# CMPT766 - Computer Animation

## Programming Assignment 2:

## Forward Kinematics (10 points)

Due date: **23:59 Tuesday Oct 17, 2024**

Marks for late assignments will be discounted by 10% each day. That is, if you are one day late, your marks will be multiplied by 0.9, two days late by 0.9*0.9…

In this assignment, you are going to apply Forward Kinematics to compute a skeleton's joints locations from the motion data obtained from a BVH file. You will also compute the joint and bone positions, scales, and rotations required for the character's final render. The coding environment will be in Unity.

## Unity Setup

To begin, download and extract **CMPT766-Assignment2.zip**. Open the extracted folder **CMPT766-Assignment2** as a Unity project in Unity Hub and select **2022.3.46f1** as the editor version. Later versions should also work, but if you run into issues then use the listed version. It should take a while to open the project for the first time.

Once the project is opened, open the scene in **Assets** → **Scenes** → **Assignment2** to begin the assignment.

## Project Script files

All the C# script files are contained in the **Assets/Scripts** folder. Here are some brief details on what they do:

1. **Tools/BVHParser.cs** Do not edit this file.
   This script (as the name suggests) parses a .bvh file and reads the motion captured data.
2. **Actor.cs** Do not edit this file.
   This script contains the main class of **Actor** and **Joint**.
3. **ActorRenderer.cs** File you need to modify.
   This script helps render the skeleton of the Actor in the Unity player.
4. **CameraController.cs**
   This script controls the camera movement. If you like, you may modify this script to have the controls bind to different keys.
5. **ForwardKinematics.cs** File you need to modify.
   This script contains a function that uses forward kinematics to update joints positions to the global space.

# BVH file structure

Here, we will briefly explain how a BVH motion capture data file works. There are two main parts in a .bvh file, the **HIERARCHY** part and the **MOTION** part.

The **HIERARCHY** section defines the joints of the actor, where the hierarchy of the joints is defined by the curly braces. Each joint has an offset position that is relative to its parent's position.

The root joint has 6 degrees of freedom (3 for translation, 3 for rotation, defined by the channels attribute), while other joints only have 3 degrees of freedom for rotation. The end joints do not have any degree of freedom, and they are there only for ending the skeletal structure.

The **MOTION** section defines the actual motion capture data. Each single line represents the frame of the motion, and each value is the degree of freedom. For example, the underlined values in the snippet below are the translation and rotational values for the **Hips** root joint in the first frame. The next 3 values are then the rotational values for the **LeftUpLeg** joint, and so on.

The rotational values will be in **Euler angles**, and the order is determined by the order defined in the channels attribute.

# What to code?

Each joint read from the BVH file is represented in a **Joint** object.
Some important fields to note here are:

1. **LocalPosition**: The joint's position w.r.t to its parent's joint
2. **LocalQuaternion**: The joint's orientation w.r.t to its parent's joint
3. **GlobalPosition**: The global position of the joint
4. **GlobalQuaternion**: The global orientation of the joint
5. **RotateOrder**: The order of the Euler angle when reading the frame data

You can also get the parent of the joint via the **GetParent()** method, and the children's joint via **GetChildren()**.

The **Actor** object contains a list of joints that can be assessed through the **Joints** property. You can get the root joint via the **GetRootJoint()** method.

1. Please complete the following function in **ForwardKinematics.cs**:

   **public static void UpdateJointPositions(Actor actor, float[] frameData)**

The first input **actor** is an object of type **Actor**. The second input **frameData** is a float array that contains the motion values of a single frame as described in the section above.

Using quaternion algebra only, for every joint you are tasked to:

- Update **Joint.LocalQuaternion** (4/10 points)
- Update **Joint.GlobalQuaternion** (1/10 points)
- Update **Joint.GlobalPosition** (2/10 points)

Keep in mind that the rotation values from the file are in **Euler Angles**, and BVH files can have different rotation order. Your code should work correctly for different rotation orders. It is also helpful to look at the default rotation order in Unity, depending upon the functions you decide to use.

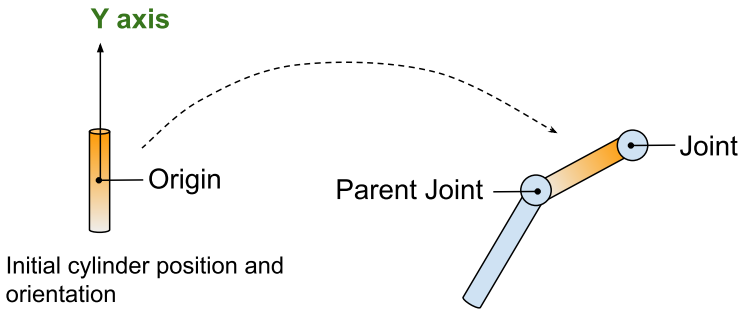
2. Please complete the following function in **ActorRenderer.cs**

   **public static void UpdateSkeletonPosition()**

This function calculates the joint and bone positions for the actor's final render. You are tasked to:

- Update each element of **JointSpheres'** position (1/10 points)
  - Joints are rendered as spheres. Using the actor's joint data, compute their positions.

- Update **Bone positions and rotation** (2/10 points)
  - The bone geometry is represented by cylinders in Unity, initially positioned at the origin and oriented along the Y-up direction. You need to update the bone position and orientation from its parent joint's location and orientation.

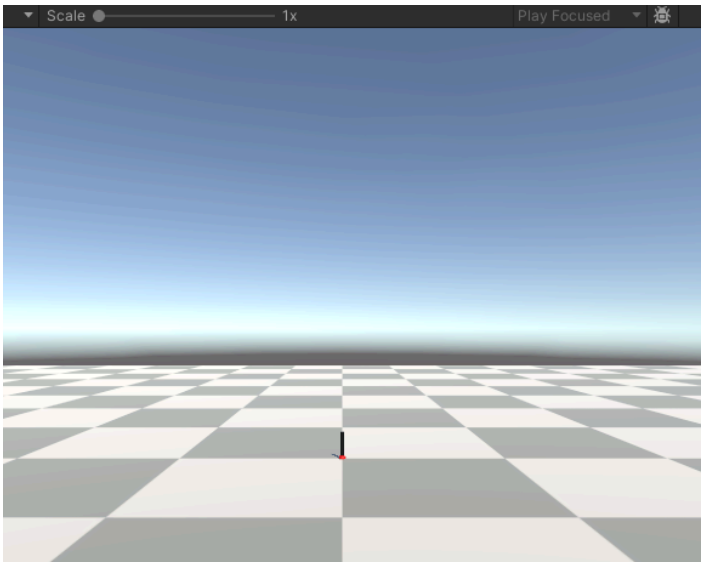



## Additional notes

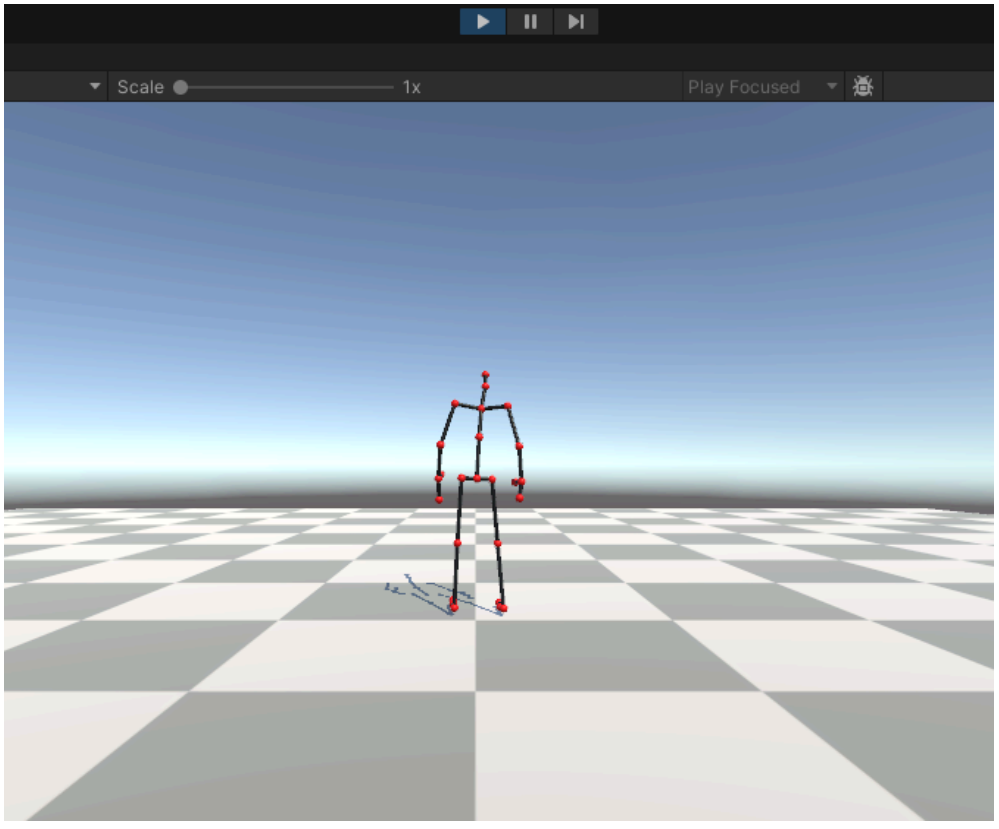
- Code to apply **ActorScale** is already implemented. You do not need to use **ActorScale** in your computations. Do not remove **ApplyActorScale()** from the method.
- Do not make changes to anything other than **UpdateSkeletonPosition()**
- **ActorScale** and **BoneWidth** are automatically calculated to have a reasonably sized character appear on the screen. You can tweak these values in the inspector if necessary.
- Your code will be tested with the BVH files included in **Assets → Resources → Bvh**

# Running the Code

Press the play button in the unity editor to run the code. Before you implement **ForwardKinematics.UpdateJointPositions()**, and **ActorRenderer.UpdateSkeletonPosition()**, you'll see a screen where all joints and bones of the skeleton collapse to the origin.

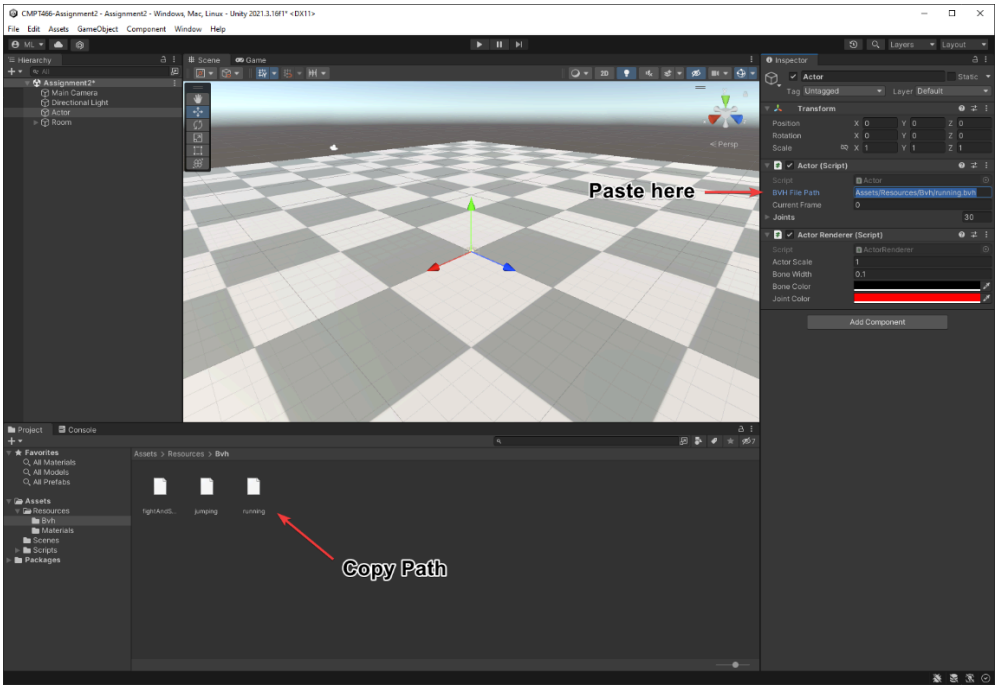



Once you have completed the implementation, you should be able to see a skeleton performing jumping motion.



You can move the camera with **W**, **A**, **S**, **D** keys, go up and down with **Q**, **E** keys, and pan the camera by **right clicking and dragging** the screen. Press **space** to pause the animation.

You can also run other motion captured files. Go to **Assets** → **Resources** → **Bvh** and select a BVH file, right click to open a drop-down menu and click copy path, paste the path in Actor GameObject's inspector BVH File Path field.




# Submission

Please create a zip containing your **ForwardKinematics.cs** and **ActorRenderer.cs** files. Your file should be named as **studentid.zip (e.g. 123456789.zip).**

At the top of your files, please leave a comment that includes your **name** and **student id**.