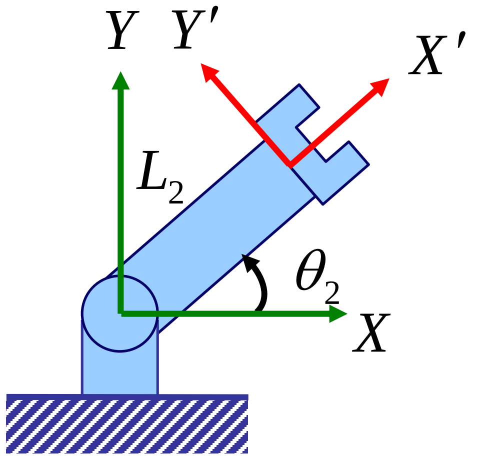


$$T = (rot\theta_1)(transl_1)(rot\theta_2)\boxed{(transl_2)}$$

$$= \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \boxed{\begin{pmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}$$

Thinking of Transformations

- In a view of global coordinate system

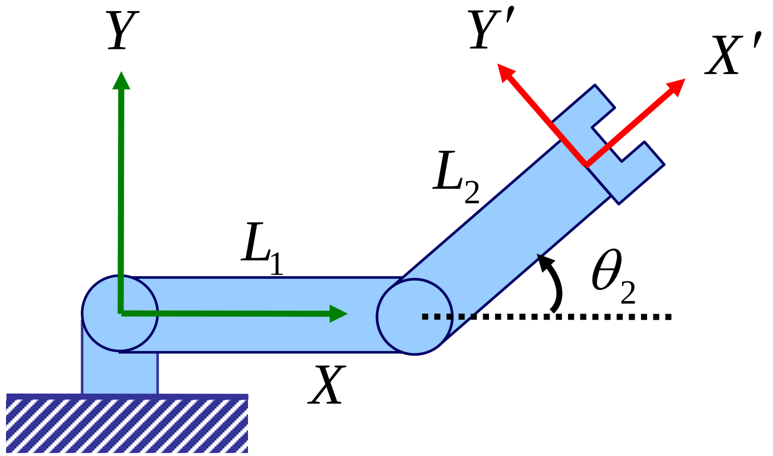


$$T = (rot\theta_1)(transl_1)\boxed{(rot\theta_2)(transl_2)}$$

$$= \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \boxed{\begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}$$

Thinking of Transformations

- In a view of global coordinate system

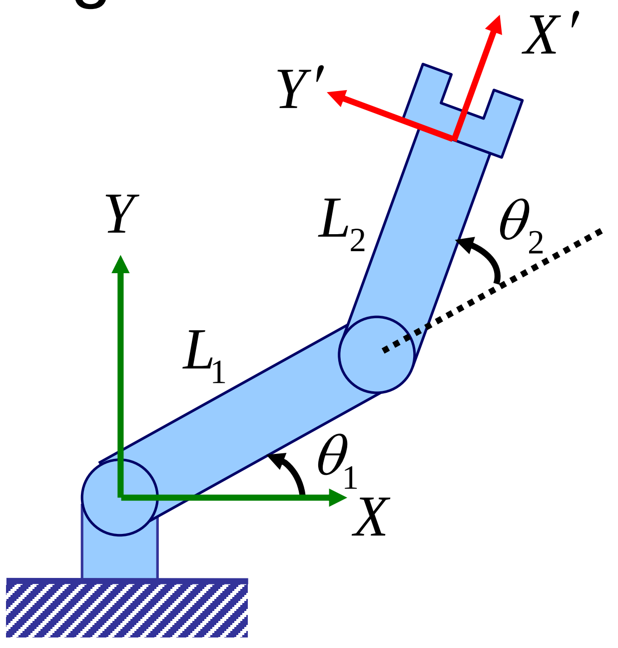


$$T = (rot\theta_1)\boxed{(transl_1)(rot\theta_2)(transl_2)}$$

$$= \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \boxed{\begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}$$

Thinking of Transformations

- In a view of global coordinate system



$$T = \boxed{(rot\theta_1)(transl_1)(rot\theta_2)(transl_2)}$$

$$= \boxed{\begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}$$

Quiz #2

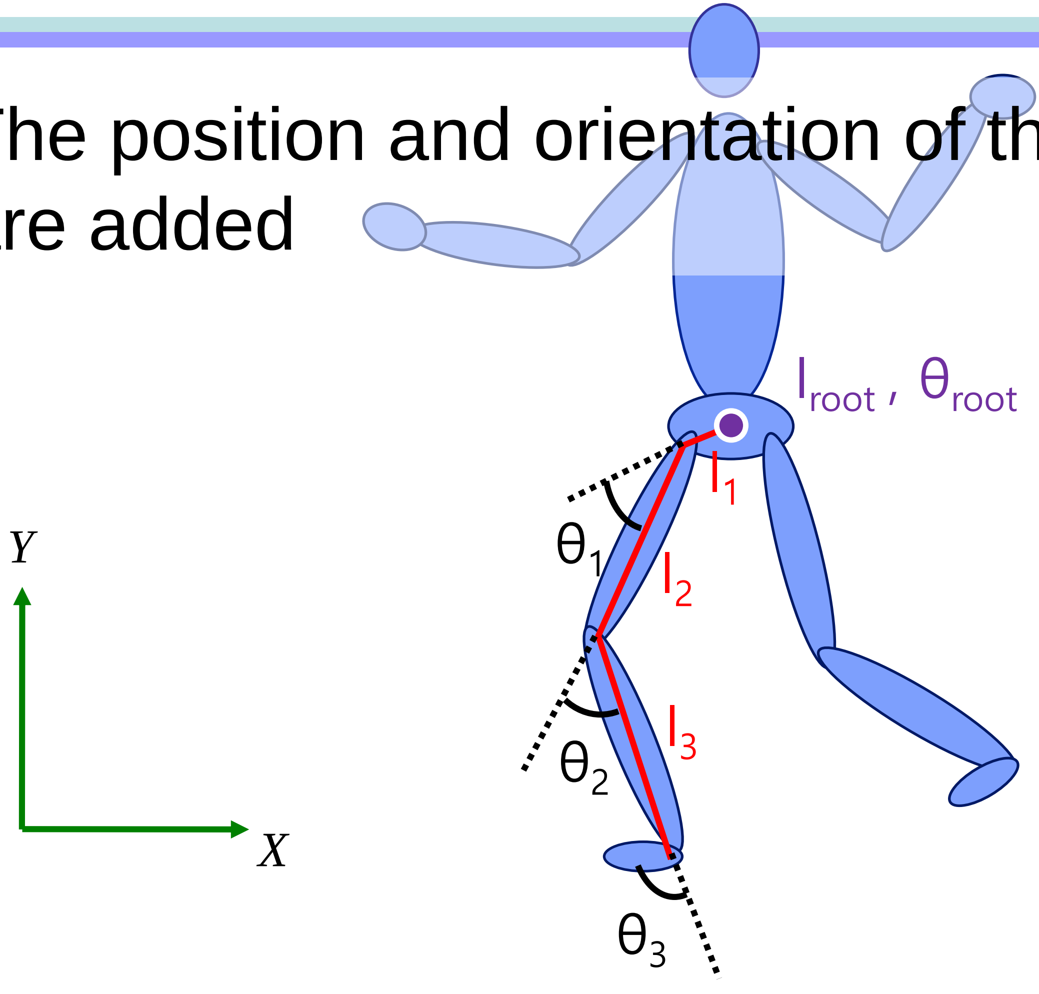
- Go to https://www.slido.com/
- Join #cg-hyu
- Click “Polls”

- Submit your answer in the following format:
  - **Student ID: Your answer**
  - **e.g. 2017123456: 4)**

- Note that you must submit all quiz answers in the above format to be checked for “attendance”.

Floating Base

- The position and orientation of the root segment are added



$$T_2 = \boxed{(transl_{root})(rot\theta_{root})}(transl_1)(rot\theta_1)(transl_2)(rot\theta_2)$$

Motion Data

- Contains two types of data:

- Time-varying data
  - Joint angles
  - “Motion”

- Static data
  - Bone lengths (parent-child joint relationship without any motion)
  - “Structure”

Joint & Link Transformations

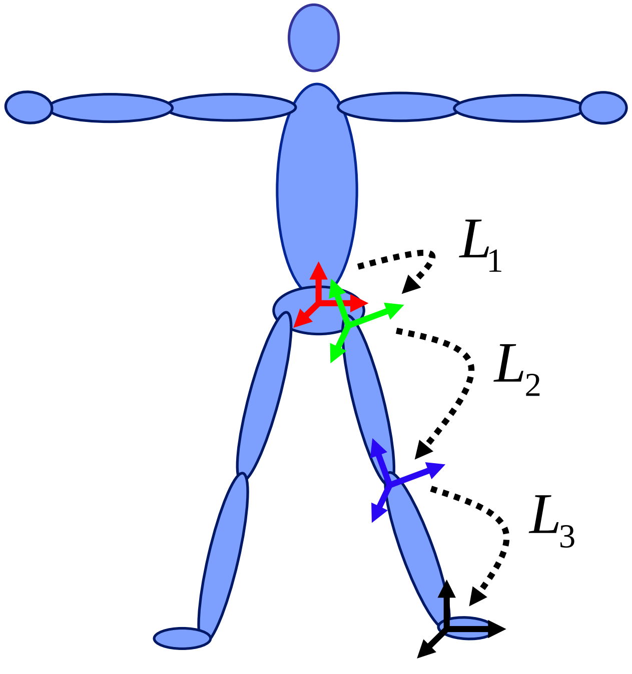
- Forward kinematics map is an alternating multiple of
  - ***Joint transformations*** : represents joint movement **(time-varying)**
    - Usually rotations
  - ***Link transformations*** : defines a frame relative to its parent **(static)**
    - Usually translations (bone-length)

1st link transformation

$$T = J_0 L_1 J_1 L_2 J_2 L_3 J_3$$

1st joint transformation

The position and orientation of the root segment



Quiz #3

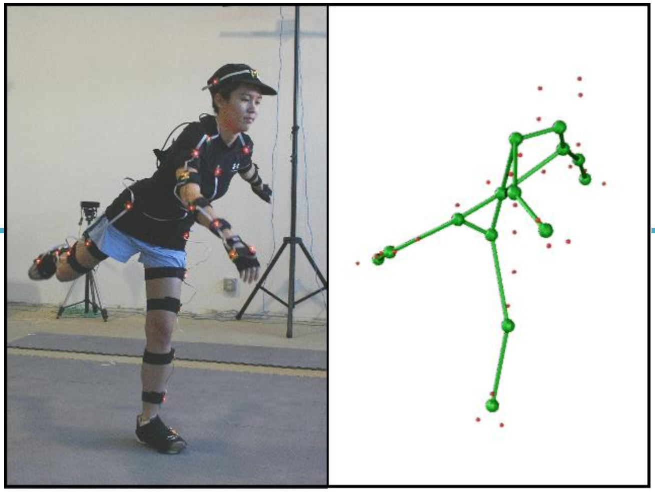
- Go to https://www.slido.com/
- Join #cg-hyu
- Click “Polls”

- Submit your answer in the following format:
  - **Student ID: Your answer**
  - **e.g. 2017123456: 4)**

- Note that you must submit all quiz answers in the above format to be checked for “attendance”.

BVH File Format

BVH File Format

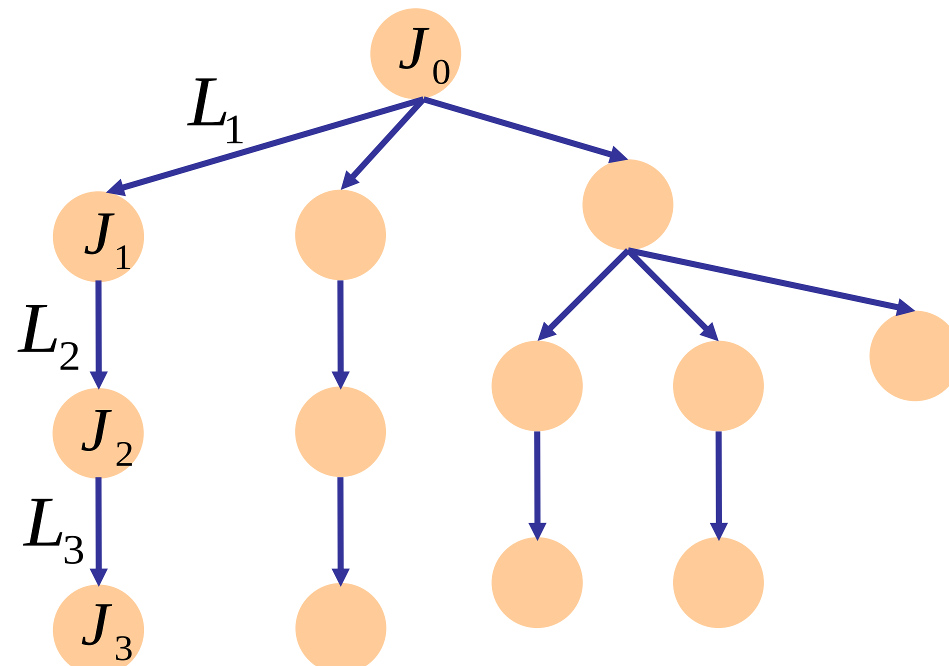


- BVH (**B**io**V**ision **H**ierarchical data)
- Developed by Biovision, a motion capture company
- Has two parts:
- **Hierarchy section**
  - Describes the hierarchy and initial pose of the skeleton
- **Motion section**
  - Contains motion data
- Text file format

Hierarchy Section

- The hierarchy is a joint tree.
- Each joint has an offset and a channel list.

- Joint 1:
  - Offset: L1
  - Channel list :
    a sequence
    of transformations
    of J1 w.r.t. J0



Biovision BVH

- **Hierarchy section**
  - **"HIERARCHY"**
    - **"ROOT"**
      - followed by the name of the root
      - **" { "** and **" } "** pair
      - **"OFFSET"**
        - » X,Y and Z offset of the segment from its parent
      - **"CHANNELS"**
        - » the number of channels
        - » the type of each channel
    - **"JOINT"**
      - identical to the root definition except for the number of channels
      - **"OFFSET " , "CHANNELS"**
    - **"End Site"**
      - **indicates that the current segment is an end effector** (no children)
      - **"OFFSET "**

```
HIERARCHY
ROOT Hips
{
    OFFSET  0.00    0.00    0.00
    CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation Yrotation
    JOINT LeftHip
    {
        OFFSET   3.430000        0.000000   0.000000
        CHANNELS 3 Zrotation Xrotation Yrotation
        JOINT LeftKnee
        {
            OFFSET  0.000000  -18.469999  0.000000
            CHANNELS 3 Zrotation Xrotation Yrotation
            JOINT LeftAnkle
            {
                OFFSET      0.000000   -17.950001  0.000000
                CHANNELS 3 Zrotation Xrotation Yrotation
                End Site
                {
                    OFFSET      0.000000  -3.119999  0.000000
                }
            }
        }
    }
...
}
```

- 6 channels for the root (Tx Ty Tz Rz Rx Ry)

- 3 channels for every other object (Rz Rx Ry)

```
HIERARCHY
ROOT Hips
{
 OFFSET 0.0 0.0 0.0
 CHANNELS 6 XPOSITION YPOSITION ZPOSITION ZROTATION XROTATION YROTAT
    JOINT chest
    {
            OFFSET  0.096536 3.475309 -0.289609
            CHANNELS 3 Xrotation Zrotation Yrotation
            JOINT neck
            {
                    OFFSET  -0.096536 13.901242 -2.027265
                    CHANNELS 3 Xrotation Zrotation Yrotation
                    JOINT head
                    {
                            OFFSET  -0.166775 1.448045 0.482682
                            CHANNELS 3 Xrotation Zrotation Yrotation
                            JOINT leftEye
```

HIERARCHY
ROOT Hips
{
OFFSET 0.0 0.0 0.0 J0 channels
CHANNELS 6 XPOSITION YPOSITION ZPOSITION ZROTATION XROTATION YROTA

```
JOINT chest
{
        OFFSET  0.096536 3.475309 -0.289609
        CHANNELS 3 Xrotation Zrotation Yrotation
        JOINT neck
        {
                OFFSET  -0.096536 13.901242 -2.027265
                CHANNELS 3 Xrotation Zrotation Yrotation
                JOINT head
                {
                        OFFSET  -0.166775 1.448045 0.482682
                        CHANNELS 3 Xrotation Zrotation Yrotation
                        JOINT leftEye
```

L1
J1 channels
L2
J2 channels
L3
J3 channels

```
HIERARCHY
ROOT Hips
{
  OFFSET 0.0 0.0 0.0
  CHANNELS 6 XPOSITION YPOSITION ZPOSITION ZROTATION XROTATION YROTAT
    JOINT chest
    {
            OFFSET  0.096536 3.475309 -0.289609
            CHANNELS 3 Xrotation Zrotation Yrotation
            JOINT neck
            {
                    OFFSET  -0.096536 13.901242 -2.027265
                    CHANNELS 3 Xrotation Zrotation Yrotation
                    JOINT head
                    {
                            OFFSET  -0.166775 1.448045 0.482682
                            CHANNELS 3 Xrotation Zrotation Yrotation
                            JOINT leftEye
```

Root Hips Joint

Root offset is generally zero (or ignored even if it's not zero)

Chest Joint

Neck Joint

Neck's offset from chest

Channel list:
Transformation from chest coordinate system to neck coordinate system

Biovision BVH

```
MOTION
Frames:     20
Frame Time: 0.033333
 0.00  39.68   0.00    0.65   ...
...
```

■ Motion Section

- **"MOTION"**
  - followed by a line indicating the number of frames
  - **"Frames:"**
    - the number of frames
  - **"Frame Time:"**
    - the sampling rate of the data
    - Ex) 0.033333 ➔ 30 frames a second
  - The rest of the file contains the actual motion data
    - The numbers appear in the order of the channel specifications as the skeleton hierarchy was parsed
  - **Each line has motion data for a single frame**
  - **Each number in a line is a value for a single channel**
  - **The unit of rotation channel values is degree**

```
HIERARCHY
ROOT Hips
{
  OFFSET 0.0 0.0 0.0
  CHANNELS 6 XPOSITION YPOSITION ZPOSITION ZROTATION XROTATION YROTATION
    JOINT chest  Column 1     Column 2     Column 3     Column 4     Column 5     Column 6
    {
            OFFSET  0.096536 3.475309 -0.289609
            CHANNELS 3 Xrotation Zrotation Yrotation
            JOINT neck    Column 7     Column 8     Column 9
            {
                    OFFSET  -0.096536 13.901242 -2.027265
                    CHANNELS 3 Xrotation Zrotation Yrotation
                    JOINT head    Column 10    Column 11    Column 12
                    {
                            OFFSET  -0.166775 1.448045 0.482682
                            CHANNELS 3 Xrotation Zrotation Yrotation
                                      Column 13    Column 14    Column 15
MOTION
Frames: 199
Frame Time: 0.033333
1.95769 0.989769479321 0.039193 -4.11275998891 -0.490682977769 -91.3519974695 0.45458697547  …
1.95769 0.989769479321 0.0392908 -4.11760985011 -0.48626597981 -91.3734989051 0.513819016282  …
1.95769 0.989769479321 0.039424 -4.12004011679 -0.488125979059 -91.387002189 0.592700017233  …
1.95771 0.989769479321 0.0395518 -4.0961698863 -0.500940000911 -91.3840993586 0.61126399115  …
1.95779 0.989759479321 0.0396999 -4.05799980101 -0.510696019006 -91.3839969058 0.58299101005  …
1.9579 0.989719479321 0.0398625 -4.0423300664 -0.503295989288 -91.3842018115 0.57718001317  …
…
```

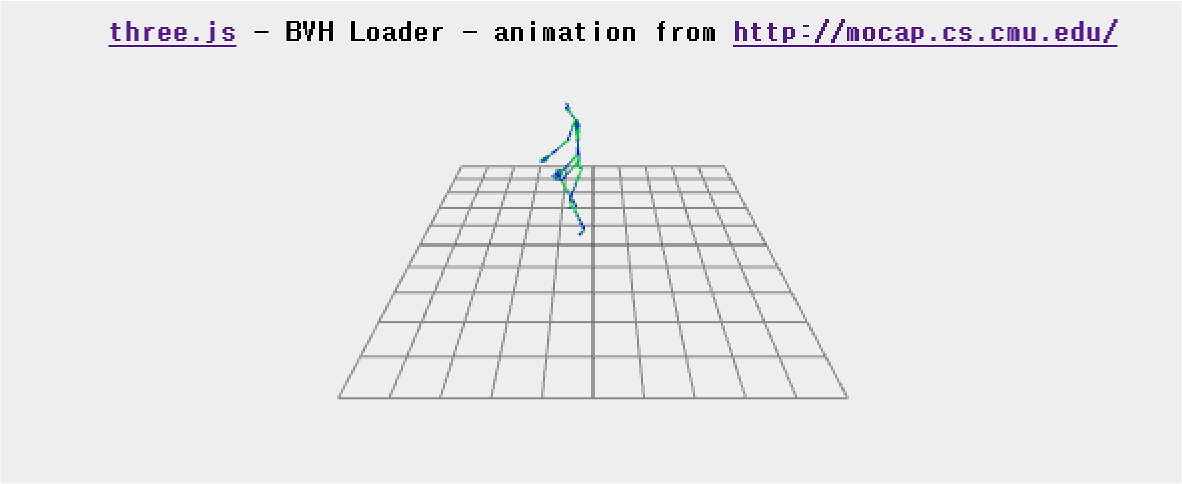
Biovision BVH

Interpreting the data

- To calculate the position of a segment
  - Translation information
    - For any joint segment
      - the translation information will simply be the offset as defined in the hierarchy section
    - For the root object
      - The translation data will be the sum of the offset data and the translation data from the motion section
  - Rotation information
    - comes from the motion section
  - **If the order is "ZROTATION XROTATION YROTATION"**
    - **Apply transformation in order of rotation about z, rotation about x, rotation about y w.r.t. local frame**
    - **→ ZXY Euler angles**

  - **Usually ZXY order is used, but other orders can be used**

[Practice] BVH Online Demo



http://motion.hahasoha.net/

- Select other motions from the list.
- Download corresponding BVH files and open them in a text editor.

Next Time

- Lab in this week:
  - Lab assignment 10

- Next lecture:
  - 11 - Curve

- **Class Assignment #3**
  - **Due: 23:59, June 10, 2019**

- Acknowledgement: Some materials come from the lecture slides of
  - Prof. Kayvon Fatahalian and Prof. Keenan Crane, CMU, http://15462.courses.cs.cmu.edu/fall2015/
  - Prof. Jinxiang Chai, Texas A&M Univ., http://faculty.cs.tamu.edu/jchai/csce441_2016spring/lectures.html
  - Prof. Jehee Lee, SNU, http://mrl.snu.ac.kr/courses/CourseGraphics/index_2017spring.html
  - Prof. Taesoo Kwon, Hanyang Univ., http://calab.hanyang.ac.kr/cgi-bin/cg.cgi