CMPT466-766 Computer Animation

Programming Assignment 2: Forward Kinematics (15 points)

Due date: **23:59 Sunday Mar 6, 2022**

In this lab, you are going to apply Forward Kinematic method with motion data obtained from a BVH file. The default coding environment is Python and PyOpenGL.

Where to code?

The template is given as `CMPT466-766 Program Assignment 2.zip`. Extract it. Open the project in your favorite editor.

1) `BvhParser.py` Do not edit this file.

   This file (as the name suggests) parses the Bvh file and creates the necessary skeleton data.

2) `utils.py` and `quaternions.py` Do not edit these files.

   Contains utility functions for quaternion and other manipulations. You can visit these files to reuse code

3) `main.py` Do not edit this file.

   This is the starting point of the code. This contains method to render stuff and hold the necessary OpenGL code.

4) `ForwardKinematics.py` The only file you need to modify.

There are **three sub-functions** needed to be completed within the recursive function in `ForwardKinematics.py`

```
calculateJointPosRecursivelyWithQuaternion()
|-- computeLocalQuaternion(joint: BvhNode)
|-- computeGlobalQuaternion(joint: BvhNode, localQuat: np.ndarray)
|-- computeGlobalPosition(joint: BvhNode)
```

This function would access every joint in a depth-first traversal of the skeleton tree and calculates every joint's global position and orientation for one frame.

1) `mocap` is an instance of class **Bvh** that hold all information about the skeleton data in `running.bvh`.
2) `pFrame` is pointing to the index for the `frame_data` which holds the data for the current frame.

You may need to:

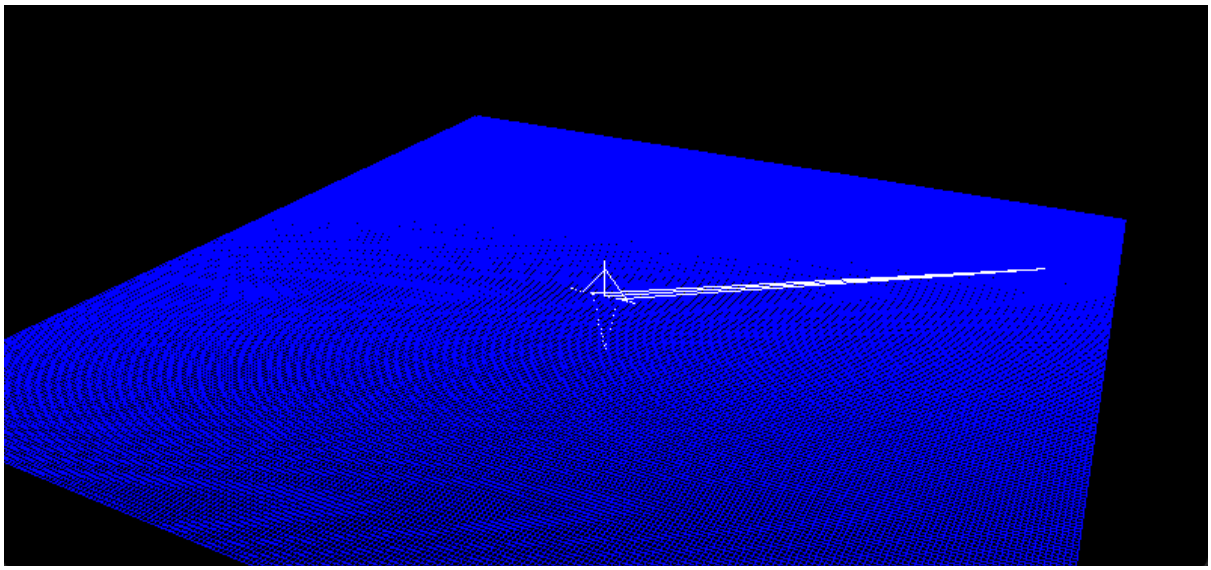
1) use `ndarray` class to represent quaternions. check more details in `quaternions.py`.
2) use `quaternionMultiplication()` to calculate quaternion multiplication
3) use the default skeleton tree's root joint (`root`) and `pFrame` to calculate every joint's global position (in quaternions).

Introduction to the template

A BVH file contains two parts: the Hierarchy part describes the skeleton tree; and the Motion part stores all joints' Euler angles, frame by frame. The skeleton tree structure and motion data of the bvh file are loaded into an instance of the `Bvh` class. Each **joint** in the skeleton data is a `BvhNode`. Some important fields to note here are -

1. `local_translation`: local position of the joint wrt to the parent joint
2. `global_position`: global position of the joint
3. `global_quaternion`: quaternion value to represent the global orientation of the joint

The overall workflow is as follows in the main function:
Each time key 'n' on the keyboard is pressed, one more frame is displayed. When you first run the code you would just see a scene like this. Here if you continue pressing 'n' you would observe one joint moving but the rest remaining static.



Coding Part

Part 1) `computeLocalQuaternion(joint: BvhNode)` (5 /15 points)

Input:
Joint: BvhNode
Output:
4-dimensional ndarray to represent joint's local rotation in quaternion

Every three adjacent elements within `frame_data (indexed by pFrame)` indicate three rotation angles along local x, y, z axis, in a specific rotation order in Euler angle (opposite to fixed-angle rotation order).

Examples:
Joint(Hips):
Rotation_order: joint->rotationOrder (e.g., XYZ=7).
Rotation_angles:
X: frame_data[0] rotation angle along X axis,
Y: frame_data[1] rotation angle along Y axis,
Z: frame_data[2] rotation angle along Z axis.
Joint(LeftUpLeg):
Rotation_order: joint->rotationOrder (e.g., ZYX=1).
Rotation_angles:
Z: frame_data[3] rotation angle along Z axis,
Y: frame_data[4] rotation angle along Y axis,
X: frame_data[5] rotation angle along X axis …

1) You can get current joint's rotation order by `joint.rotation_order`. The rotation order structure is as follows:

```
{ NONE = 0, ZYX = 1, YZX = 2, ZXY = 3, XZY = 5, YXZ = 6, XYZ = 7 }
```

Note that when rotation order is **NONE**, you shouldn't move **pFrame** forward.

2) You can get current rotation angle from `frame_data` by `frame_data[pFrame]`, which is global variable within scope of **ForwardKinematics**.

Note: don't forget to update **pFrame** after getting current rotation angle, or you will get root rotation angle for all joints.

3) Look for appropriate methods in **quaternions.py** to help you build the quaternions from the rotation angles and other useful functions.

Part 2) `computeGlobalQuaternion(joint: BvhNode, localQuat: ndarray)` (5 /15 points)

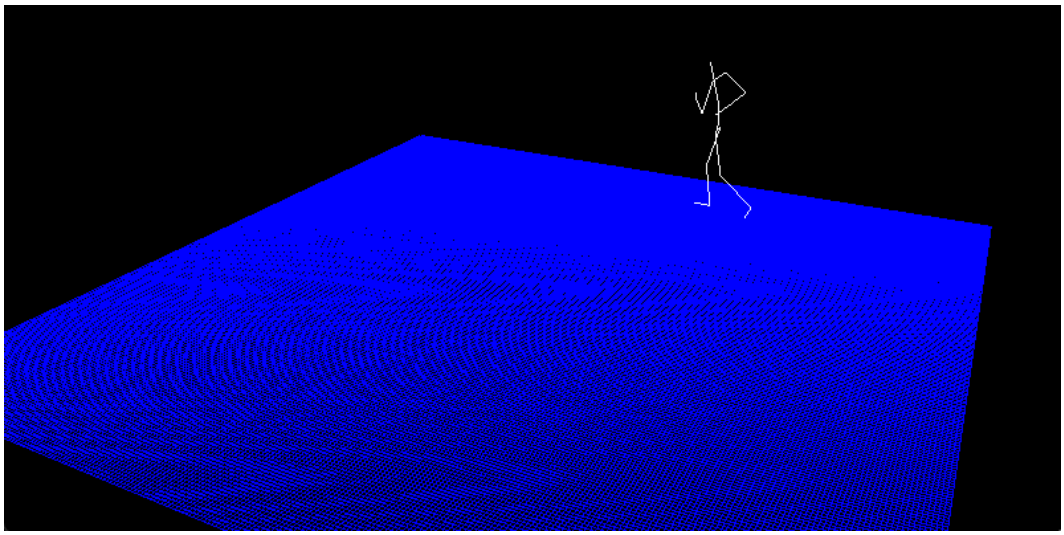
Input:
    Joint: BvhNode, localQuat: 4-dim ndarrat
Output:
    Joint's global orientation in quaternion

The local rotation for each joint is its rotation with respect to its parent. The global orientation for one joint is the accumulated rotations from the root.

(Note: Remember you are coding in a recursive function, so you just need to deal with **current joint's local quaternion** and its **parent joint's global quaternion**).

(Note: Be careful with root)

Part 3) `computeGlobalPosition(joint: BvhNode)` (5 /15 points)



If you complete part 3, the skeleton runs in circles when the 'n' key on the keyboard is pressed.

Submittion

Please submit a zip file with student number and your name (i.e., **`300000001_TerryFox.zip`**). The zip file should contain all the python files (edited or unedited) as well as the .bvh file.