CMPT466-985 Computer Animation

Programming Assignment 2: Inverse Kinematics (20 points)

Due date: **23:59 Wednesday October 30, 2019**

In this assignment, you are going to implement Inverse Kinematics for the skeleton's **right arm** using two methods: 1) CCD method; 2) Jacobian method. The default coding environment is Visual Studio 2017 Community (workload: Desktop development with C++). Please ensure that your submission is runnable on the desktops with VS2017 (**x64 compiler**) and Windows 10 installed in ABS9840.

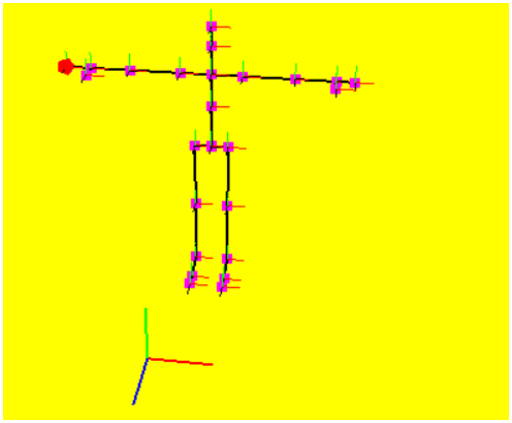
Folder Structure

The template is given as `CMPT466-985 Program Assignment 2.zip` and a demo is given with the zip file. You can firstly play with the exe file inside the `ResultDemo` folder. **Visual Studio 2017** solution file path is `build/CMPT466-985 Program Assignment 2.sln`.

The source code files are structured as follows:

1) `main.cpp`: main workflow and OpenGL functions**.**
2) `Skeleton.h/cpp`: load/store skeleton and motion data.
3) `ForwardKinematics.h/cpp`: calculate each joint's global position and orientation given skeleton and motion data. The Joint class is also located here. Check more details in source code
4) `InverseKinematics.h/cpp`: calculate each joint's local quaternions with given a given target object. **This is the only file you need to change to complete this assignment.**
5) `Vectors.h`: vector manipulation functions (e.g., dot product). Feel free to use them.
6) Eigen library environment has been setup for building Jacobian matrix, check more details on Eigen C++ library's official web: Link.

How to Control



1) You can switch between CCD mode and Jocobian mode by pressing `c` and `j`, respectively.
2) You can rotate your camera around the character by pressing `n` and `m`.
3) You can change the target's position by presing number keys:
   a. `1` and `2` for moving left / right along X axis;
   b. `3` and `4` for moving up / down along Y axis;
   c. `5` and `6` for moving in / out along Z axis.
4) You can reset the skeleton to T-pose and target to the position of the right hand in T-pose by presing `r`.

Project Structure

The main function in **`main.cpp`** contains the workflow. We have completed the code for displaying an initial T-Pose skeleton with a given target location (The red sphere on the character's right hand). The global x,y and z axes' directions are also given as three perpendicular lines (colored with red, green and blue) in front of the skeleton. You only need to complete the following two functions:

1) **`void InverseKinematics::CCDMode()`**
2) **`void InverseKinematics::JacobianMode()`**

IK Workflow

The Inverse Kinematics workflow is implemented in **`void InverseKinematics::IK().`** Each time the target's position is changed by pressing a number key, **`void InverseKinematics::IK()`** would be called. According to current selected mode, **`CCDMode()`** or **`JacobianMode()`** is executed for a maximum of **`iterNum`** times (initialized to **`300`**). When the distance to target is smaller than **`threshold`** (initialized to **`0.5f`**), the iterative IK is considered converged and will stop.

```
//IK workflow
void InverseKinematics::IK() {

  //check if the endeffector and the target are close enough
  Vector3 endPos = Vector3(endEffector->GlobalPos.x, endEffector->GlobalPos.y, endEffector->GlobalPos.z);
  Vector3 tarPos = Vector3(target->x, target->y, target->z);

  int i = 0;
  while ((endPos - tarPos).length() > threshold && i < iterNum)
  {
    if (mode == 0)
    {
      CCDMode();
    }
    else if (mode == 1)
    {
      JacobianMode();
    }
    endPos = Vector3(endEffector->GlobalPos.x, endEffector->GlobalPos.y, endEffector->GlobalPos.z);
    i++;
  }
}
```

Grading

There are two functions need to be completed within **InverseKinematics.cpp** as follows. The **CCDMode()** and **JacobianMode()** functions:

```
//CCD Mode IK for right arm
void InverseKinematics::CCDMode() { ... }

//Entire Right Arm Jacobian IK
void InverseKinematics::JacobianMode() { ... }
```

What you have:

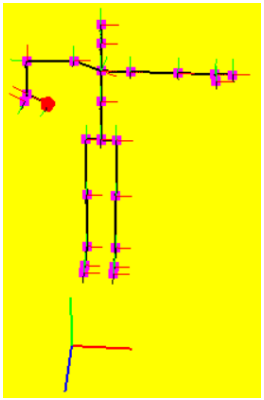
1) **endEffector** is initialized to **R_Wrist_End** joint;
2) **base** joint is initialized to **RightShoulder** joint;
3) **target** is initialized to the target location.

```
Joint* endEffector;
Joint* base;
Target* target;
```

Coding Part 1): 10 /20 points

```
//CCD Mode IK for right arm
void InverseKinematics::CCDMode()
{
    //add your code here
    //hint: Do forward kinematics when the endeffector's global position need to be updated
    //with newly computed quaternions.

    //I.e., use forwardKinematcis->forwardKinematicsComputation(base)
    //whenever you need to update endeffector's global position.
}
```

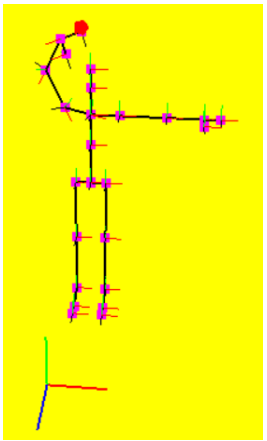


You need to calculate all the global quaternions of the joints on the right arm (from **RightHand** to **RightShoulder**) to reach the current target using CCD.

Coding Part 2): 10 /20 points

```
//Entire Right Arm Jacobian IK
void InverseKinematics::JacobianMode()
{
    //add your code here
    //hint: Do forward kinematics when the endeffector's global position need to be updated
    //with newly computed quaternions.

    //I.e., use forwardKinematcis->forwardKinematicsComputation(base)
    //whenever you need to update endeffector's global position.
}
```



You need to calculate all the global quaternions of the joints on the right arm (from **RightHand** to **RightShoulder**) to reach the current target using Jacobian-based approach.

Submission

Please submit the **`InverseKinematics.cpp`** file. Please include your student number and your name at the top of the source file in the comment section. Please make sure the code is runnable in VS2017 community under Windows 10 environment before you submit it.

TA will test your code on the desktops in ASB9840 with VS2017.

Note:

If you meet with following error:

```
Error  MSB8036   The Windows SDK version 10.0.16299.0 was not found.
```

Go to **`Project->Property`** Open the window, change **`Configuration_Properties->General->Windows SDK Version`** to change the SDK version to your VS2017 SDK version.