

# Inverse Kinematics

2021.5.4

# Outline

- Demo
- Project overview
- Scoring criteria
- Objective and explanation
- Submission detail
- Hint and reminder

# Demo

- Inverse Kinematics
- Bone touch the target
  - rfingers (id=29)
- Last movable bone
  - lowerback (id=11)



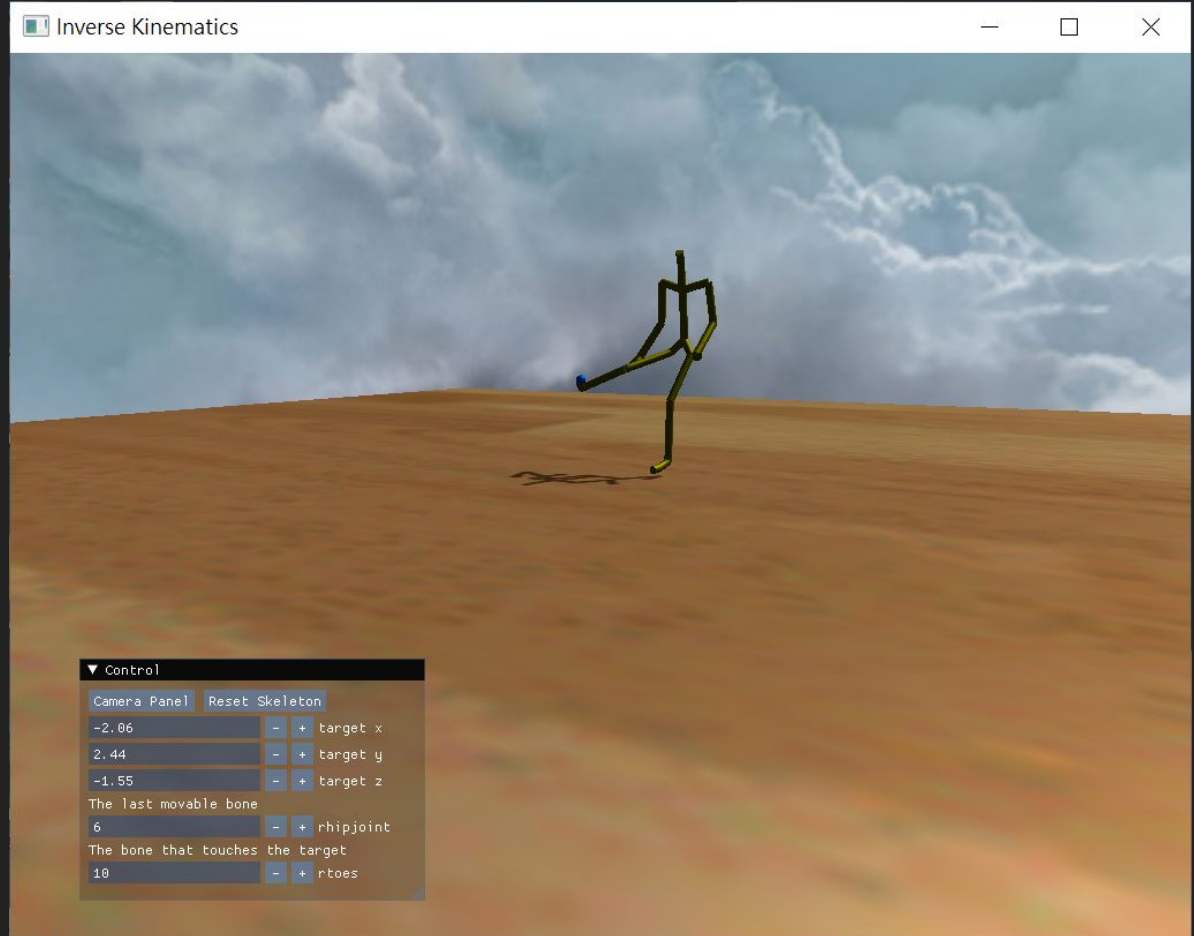
# Demo (cont.)

- When target is unreachable ...



# Demo (cont.)

- Bone touch the ball
  - rtoes (id=10)
- Last movable bone
  - rhipjoint (id=6)



# Project overview (Same as HW2)

- Solution layout
  - bin
  - assets
    - Shader and Texture
    - Acclaim - skeleton (asf) and motion (amc) files
  - src (source code)
  - include (header files for src)
  - extern (project dependencies)
  - InverseKinematics (Visual Studio project and main)

# Project overview (cont.)

- Environment
  - IDE: Visual studio 2017 / 2019
  - Platform: Windows
  - Graphics API: OpenGL
  - OpenGL Loading Library: glad2
  - OpenGL Toolkit: glfw
  - UI Library: dear imgui
  - Math Library: Eigen

# Project overview (cont.)

- src
  - acclaim (code for parsing acclaim files)
  - graphics (code for rendering geometries)
  - simulation (code for running simulation)
  - util (utilities)
- Everything you need to implement is in the simulation folder



# Scoring Criteria

- Inverse kinematics - 80%
  - Pseudoinverse - 20%
  - Related Implementation - 60%
- Report - 20%
- Bonus - 5%

local  
unit rotation axis 手到世界坐标的  
应该是 rotation matrix 的 column 0, 1, 2  
↓ ↓ ↓  
 $\theta_x$  轴  $\theta_y$  轴  $\theta_z$  轴

# Objective and explanation

- `pseudoinverseLinearSolver(Jacobian, target)`
  - Find solution of linear least squares system
  - i.e. find  $x$  which  $\min(|\text{Jacobian} * x - \text{target}|)$
- `inverseJacobianIKSolver(target_pos, start_bone, end_bone, posture)`
  - Solve inverse kinematics using **inverse jacobian method**.
  - Return true if IK is stable, false otherwise (Bonus)

# Objective and explanation (cont.)

- Report (below is a suggested outline)
  - Introduction/Motivation
  - Fundamentals  $\rightarrow$   $\mathbb{R}^n$  原理
  - Implementation
  - Result and Discussion
    - How different step and epsilon affect the result
  - Conclusion

# Submission detail

- Compress required files into a .zip file
  - Naming rule: CA3\_StudentID.zip
    - e.g., CA3\_309553010.zip
- Your zip file should contain following components
  - simulation/kinematics.cpp
  - Report in **PDF** format, no more than 10 pages
  - **DO NOT** include whole project

## Submission detail (cont.)

- Upload all your materials to new E3
  - No limit to the number of times of upload
  - The latest version is your final submission

# Submission detail (cont.)

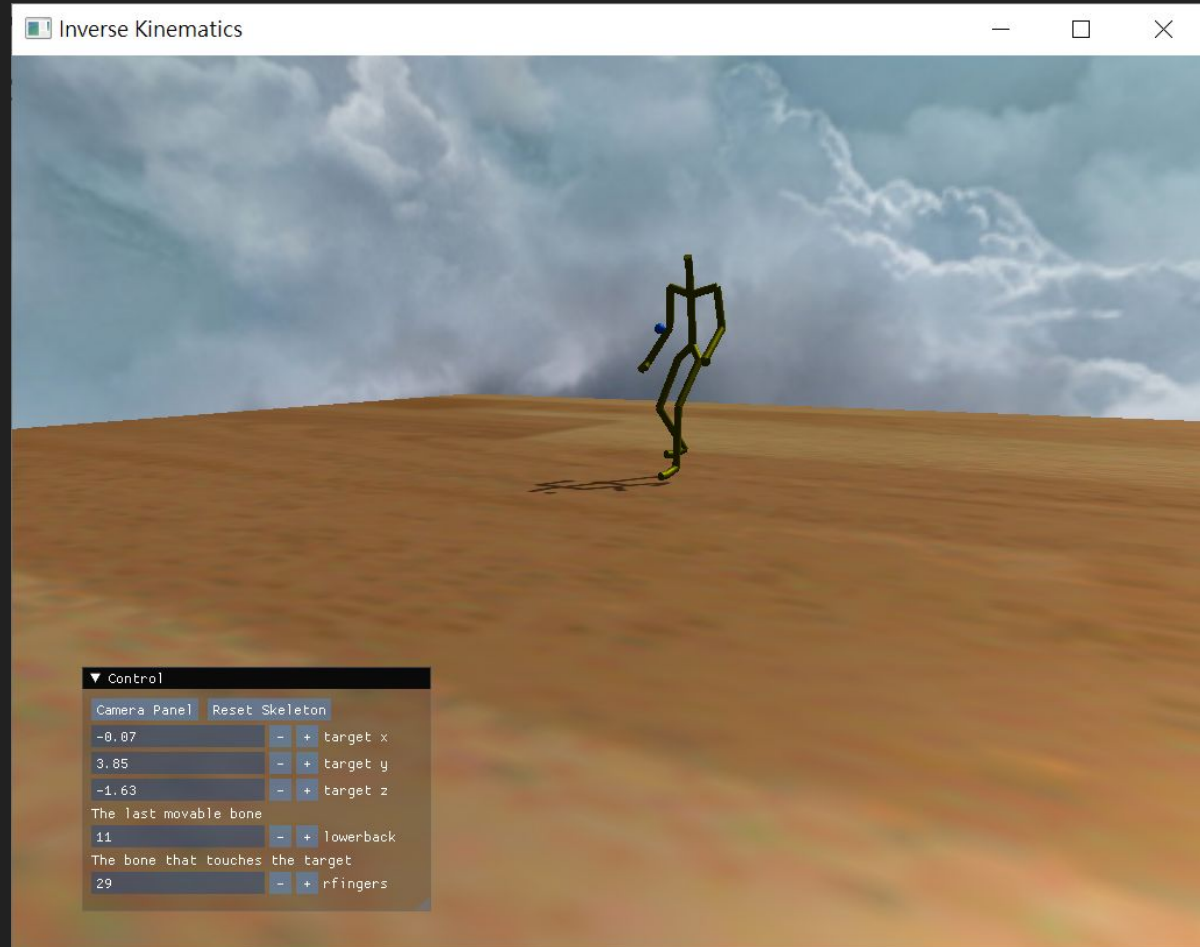
- Late policies
  - Penalty of 10 points on each day after deadline
- Cheating policies
  - 0 points for any cheating on assignments
  - Allowing another student to examine your code is also considered as cheating
- Deadline
  - Monday, 2021/05/24, 23:55

# Hint and Reminder

- Hint: course materials
  - Inverse Kinematics
    - Review "Kinematics.pptx" from p.20 - p.50
    - Review "acclaim\_FK\_IKnote.pdf" Inverse Kinematics part
- You need to implement Inverse-Jacobain method in this homework.

# Hint and Reminder

- Blank template:





# Hint and Reminder (cont.)

- Computing Jacobian geometrically

## Rotational DOFs

- Let's consider a 1-DOF rotational joint first
- We want to know how the global position  $\mathbf{p}$  will change if we rotate around the axis.

$$\mathbf{v} = \boldsymbol{\omega} \times \mathbf{r}$$

$$\frac{d\mathbf{p}}{dt} = |\boldsymbol{\omega}| \frac{\boldsymbol{\omega}}{|\boldsymbol{\omega}|} \times \mathbf{r} = \frac{d\theta}{dt} \mathbf{a} \times \mathbf{r}$$

$$\frac{d\mathbf{p}}{d\theta} = \mathbf{a} \times \mathbf{r}$$

$$\frac{\partial \mathbf{p}}{\partial \theta_1} = \begin{bmatrix} \frac{\partial p_x}{\partial \theta_1} \\ \frac{\partial p_y}{\partial \theta_1} \end{bmatrix} = \mathbf{a}_1 \times (\mathbf{p} - \mathbf{r}_1)$$

↑  
unit-length rotation  
axis vector

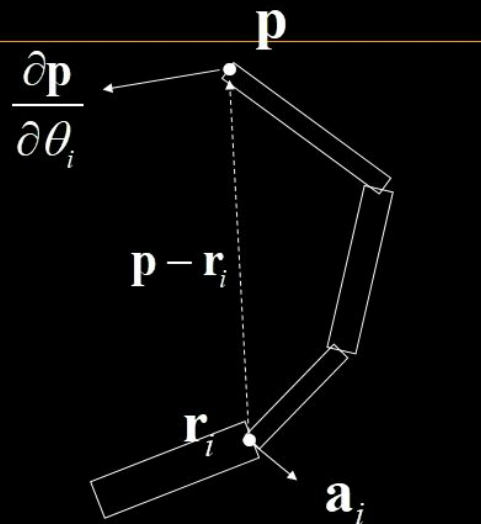
$$\mathbf{a}_1 = \frac{\boldsymbol{\omega}_1}{|\boldsymbol{\omega}_1|}$$



## Hint and Reminder (cont.)

### Rotational DOFs

$$\frac{\partial \mathbf{p}}{\partial \theta_i} = \mathbf{a}_i \times (\mathbf{p} - \mathbf{r}_i)$$



$\mathbf{a}_i$ : unit length rotation axis in world space

$\mathbf{r}_i$ : position of joint pivot in world space

$\mathbf{p}$ : end effector position in world space

## Hint and Reminder (cont.)

### Iterative IK Using Inverse Jacobian

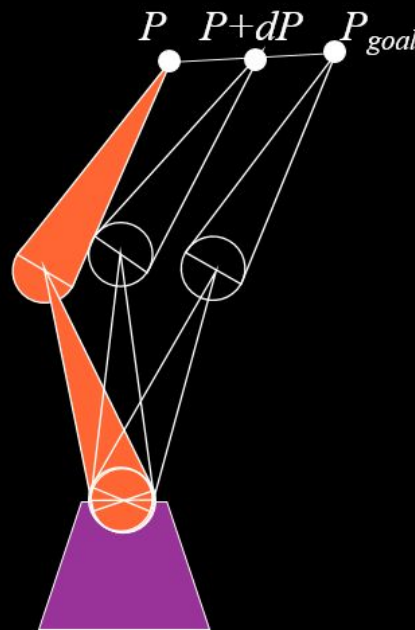
$$\theta = f^{-1}(P)$$

$$V = J(\theta)\dot{\theta}$$

$$\dot{\theta} = J^{-1}(\theta)V$$

$$\theta_{k+1} = \theta_k + \Delta t J^{-1}(\theta_k)V$$

- Linearize about  $\theta_k$  locally
- Small increments



## Hint and Reminder (cont.)

- What is Eigen::Matrix4Xd?
  - A matrix that has 4 rows, but unknown columns
  - e.g. Eigen::Matrix4Xd a(4, 30); // a has 4 rows, 30 columns of doubles.
- What is Eigen::VectorXd?
  - A vector with unknown size.
  - e.g. Eigen::VectorXd b(30); // b has 30 doubles
- Target position change too slow!
  - Hold ctrl when you press + or - *假设*
- What is "last movable bone"? *→ 有一个bone在图中, 会一直回着找 root 找到 last movable bone*
  - While you are traversing parents of "touch target bone", you should stop at this bone. *bone*

# Hint and Reminder (cont.)

- How to contact TA?
  - Please ask your question on new E3 forum. (recommend)
    - Questions are similar, you can find answer easier.
    - If you cannot ask question without providing your code
      - You are probably asking TA debug your code.
    - Send email to **BOTH** TAs via new E3 if you think the question is personal.
  - If you need to ask question face-to-face, please send email for appointment.
  - IMPORTANT: please sort out and arrange your question, so we can help you without wasting time on trivial matters.