Lecture 1: Introduction & Animation Basics

Computer Animation
Teaching Team

Dr. Zerrin Yumak, Assistant Professor
Human-Centered Computing Group

Kazi Haque, PhD Candidate
Human-Centered Computing Group

Stefan Stan, Teaching Assistant
Game and Media Tech student
Today’s course

- Goals of the course
- Assignments
- History of Computer Animation
- Animation basics
- Forward Kinematics
- Orientation
- Quaternions
Goals of the course

- Introduction to basic techniques in Computer Animation
  - Motion synthesis, facial & body animation, ...

- Introduction to research topics
  - Giving presentations
  - Reading and evaluating research papers
  - Writing an essay about an animation topic

- Hands-on experience
  - Short animation movie production or programming exercise
Grading

- Research papers (R)
- Project (P)
- Essay (E)

Final grade = 0.3*R + 0.3*P + 0.4*E
- Condition: E >= 5

*Pay attention that R is based on your presentations but also involves paper summaries. You will not get a separate grade for the summaries but it is part of the overall grade R.
Attendance

- Attendance is overall not mandatory, but..
  - You are required to attend the lectures with student presentations you wrote a review for.
**Research papers**

- You will send a one A4 page review for each paper. In total, you should have 6 reviews.
  - Similar to peer-review process of conferences and journals
- Deadline for submitting these reviews is one day before the lecture until 23:59
- You are not limited to 6 papers though, read as much as you can, participate the presentations and ask questions!
- **Note:** In all your emails to the teacher or the TA, you must include [INFOMCANIM 2023] in the subject line of your email.
Research papers

Content of one page review:

- **Summary:** A short summary about what the paper is about, what is the method and results.

- **Assessment:** Critical analysis of the paper, advantages/disadvantages of the method, clarity, technical correctness, novelty

- **Questions:** Two questions about the paper (to be asked in the lecture)
Presentation

- Presentation should discuss the contents of the paper
  - What is it about, what is the contribution, what are the drawbacks of the proposed method...

- Presentations will be evaluated according to:
  - Content
  - Visual aids
  - Delivery
  - Ability to answer questions

- Presentation grade will be given to you the next session.
Presentation outline

- High-level overview and motivation of the paper
- Clear problem statement
- Related work and background information
- Technical details of the approach
- Critical analysis of the approach & evaluation
- Possible improvements of the paper (future work)
- Questions from the audience & discussion
Presentation preparation

- Do a ‘test-drive’ to check the presentation is not too long or too short
- Divide the slides with your team members
- Check your presentation in advance for hybrid set-up and for MS Teams to be sure it works well (especially audio/video files)
To be done in teams of 5

Two options:

- **Option 1: Movie production**
  - **Goal:** Learn state-of-the-art animation techniques used in animation packages, i.e. Blender, Unity, Unreal Engine

- **Option 2: Skeletal motion and skinning**
  - **Goal:** Practice basic animation techniques by programming them yourselves from scratch
Option 1: Animation movie

- Assignment: short animation movie
  - Make a storyboard and a previz movie
  - Capture in the mocap lab (mandatory) and find existing animations from the internet or design/edit yourself
  - Animate one or more characters and make a nice rendering

- Tools:
  - MotionBuilder, Maya, 3dsMax, Blender, Unity, Unreal Engine
  - We have a number of links for 3D Models
  - Look on the Internet for additional resources

Example videos from previous years can be found on the course website (External link).
Motion Capture and Virtual Reality Lab

- Full body motion capture equipment
  - Multiple actors tracking including their facial expressions and finger movements
  - 50 m² capture area
- Vicon Optical Motion Capture System
  - 14 cameras
  - Vicon Shogun (and Shogun Post) as software
- Facial capture using Dynamixyz
  - Single camera video-based system
  - Software Performer, Grabber and Live Pro
- Two high-performance PCs
  - One for Vicon and Dynamixyz tracking
  - Another for post-processing (Motion Builder is installed)
- Virtual Reality is also possible (not for this course)
  - Vicon Pulsars and HP Reverb G2 headsets
Evaluation criteria:
- Quality & complexity of animations (main criteria)
- Quality & complexity of rendering
- Story (engaging, entertaining..)
- Use of other supportive media (music, effects)

Send intermediary deliverables by 17 May.
- Previz movie and a 1-page planning report

Presentation on 28th June. Send final movie by 28 June 23:59!
- Movie and source project.
Option 2: Skeletal motion and skinning

- Composed of three parts:
  - **Skeletal animation:** Draw a skeleton in your preferred 3D graphics environment. Write classes to represent the joints and skeleton hierarchy and an animation player class to play key frame animations.
  
  - **Animations:** Find a basic set of motions keyframe/motion captured animations (e.g. walking, running, dancing, jumping). The demo should include the video of the original ground truth motion and 3D animation player side-by-side.

  - **Skinning (bonus):** Add a full body mesh to your program and apply linear blend skinning algorithm (or possibly dual-quaternion skinning) to animate the mesh together with the bones.

Example videos from previous years can be found on the course website (External link).
Option 2: Skeletal motion and skinning

- You can use your preferred 3D graphics environment or programming language
  - OpenGL/WebGL, DirectX, Vulkan etc.

- Cal3D and SmartBody are open-source skeletal animation libraries (something similar to what you will do)
  - You are not allowed to use these or similar high-level libraries.
Option 2: Skeletal motion and skinning

- **Evaluation criteria**
  - Program working as set by the initial goals
  - Well-written code and documentation
  - Demonstration

- **Send intermediary deliverables by 17 May.**
  - 2 page specification & planning report describing the programming environment/language you will use, description of the tasks and a planning of who will do what

- **Record a demo video of your software. Present it on 28\textsuperscript{th} June. Send your final material by 28\textsuperscript{th} June 23:59!**
  - Video, link to Github code and user manual
Essay

- 6 pages in double-column ACM Latex template
  - https://www.acm.org/publications/proceedings-proceedings-template

- Can be on any topic related to animation
  - Facial animation
  - Motion synthesis
  - Animation & emotions
  - Expressive animations (gaze, gesture, speech)
  - Locomotion
  - Hand animation
  - Skinning & Automatic Rigging
  - Deep learning and animation
Essay

- Choose **three papers** with different but related techniques
  - You can use your presentation paper or any other paper from the paper set as one of the papers in your essay if you want

- Describe the techniques in your essay, compare them in terms of pros and cons, show what the possible application areas are

- Describe the techniques in enough depth

- **Write the essay in your own words!**

- You can start writing it anytime during the course. The earlier the better! **Deadline 30 June until midnight!**
  - 0-24 hours late: 1 point deduction
  - 24-48 hours late: 2 points deduction
  - >48 hours late: submission rejected
Essay

- Evaluation criteria
  - depth and completeness of the description of the techniques,
  - completeness of the comparison between techniques, including strengths and drawbacks,
  - showing how the techniques can be applied,
  - writing the essay in your own words, instead of copying phrases from the original papers,
  - correct layout, spelling and grammar.
Retake

- You can do a retake essay
  - If you did all the assignments
  - If your grade is at least 4

- You may improve the essay you originally handed in, taking into account the feedback from the teacher

- or choose a new topic and write an essay about that.
Use of Generative AI Tools such as ChatGPT and Co-Pilot for the production of assignments is not allowed and considered as fraud!
Supplementary Material

- Computer Animation, Rick Parent
- Courses and tutorials at conferences (e.g. Siggraph and Eurographics)
- Research papers
  - See the course website (Assignments doc) for pointers to various conferences and journals
More info

Read the instructions on the course website carefully and check regularly for updates, check your email and MS Teams messages: [http://www.cs.uu.nl/docs/vakken/mcanim](http://www.cs.uu.nl/docs/vakken/mcanim)

For questions, ask during the lecture, write in MS Teams or send an email!
Animation

- Animate means “to give life to” according to Merriam-Webster
- Based on Latin word “anima”, meaning “breath” or “spirit”
  - E.g. Animal
- Animation is the process of making the illusion of motion and the illusion of change by means of the rapid display of a sequence of images that minimally differ from each other.

Pixar in a Box at Khan Academy - https://www.khanacademy.org/partner-content/pixar
Computer Animation

- Computer-based computation used in producing images intended to create the perception of motion
  - Algorithms and techniques that process 3D graphical data

- Object’s position and orientation
  - But also shape, shading parameters, texture coordinates, light source parameters, camera parameters
Two main categories

- Computer-assisted animation
  - 2D & 2 1/2 D
  - Inbetweening
  - Inking, virtual camera, managing data, etc.
  - Beauty and the Beast, Antz

- Computer generated animation
  - Low level techniques
    - Precisely specifying motion
  - High level techniques
    - Describe general motion behavior
  - Toy Story, Frozen, Inside Out, Shrek
Low level vs high level animation

- **Low-level techniques**
  - Shape interpolation
  - Helps the animator fill in the details of the motion given enough information
  - Animator has a fairly specific idea of target motion

- **High-level techniques**
  - Generate a motion given a set of rules or constraints
  - Object motion is controlled by a model/algorithm
  - Fairly sophisticated computation, such as physically-based motion
Example

- **Very low-level:** animator colours every pixel individually in every frame

- **Very-high level:** tell the computer “make a movie about a dog”

- Challenge lies in developing tools that allow animators to animate on different levels
Applications

- Special Effects (Movies, TV)
- Video Games
- Virtual Reality
- Simulation, Training, Military
- Medical
- Robotics, Animatronics
- Visualization
- Communication
Three general approaches to computer animation

- **Artistic animation**
  - Animator crafts the motion
  - Keyframing and interpolation

- **Data-driven animation**
  - Digitizing live motion and applying to 3D objects
  - Motion capture
  - AI-driven animation

- **Procedural animation**
  - Computational models of motion
  - By setting initial conditions for physical and behavioral simulation
Animation Tools

- **Animation tools:**
  - Maya
  - 3D Studio Max
  - MotionBuilder
  - Blender

- **Game/graphics engines:**
  - OGRE3D
  - Unity
  - Unreal

- **Open source skeletal animation library:**
  - Cal3D
  - Smartbody

- **3D graphics programming:**
  - OpenGL
  - DirectX
  - Vulkan
Virtual Characters

- Different levels of character motion
Real Time Animation of Virtual Humans: A Trade-off Between Naturalness and Control

H. van Welbergen\textsuperscript{1}, B. J. H. van Basten\textsuperscript{2}, A. Egges\textsuperscript{2}, Zs. M. Ruttkay\textsuperscript{1} and M. H. Overmars\textsuperscript{2}

\textsuperscript{1}Human Media Interaction, University of Twente, Enschede, The Netherlands
\{welberge, zsofi\}@ewi.utwente.nl
\textsuperscript{2}Center for Advanced Gaming and Simulation, Utrecht University, The Netherlands
\{basten, egges, m.h.overmars\}@cs.uu.nl

Abstract
Virtual humans are employed in many interactive applications using 3D virtual environments, including (serious) games. The motion of such virtual humans should look realistic (or ‘natural’) and allow interaction with the surroundings and other (virtual) humans. Current animation techniques differ in the trade-off they offer between motion naturalness and the control that can be exerted over the motion. We show mechanisms to parametrize, combine (on different body parts) and concatenate motions generated by different animation techniques. We discuss several aspects of motion naturalness and show how it can be evaluated. We conclude by showing the promise of combinations of different animation paradigms to enhance both naturalness and control.

Keywords: real-time animation, naturalness, control

ACM CCS: I.3.7 [Three-Dimensional Graphics and Realism]: Animation

1. Introduction
Virtual environments inhabited by virtual humans (VHs) are

First, we discuss models of the VH's body that are

steered by animation. In Section 3, we classify anima-

tion techniques that are used to generate motion prim-
History of Computer Animation
Perception

- Eye/brain assembles images and interprets them as continuous movement

- Persistence of vision: sequence of still images shown at a fast enough rate to induce sensation of continuous imagery

- Eye retains visual imprint once stimulus is removed
  - “positive afterimages”
Perception

- **Flicker**: the image flickers when persistence of vision does not occur
  - When the perception of continuous imagery fails to be created, the display is said to flicker.

- **Critical flicker frequency** is the rate at which the images must be shown in order to maintain persistence of vision.
  - Different things that affect it are: room lighting, viewing distance, etc. The critical flicker frequency determines the **lower bound** on the **playback rate**.
Perception

- **Lower bound:**
  - **Critical flicker frequency**: lower limits for establishing the perception of continuous imagery
  - **Playback/refresh rate**: number of images displayed per second

- **Upper bound:**
  - Upper limits the eye can perceive motion, if an object moves too quickly
  - **Motion blur**: the receptors in the eye will not be able to respond fast enough for the brain to distinguish sharply defined, individual detail and motion blur occurs.
Two important rates:

- **Playback/refresh rate**: number of images displayed per second
  - Related to flicker
- **Sampling/update rate**: number of distinct images that occur per second
  - How jerky the motion will appear
- e.g. Saturday morning cartoons have a sampling rate of 6 frames per second (fps) but each image is repeated five times, so the playback rate is 30 fps.
Frame Rates

- Film: 24 fps
- Imax: 48 fps
- NTSC TV: 30 fps (interlaced)
- PAL TV: 25 fps (interlaced)
- HDTV: 50-60 fps
- Computer: >60 fps

History of frame rate: [https://www.youtube.com/watch?v=mjYjFEp9Yx0](https://www.youtube.com/watch?v=mjYjFEp9Yx0)
Persistence of vision: discovered in the 1800s.

- Zoetrope
- Flipbook
- Thaumatrope
Eye Museum, Amsterdam
Computer Animation Production

- Conceptual Design
- Production Design
- Modeling
- Materials & Shaders
- Rigging
- Blocking
- Animation
- Lighting
- Effects
- Rendering
- Post-Production
12 Principles of Animation

http://www.siggraph.org/education/materials/HyperGraph/animation/character_animation/principles/prin_trad_anim.htm
12 Principles of Animation

- Squash and Stretch - defining the rigidity and mass of an object by distorting its shape during an action
- Timing and Motion - spacing actions to define the weight and size of objects and the personality of characters
- Anticipation - the preparation for an action
- Follow Through and Overlapping Action - the termination of an action and establishing its relationship to the next action
- Slow In and Out - the spacing of the in-between frames to achieve subtlety of timing and movement
- Arcs - the visual path of action for natural movement
- Exaggeration - Accentuating the essence of an idea via the design and the action
- Secondary Action - the action of an object resulting from another action
- ...
Lasseter translated traditional principles of animation to computer animation (1980s)

- Lasseter is conventionally trained animator
  - Worked at Disney before going to Pixar
    - Chief creative officer Pixar
  - Many animation films
    - Toy story, Wall-e, Frozen, Inside Out
Computer Animation Research

In Research labs (from 1970s)

- New York Institute of Technology (NYIT)
  - Gumby (Presented at SIGGRAPH '84 & '85 Electronic Theatres)

Still frame from Gumby animation by Hank Grebe and Dick Lundin, 1984.
Computer Animation Research

- University of Utah
  - Films on walking and talking figure
  - Animated hand and animated face (Ed Catmull (president of Pixar), 1972) (Fred Parke, Talking face, 1974)

- University of Pennsylvania
  - Human figure animation (Norm Badler, 1975)

- MIRALab, Geneva
  - Virtual Humans (Daniel & Nadia Thalmann, 1980s)
Ed Catmull: Co-founder of Pixar and was president of Walt Disney Animation Studios
More recent movies with CG

- Final fantasy (2001)
  - Fully 3D simulated environment

  - One of the first movies using crowds (Massive)

- Avatar (2009)
- Benjamin Button (2008)
- Prometheus (2012)
Animation Basics
Position Vector Dot Matrix

\[
v = \begin{bmatrix} v_x & v_y & v_z & 1 \end{bmatrix}
\]

\[
M = \begin{bmatrix} a_x & a_y & a_z & 0 \\ b_x & b_y & b_z & 0 \\ c_x & c_y & c_z & 0 \\ d_x & d_y & d_z & 1 \end{bmatrix}
\]

\[
v' = v \cdot M
\]

\[
\begin{align*}
v'_x &= v_x a_x + v_y b_x + v_z c_x + d_x \\
v'_y &= v_x a_y + v_y b_y + v_z c_y + d_y \\
v'_z &= v_x a_z + v_y b_z + v_z c_z + d_z \\
v'_w &= 1
\end{align*}
\]

\[
v' = v_x a + v_y b + v_z c + d
\]
\[ v' = v_x a + v_y b + v_z c + d \]
\[ \mathbf{v}' = v_x \mathbf{a} + v_y \mathbf{b} + v_z \mathbf{c} + \mathbf{d} \]

Local Space

World Space

Matrix \( \mathbf{M} \)
\[ v' = v_x a + v_y b + v_z c + d \]
Direction Vector Dot Matrix

\[ \mathbf{v} = \begin{bmatrix} v_x & v_y & v_z & 0 \end{bmatrix} \]

\[ \mathbf{v}' = \mathbf{v} \cdot \mathbf{M} \]

\[ \begin{align*}
  v'_x &= v_x a_x + v_y b_x + v_z c_x \\
  v'_y &= v_x a_y + v_y b_y + v_z c_y \\
  v'_z &= v_x a_z + v_y b_z + v_z c_z \\
  v'_w &= 0
\end{align*} \]

\[ \mathbf{v}' = v_x \mathbf{a} + v_y \mathbf{b} + v_z \mathbf{c} \]

\[ \mathbf{M} = \begin{bmatrix} a_x & a_y & a_z & 0 \\
  b_x & b_y & b_z & 0 \\
  c_x & c_y & c_z & 0 \\
  d_x & d_y & d_z & 1 \end{bmatrix} \]
Transformations

- To transform a vector \( \mathbf{v} \) by matrix \( \mathbf{M} \):
  \[
  \mathbf{v}' = \mathbf{v} \cdot \mathbf{M}
  \]

- If we want to apply several transformations, we can just multiply by several matrices:
  \[
  \mathbf{v}' = (((\mathbf{v} \cdot \mathbf{M}_1) \cdot \mathbf{M}_2) \cdot \mathbf{M}_3) \cdot \mathbf{M}_4 \ldots
  \]

- Or we can concatenate the transformations into a single matrix:
  \[
  \mathbf{M}_{\text{total}} = \mathbf{M}_1 \cdot \mathbf{M}_2 \cdot \mathbf{M}_3 \cdot \mathbf{M}_4 \ldots
  \]
  \[
  \mathbf{v}' = \mathbf{v} \cdot \mathbf{M}_{\text{total}}
  \]
If $M$ transforms $v$ into world space, then $M^{-1}$ transforms $v'$ back into local space.

$$v' = v \cdot M$$

$$v = v' \cdot M^{-1}$$

$$M \cdot M^{-1} = I$$
Matrix Transformations

- We usually transform vertices from some local space where they are defined into a world space:
  \[ \mathbf{v}' = \mathbf{v} \cdot \mathbf{W} \]

- Once in world space, we can perform operations that require everything to be in the same space (collision detection, high quality lighting...)

- Then, they are transformed into a camera’s space, and then projected into 2D:
  \[ \mathbf{v}'' = \mathbf{v}' \cdot \mathbf{C}^{-1} \cdot \mathbf{P} \]

- In simple situations, we can do this all in one step:
  \[ \mathbf{v}'' = \mathbf{v} \cdot \mathbf{W} \cdot \mathbf{C}^{-1} \cdot \mathbf{P} \]
(Forward) Kinematics
Virtual Characters

- **Representation: skeletal model**
  - A VH is represented by a polyhedral model (or mesh)
  - An underlying skeleton deforms this mesh
    - Joints, connected by bones
  - A pose is defined by the rotations of the joints and the position of the root joint
Example Joint Hierarchy

Root

- Torso
  - Neck
  - Head
  - ShoulderL
  - ElbowL
  - WristL
- ShoulderR
  - ElbowR
  - WristR

- Pelvis
  - HipL
  - KneeL
  - AnkleL
  - HipR
  - KneeR
  - AnkleR
**DOFs**

- **Degree of Freedom (DOF)**
  - A variable $\phi$ describing a particular axis or dimension of movement within a joint

- Joints typically have around 1-6 DOFs ($\phi_1 \ldots \phi_N$)

- Changing the DOF values over time results in the animation of the skeleton

- Note: in a mathematical sense, a free rigid body has 6 DOFs: 3 for position and 3 for rotation
Kinematics

- **Kinematics**
  - The analysis of motion independent of physical forces. Kinematics deals with position, velocity, acceleration, and their rotational counterparts, orientation, angular velocity, and angular acceleration.

- **Forward Kinematics**
  - The process of computing world space geometric data from DOFs

- **Inverse Kinematics**
  - The process of computing a set of DOFs that causes some world space goal to be met (i.e., place the hand on the door knob...).
Skeletons

- **Skeleton**
  - A pose-able framework of joints arranged in a tree structure.

- **Joint**
  - Allows relative movement within the skeleton.
  - Are essentially $4 \times 4$ matrix transformations
  - Can be rotational, translational, or other
  - Synonym: **bone**
Joints

- **Core Joint Data**
  - DOFs (N floats)
  - Local matrix: \( \mathbf{L} \)
  - World matrix: \( \mathbf{W} \)

- **Additional Data**
  - Joint offset vector: \( \mathbf{r} \)
  - DOF limits (min & max value per DOF)
  - Tree data (pointers to children, siblings, parent...)

Skeleton Posing Process

1. Specify all DOF values for the skeleton (**done by higher level animation system**)

2. Recursively traverse through the hierarchy starting at the root and use forward kinematics to compute the world matrices (**done by skeleton system**)

3. Use world matrices to deform skin & render (**done by skinning system**)

Note: the matrices can also be used for other things such as collision detection, FX, etc.
In the recursive tree traversal, each joint first computes its local matrix $L$ based on the values of its DOFs and some formula representative of the joint type:

$$\text{Local matrix } L = L_{\text{joint}}(\phi_1, \phi_2, \ldots, \phi_N)$$

Then, world matrix $W$ is computed by concatenating $L$ with the world matrix of the parent joint

$$\text{World matrix } W = L \cdot W_{\text{parent}}$$
Joint Offsets

- It is convenient to have a 3D offset vector \( \mathbf{r} \) for every joint which represents its pivot point relative to its parent’s matrix

\[
\mathbf{L}_{\text{offset}} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
\mathbf{r}_x & \mathbf{r}_y & \mathbf{r}_z & 1
\end{bmatrix}
\]
It is nice to be able to limit a DOF to some range (for example, the elbow could be limited from 0° to 150°)

Usually, in a realistic character, all DOFs will be limited except the ones controlling the root
One can then adjust each of the DOFs to specify the pose of the skeleton.

We can define a pose $\Phi$ more formally as a vector of $N$ numbers that maps to a set of DOFs in the skeleton:

$$\Phi = [\phi_1 \phi_2 \ldots \phi_N]$$

A pose is a convenient unit that can be manipulated by a higher level animation system and then handed down to the skeleton.

Usually, each joint will have around 1-6 DOFs, but an entire character might have 100+ DOFs in the skeleton.
Rotational DOFs are widely used in character animation

3 translational DOFs
48 rotational DOFs

Each joint can have up to 3 DOFs

1 DOF: knee
2 DOF: wrist
3 DOF: arm
Joint Types

- Rotational
  - Hinge: 1-DOF
  - Universal: 2-DOF
  - Ball & Socket: 3-DOF
    - Euler Angles
    - Quaternions

- Translational
  - Prismatic: 1-DOF
  - Translational: 3-DOF (or any number)
Hinge Joints (1-DOF Rotational)

- Rotation around the x-axis:

\[ L_{Rx}(\theta_x) = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & \cos \theta_x & \sin \theta_x & 0 & 0 \\
0 & -\sin \theta_x & \cos \theta_x & 0 & 0 \\
r_x & r_y & r_z & 1 & 1
\end{bmatrix} \]
Hinge Joints (1-DOF Rotational)

- Rotation around the y-axis:

\[
L_{R_y}(\theta_y) = \begin{bmatrix}
\cos \theta_y & 0 & -\sin \theta_y & 0 \\
0 & 1 & 0 & 0 \\
\sin \theta_y & 0 & \cos \theta_y & 0 \\
rx & ry & rz & 1
\end{bmatrix}
\]
Hinge Joints (1-DOF Rotational)

- Rotation around the z-axis:

\[
L_{Rz}(\theta_z) = \begin{bmatrix}
\cos \theta_z & \sin \theta_z & 0 & 0 \\
-\sin \theta_z & \cos \theta_z & 0 & 0 \\
0 & 0 & 1 & 0 \\
r_x & r_y & r_z & 1
\end{bmatrix}
\]
Universal Joints (2-DOF)

For a 2-DOF joint that first rotates around x and then around y:

\[
L_{Rxy}(\theta_x, \theta_y) = \begin{bmatrix}
    c_y & 0 & -s_y & 0 \\
    s_x s_y & c_x & s_x c_y & 0 \\
    c_x s_y & -s_x & c_x c_y & 0 \\
    r_x & r_y & r_z & 1
\end{bmatrix}
\]
Ball & Socket (3-DOF)

- For a 3-DOF joint that first rotates around $x$, $y$, then $z$:

$$L_{R_{xyz}}(\theta_x, \theta_y, \theta_z) = \begin{bmatrix}
    c_y c_z & c_y s_z & -s_y \\
    s_x s_y c_z - c_x s_z & s_x s_y s_z + c_x c_z & s_x c_y \\
    c_x s_y c_z + s_x s_z & c_x s_y s_z - s_x c_z & c_x c_y \\
    r_x & r_y & r_z & 1
\end{bmatrix}$$

- Different matrices can be formed for different axis combinations.
**Prismatic Joints (1-DOF Translation)**

- 1-DOF translation along an arbitrary axis $\mathbf{a}$:

\[
\mathbf{L}_{Ta}(t) = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
rx + t \cdot ax & ry + t \cdot ay & rz + t \cdot az & 1
\end{bmatrix}
\]
Translational Joints (3-DOF)

For a more general 3-DOF translation:

\[ L_{Txyz}(t) = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

where \( r_x + t_x \), \( r_y + t_y \), and \( r_z + t_z \) are the translation components.
Orientation
We will define ‘orientation’ to mean an object’s instantaneous rotational configuration.

Think of it as the rotational equivalent of position.
Is there a simple means of representing a 3D orientation? (analogous to Cartesian coordinates?)

Not really.

There are several popular options though:
- 3x3 matrices
- Fixed/Euler angles
- Rotation vectors (axis/angle)
- Quaternions
- and more...
This means that we can represent an orientation with 3 numbers.

A sequence of rotations around principle axes is called an *Euler Angle Sequence*.

Assuming we limit ourselves to 3 rotations without successive rotations about the same axis, we could use any of the following 12 sequences:

<table>
<thead>
<tr>
<th>XYZ</th>
<th>XZY</th>
<th>XYX</th>
<th>XZX</th>
</tr>
</thead>
<tbody>
<tr>
<td>YXZ</td>
<td>YZX</td>
<td>YXY</td>
<td>YZY</td>
</tr>
<tr>
<td>ZXY</td>
<td>ZYX</td>
<td>ZXZ</td>
<td>ZYZ</td>
</tr>
</tbody>
</table>
To build a matrix from a set of Euler angles, we just multiply a sequence of rotation matrices together:

\[
\mathbf{R}_x \cdot \mathbf{R}_y \cdot \mathbf{R}_z = 
\begin{bmatrix}
1 & 0 & 0 \\
0 & c_x & s_x \\
0 & -s_x & c_x \\
\end{bmatrix}
\begin{bmatrix}
y & 0 & -s_y \\
0 & 1 & 0 \\
y & 0 & c_y \\
\end{bmatrix}
\begin{bmatrix}
c_z & s_z & 0 \\
-s_z & c_z & 0 \\
0 & 0 & 1 \\
\end{bmatrix}
= 
\begin{bmatrix}
c_y c_z & c_y s_z & -s_y \\
S_x S_y c_z - c_x S_z & S_x S_y S_z + c_x c_z & S_x c_y \\
c_x S_y c_z + S_x S_z & c_x S_y S_z - S_x c_z & c_x c_y \\
\end{bmatrix}
\]
As matrix multiplication is not commutative, the order of operations is important.

Rotations are assumed to be relative to fixed world axes or local to the object.
Euler Angles

- They can suffer from Gimbal lock and related problems
- They do not interpolate in a consistent way
- There is no simple way to concatenate rotations
- Conversion to/from a matrix requires several trigonometry operations
- They are compact (requiring only 3 numbers)
Gimbal Lock

- One potential problem that they can suffer from is ‘gimbal lock’
- This results when two axes effectively line up, resulting in a temporary loss of a degree of freedom

https://www.youtube.com/watch?v=zc8b2Jo7mno
We use a 3x3 matrix to represent an orientation as well.

This means we now have 9 numbers instead of 3, and therefore, we have 6 extra degrees of freedom.

NOTE: We don’t use 4x4 matrices here, as those are mainly useful because they give us the ability to combine translations. We will not be concerned with translation, so we will just think of 3x3 matrices.
Matrix Representation

- Matrices are usually the most computationally efficient way to apply rotations to geometric data, and so most orientation representations ultimately need to be converted into a matrix in order to do anything useful (transform verts...)

- Why then, shouldn’t we just always use matrices?
  - Numerical issues
  - Storage issues
  - User interaction issues
  - Interpolation issues
Any two orientations can be related by a single rotation about some axis (not necessarily a principle axis)

This means that we can represent an arbitrary orientation as a rotation about some unit axis by some angle (4 numbers) (Axis/Angle form)

Alternately, we can scale the axis by the angle and compact it down to a single 3D vector
- rotation vector or exponential map
To generate a matrix as a rotation $\theta$ around an arbitrary unit axis $\mathbf{a}$:

$$
\begin{bmatrix}
    a_x^2 + c_\theta (1-a_x^2) & a_x a_y (1-c_\theta) + a_z s_\theta & a_x a_z (1-c_\theta) - a_y s_\theta \\
    a_x a_y (1-c_\theta) - a_z s_\theta & a_y^2 + c_\theta (1-a_y^2) & a_y a_z (1-c_\theta) + a_x s_\theta \\
    a_x a_z (1-c_\theta) + a_y s_\theta & a_y a_z (1-c_\theta) - a_x s_\theta & a_z^2 + c_\theta (1-a_z^2)
\end{bmatrix}
$$
Quaternions
Quaternions are an interesting mathematical concept with a deep relationship with the foundations of algebra and number theory.

- Invented by W.R. Hamilton in 1843.
- In practice, they are most useful to us as a means of representing orientations.
- A quaternion has 4 components.

\[
q = [q_0 \quad q_1 \quad q_2 \quad q_3]
\]
Quaternions (Imaginary Space)

- Quaternions are actually an extension to complex numbers.
- Of the 4 components, one is a ‘real’ scalar number, and the other 3 form a vector in imaginary ijk space!

\[ q = q_0 + iq_1 + jq_2 + kq_3 \]

\begin{align*}
  i^2 &= j^2 &= k^2 &= ijk = -1 \\
  i &= jk &= -kj \\
  j &= ki &= -ik \\
  k &= ij &= -ji
\end{align*}
Quaternions (Scalar/Vector)

- Sometimes, they are written as the combination of a scalar value $s$ and a vector value $v$

$$q = \langle s, v \rangle$$

where

$$s = q_0$$

$$v = \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix}$$
Unit Quaternions

- For convenience, we will use only unit length quaternions, as they will be sufficient for our purposes and make things a little easier.

\[ |q| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1 \]

- These correspond to the set of vectors that form the ‘surface’ of a 4D hypersphere of radius 1.

- The ‘surface’ is actually a 3D volume in 4D space, but it can be visualized as an extension to the concept of a 2D surface on a 3D sphere.
Visualizing quaternions

1-angle rotation can be represented by a unit circle

2-angle rotation can be represented by a unit sphere

- What about 3-angle rotation?
- A unit quaternion is a point on the 4D sphere
Quaternions as Rotations

- A quaternion can represent a rotation by an angle $\theta$ around a unit axis $\mathbf{a}$:

$$\mathbf{q} = \begin{bmatrix}
\cos \frac{\theta}{2} & a_x \sin \frac{\theta}{2} & a_y \sin \frac{\theta}{2} & a_z \sin \frac{\theta}{2}
\end{bmatrix}$$

or

$$\mathbf{q} = \langle \cos \frac{\theta}{2}, \mathbf{a} \sin \frac{\theta}{2} \rangle$$

- If $\mathbf{a}$ is unit length, then $\mathbf{q}$ will be also
Quaternions as Rotations

\[
|q| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}
\]

\[
= \sqrt{\cos^2 \frac{\theta}{2} + a_x^2 \sin^2 \frac{\theta}{2} + a_y^2 \sin^2 \frac{\theta}{2} + a_z^2 \sin^2 \frac{\theta}{2}}
\]

\[
= \sqrt{\cos^2 \frac{\theta}{2} + \sin^2 \frac{\theta}{2} \left(a_x^2 + a_y^2 + a_z^2\right)}
\]

\[
= \sqrt{\cos^2 \frac{\theta}{2} + \sin^2 \frac{\theta}{2} |a|^2} = \sqrt{\cos^2 \frac{\theta}{2} + \sin^2 \frac{\theta}{2}}
\]

\[
= \sqrt{1} = 1
\]
Quatertion to Matrix

- To convert a quaternion to a rotation matrix:

\[
\begin{bmatrix}
1 - 2q_2^2 - 2q_3^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\
2q_1q_2 - 2q_0q_3 & 1 - 2q_1^2 - 2q_3^2 & 2q_2q_3 + 2q_0q_1 \\
2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & 1 - 2q_1^2 - 2q_2^2
\end{bmatrix}
\]
Matrix to Quaternion

- Matrix to quaternion is not too bad, I just don’t have room for it here

- It involves a few ‘if’ statements, a square root, three divisions, and some other stuff

- For the algorithm, see:
  http://www.euclideanspace.com/maths/geometry/rotations/conversions/matrixToQuaternion/index.htm
Hyperspheres

- A distance of $x$ along the surface of the hypersphere corresponds to a rotation of angle $2x$ radians.

- This means that moving along a 90 degree arc on the hypersphere corresponds to rotating an object by 180 degrees.

- Traveling 180 degrees corresponds to a 360 degree rotation, thus getting you back to where you started.

- This implies that $q$ and $-q$ correspond to the same orientation.
Consider what happens if you rotate a book 180 around x, then 180 around y, and then 180 around z.

You end up back where you started.

This corresponds to traveling along a triangle on the hypersphere where each edge is a 90 degree arc, orthogonal to each other edge.
Quatertion Joints

- One can create a skeleton using quaternion joints
- One possibility is to simply allow a quaternion joint type and provide a local matrix function that takes a quaternion
- Another possibility is to also compute the world matrices as quaternion multiplications. This involves a little less math than matrices, but may not prove to be significantly faster. Also, one would still have to handle the joint offsets with matrix math
Quaternions in the Pose Vector

- Using quaternions in the skeleton adds some complications, as they can’t simply be treated as 4 independent DOFs through the rig.

- The reason is that the 4 numbers are not independent, and so an animation system would have to handle them specifically as a quaternion.

- To deal with this, one might have to extend the concept of the pose vector as containing an array of scalars and an array of quaternions.

- When higher level animation code blends and manipulates poses, it will have to treat quaternions specially.
Quaternions in the Pose Vector

- Using quaternions in the skeleton adds some complications, as they can’t simply be treated as 4 independent DOFs through the rig.
- The reason is that the 4 numbers are not independent, and so an animation system would have to handle them specifically as a quaternion.
- To deal with this, one might have to extend the concept of the pose vector as containing an array of scalars and an array of quaternions.
- When higher level animation code blends and manipulates poses, it will have to treat quaternions specially.
Quaternion Interpolation
If we want to do a linear interpolation between two points \( a \) and \( b \) in normal space

\[
\text{Lerp}(t, a, b) = (1-t)a + (t)b
\]

where \( t \) ranges from 0 to 1

- Note that the Lerp operation can be thought of as a weighted average

- We could also write it in its additive blend form:

\[
\text{Lerp}(t, a, b) = a + t(b-a)
\]
If we want to interpolate between two points on a sphere (or hypersphere), we don’t just want to Lerp between them.

Instead, we will travel across the surface of the sphere by following a ‘great arc’.
We define the spherical linear interpolation of two unit vectors in N dimensional space as:

\[ Slerp(t, \mathbf{a}, \mathbf{b}) = \frac{\sin((1-t)\theta)}{\sin \theta} \mathbf{a} + \frac{\sin(t\theta)}{\sin \theta} \mathbf{b} \]

where \( \theta = \cos^{-1}(\mathbf{a} \cdot \mathbf{b}) \)
Remember that there are two redundant vectors in quaternion space for every unique orientation in 3D space.

What is the difference between: $\text{Slerp}(t, a, b)$ and $\text{Slerp}(t, -a, b)$?

One of these will travel less than 90 degrees while the other will travel more than 90 degrees across the sphere.

This corresponds to rotating the ‘short way’ or the ‘long way’.

Usually, we want to take the short way, so we negate one of them if their dot product is $< 0$ (angle greater than 90 degrees).
We can construct Bezier curves on the 4D hypersphere by following the exact same procedure using Slerp instead of Lerp.

It’s a good idea to flip (negate) the input quaternions as necessary in order to make it go the ‘short way’.
Quaternions are 4D vectors that can represent 3D rigid body orientations

We choose to force them to be unit length

Key animation functions:
- Quaternion-to-matrix / matrix-to-quaternion
- Quaternion multiplication: faster than matrix multiplication
- Slerp: interpolate between arbitrary orientations
- Spherical curves: de Castlejau algorithm for cubic Bezier curves on the hypersphere
Comparison of representations

- **For animation input:** Euler angles
- **Setting joint limits:** Euler angles, axis angle
- **Good interpolation:** Axis angle and quaternions
- **Good composition:** Quaternions and orientation matrix
- **Rendering:** Orientation matrix
References

- Rick Parent book Chapter 1, 2 and 3
- “Animating Rotation with Quaternion Curves”, Ken Shoemake, SIGGRAPH 1985
- Computer Animation course at the University of California San Diego.
What to do after this lecture

- Fill in the Google forms to make groups for the paper presentation and for the project and select the paper that you will present.
  - [Presentation Form](#)
  - [Project Form](#)
  - [Link to 22 papers](#)

- **Deadline: 3 May**