Kinematics

HW2
2025 Computer Animation and Special Effects

Outline

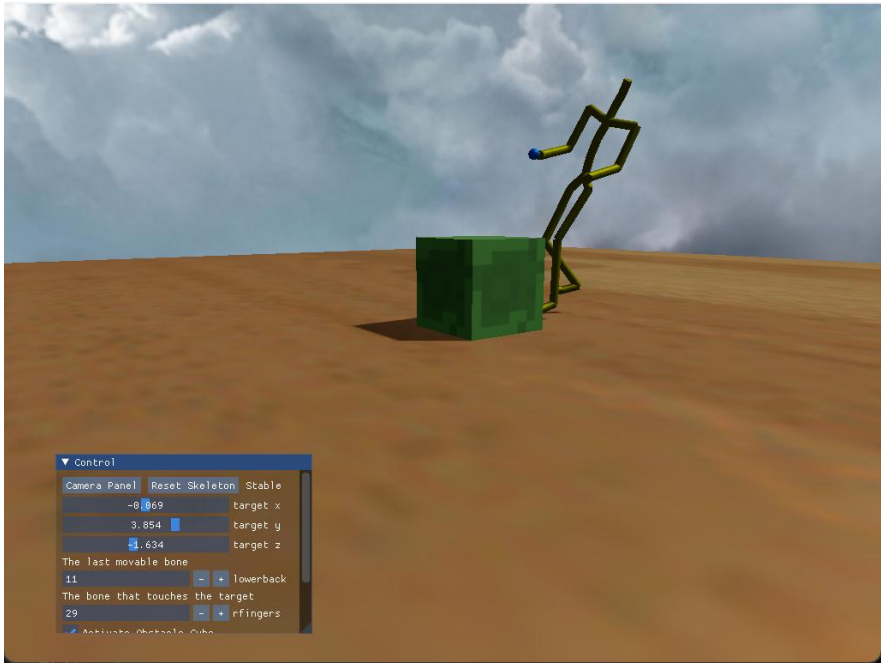
- Overview
- Objective
- Report
- Grading
- Submission
- Note

Overview

Use different bones to touch the ball

- Start bone
  - The last movable bone
- End bone (End Effector)
  - The bone that touches the ball

- Obstacle Avoidance
  - More information on the next page

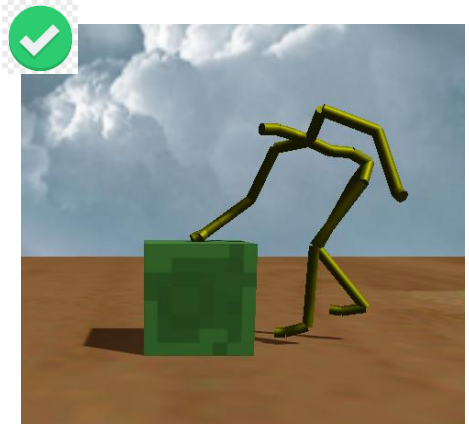


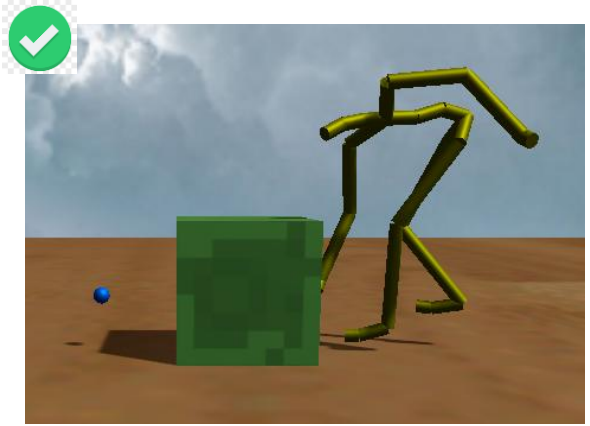
Demo link: https://youtu.be/O6-VDhfMGPM

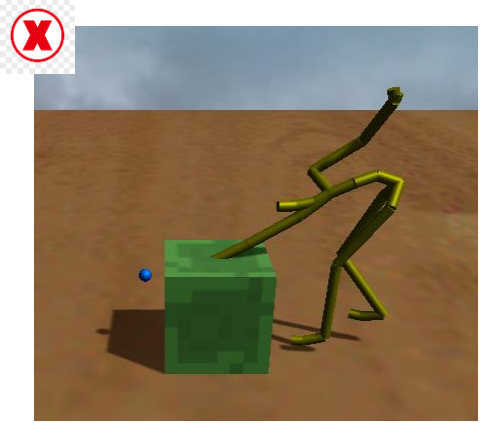
cube size: 1 * 1 * 1

Obstacle Avoidance

- The bones should not penetrate the obstacle cube
- Avoidance can be implemented by injecting repulsion into the Inverse Kinematic target when bones approach the obstacle.







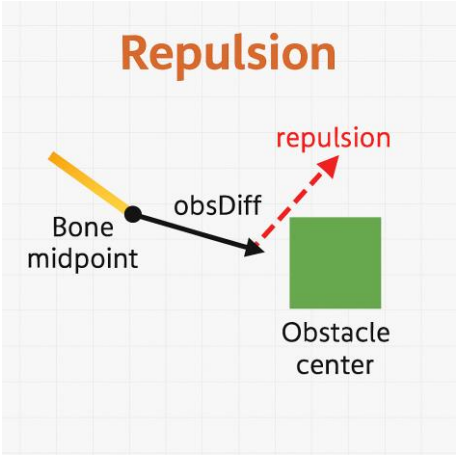
Obstacle Avoidance-Repulsion Mechanism

- When a bone is too close to the obstacle, we apply a **repulsive vector** to push the bone away from the obstacle.

- This repulsive vector is added to the **IK target direction** to bias the solution away from the obstacle.

repulse = normalize(obsDiff) * (threshold - dist)

obsDiff = bone midpoint - obstacle center



Objective

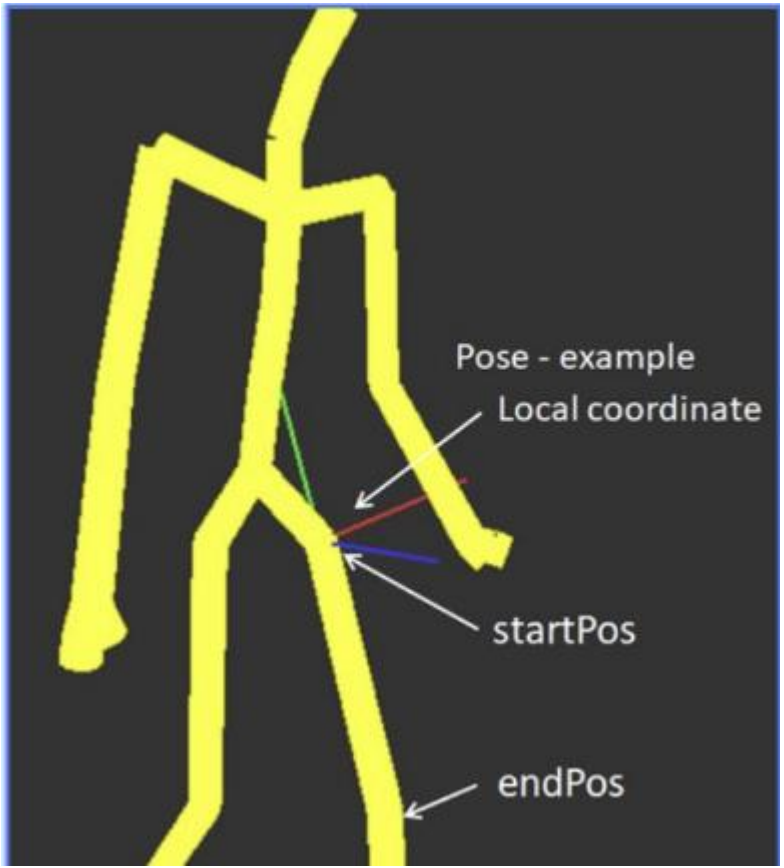
- Everything you need to implement is in kinematics.cpp
- There are three functions you need to implement in this homework
  - void forwardSolver(...)
  - VectorXd pseudoInverseLinearSolver(...)
  - bool inverseJacobianIKSolver(...)
- Bonus
  - Return whether IK is stable so that the skeleton would not swing its hand in the air
  - Take joint limits into account in bool inverseJacobianIKSolver

Implement Forward Kinematics: forwardSolver()

- Goal
  - Implement forward kinematics
- Convert motion data from joint space to the Cartesian space
  - set each bone's global start and end position and rotation
  - Hint
    - review "kinematics.pptx" from p.1 - p.19 (may be updated later)
    - review "acclaim_FK_IKnote.pdf" from p.1 - p.4 (may be updated later)
    - read local coordinate data from posture first
    - you can probably use DFS or BFS to traverse all bones
    - you can check
      - struct Posture in posture.h
      - struct Bone in bone.h

Forward Kinematics (cont.)

- Pose example
- Each bone has
  - local coordinate
  - start position
  - end position



VectorXd pseudoInverseLinearSolver(...)

- Goal
  - Find solution of linear least squares system, which will be needed for inverse kinematics
  - i.e solve min(|| J(θ_k) * ω_k - V ||) over ω_k
- Hint
  - You might use some pseudo-inverse methods such as SVD
  - There are some built-in functions in Eigen that you can use
  - Eigen::Matrixs4Xf means a matrix with 4 rows and unknown columns
    - Eigen::Matrix4Xf m(4, 10); // A matrix with 3 rows and 10 columns
  - Eigen::VectorXf means a vector with unknown size
    - Eigen::VectorXf v(10); // A vector with 10 elements

bool inverseJacobianIKSolver(...)

- Goal
  - Implement inverse kinematics
  - We use inverse-Jacobain method in this homework
- Hint
  - Review "kinematics.pptx" from p.20 - p.50
  - Review "acclaim_FK_IKnote.pdf" Inverse Kinematics part
  - Traverse from end bone to start bone
    - Make end bone touch the ball (target)
    - Start bone is the last movable bone, so you should stop at this bone
  - You can check struct Bone in bone.h

Report

- Suggested outline
  - Introduction/Motivation
  - Fundamentals
  - Implementation
  - Result and Discussion
    - How different step and epsilon affect the result
    - Touch the target or not
    - Least square solver
  - Bonus (Optional)
  - Conclusion

Grading

- Forward kinematics - 30%
- Least square solver - 10%
- Inverse kinematics - 40%
- Report - 20%
- Bonus - up to 15%

Submission

- Please upload only two files respectively
  - kinematics.cpp
  - report_< your student ID>.pdf
  - other necessary files (optional)
- Late policies
  - Penalty of 10 points on each day after deadline
- Cheating policies
  - 0 points for any cheating on assignments
- Deadline
  - Sunday, 2025/04/30, 23:59

Note

- Read TODOs in the template and follow TODOs' order

```
// TODO#1: Forward Kinematic
// Hint:
// - Traverse the skeleton tree from root to leaves.
// - Compute each bone's global rotation and global position.
// - Use local rotation (from posture) and bone hierarchy (parent rotation, offset, etc).
// - Remember to update both bone->start_position and bone->end_position.
// - Use bone->rotation to store global rotation (after combining parent, local, etc).
```

- How to contact TAs?
  - please ask your questions on new E3 forum or send email to ALL TAs via new E3
  - if you need to ask questions face-to-face, please send an email for appointment