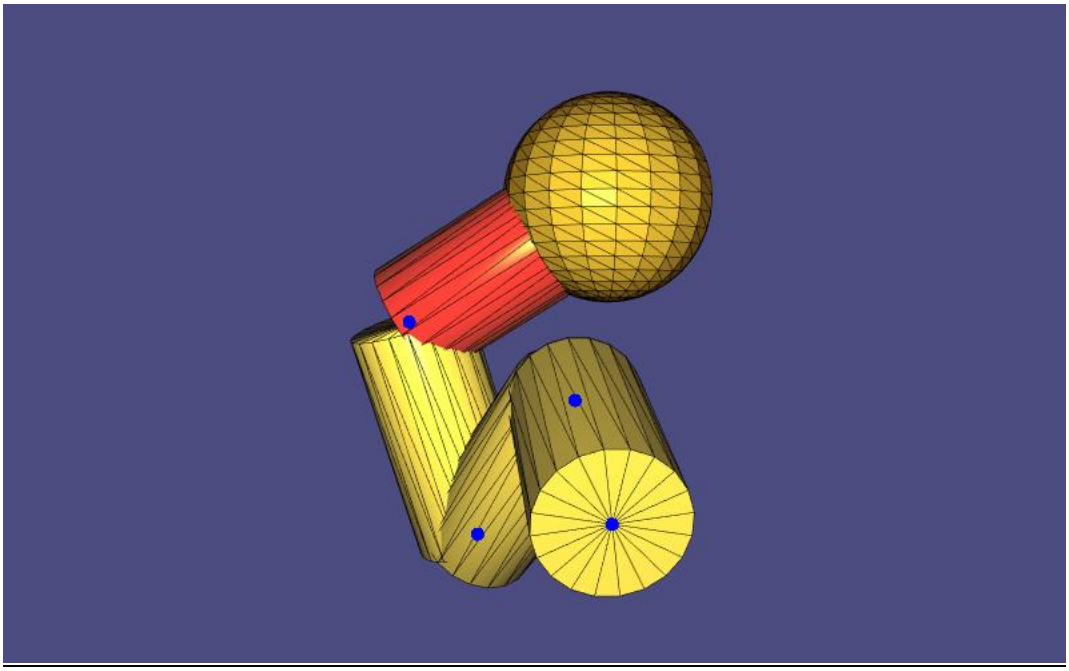
Assignment 3 - Kinematic Chain and Inverse Kinematics



Part 1 – System initialization

- Use the sphere.obj mesh to render a sphere in (5,0,0) and zCylinder.obj to render multi-links arm (located along the Z axis).
- Draw axis in the rotation centre of every cylinder according to the system of the previous link (use overlay edges, see Libigl tutorial 105). Axis length is twice link length.

Part 2 – data structures (recommendation)

- Save links number
- Save tip position (in global coordinate)
- Save distention (sphere) position

Part 3 – Requirements

1. **Add axis system for each joint** according to the previous link axis system
2. **Add to callback_key function:**
    - 'space' – starts and stops IK solver animation.
    - 'p' – prints rotation matrices (phi, theta) of the picked link. If no link is picked prints the rotation matrix of the whole scene.
    - 't' prints arms tip positions.

- 'd' prints destination position
- 'right and left arrows' – rotates picked link around the previous link Y axis (the first link will rotate around the scene Y axis). When nothing is picked rotate the whole scene.
- 'up and down arrows' – rotates picked link around the current X axis (use Euler angles). When nothing is picked rotate the whole scene.
- **Axis of rotations must stay the same before and after IK solver action.**

3. **Mouse callbacks:**
   - Change `glfw_mouse_scroll` callback to translate the picked object away and to the camera (perpendicular to camera plane). When no object is picked translate the whole scene.
   - When one link of the arm is picked and being translated move all the arm accordingly. **The arm must not break!**
   - Left mouse button will rotate objects or the scene in the same manner of the arrows
   - Right mouse button will translate the whole scene or the picked object.
4. **GUI**
   - When a cylinder is loaded it will be added to the end of the arm as an additional link (change the arm tip accordingly).
5. **IK implementation:**
   - Each joint has 2 or 3 degrees of freedom (for your choice).
   - Implement Cyclic Coordinate Decent method for IK solver (use rotation around vector). (70% of the grade, if Euler angles are implemented properly)
   - Implement FABRIK method for IK solver (use rotation around vector). (additional 30%) . Try to use additional array to save the positions of links during the calculations, instead of breaking the arm. Use CCD implementation as a basis for the FABRIK implementation.
   - The chain tip reaches the destination when the distance between them is smaller than delta = 0.1. print the distance at when destination is reached.
   - The solver **will not** start when the destination is too far. In this case it will print "cannot reach".
   - Use small steps when calculating angles.
   - When you use acos(A) make sure A's value is between -1 and 1.

6. **Bonuses:**
    - 5 points for adding constrains such that the minimal angle between 2 links in the chain will be 30 degrees.

Part 4 – Submission

- If you did the bonus add a readme file which explain what you did and how to activate it.
- Zip text file with link to Github repository and exe file of your solution. The zip file name will be ID1_ID2.zip