COSC422 Advanced Computer Graphics
Programming Exercise 15
Skeletal Animation

This programming exercise provides an overview of the implementation of fundamental skeletal animation techniques using Assimp. After completing this exercise, you will be able to animate skeletal structures using motion capture data stored in Bounding Volume Hierarchy (BVH) format. Three BVH files, "Dance.bvh", "Walk.bvh" and “Boxing.bvh” are provided with this exercise. A test file, "Test.bvh" containing a very simple 3-link skeletal structure and a 10 frame motion sequence is also included (Slides [10]-16,17).

The Carnegie Mellon University's motion capture dataset contains animation data for around 2500 motion sequences. Files from this dataset can be downloaded from here: https://sites.google.com/a/cgspeed.com/cgspeed/motion-capture

Important Note: Recent versions of Assimp (version 4.1.0 and above) contain an error in the BVH loader module that causes BVH animations to be displayed incorrectly. The sequence of Euler angle rotations was wrongly calculated in the implementation file. The correct version of BVHLoader.cpp is provided in the programming exercises section on Learn (“13. Setting Up Assimp..”). Please go through the installation notes given in the section (updated 3rd Oct) before installing Assimp.

SkeletalAnimation.cpp:

This is a simplified version of the program `AssimpModelLoader.cpp` (Exercise 14), designed to load files and BVH format and to display the skeletal mesh models generated by Assimp. The program does not contain any functions to load textures, and assumes that the model is a triangular mesh. It uses the following post processing preset that generates separate mesh objects for each link of the skeleton.

```
aiProcessPreset_TargetRealtime_MaxQuality | aiProcess_Debone
```

When Assimp loads a BVH file, it will create a mesh structure, the contents of which can be printed out by un-commenting the lines, "`printSceneInfo()`" and "`printTreeInfo()`" in the `initialise()` function. Assimp generates a node hierarchy for the mesh objects with initial transformation matrices in each node computed using the corresponding joint offsets.

A BVH file can contain only one animation sequence, and the motion data corresponding to this sequence can be accessed as below:

```
aiAnimation* anim = scene->mAnimations[0];
```

The animation channels (position keys and rotation keys for each node) can be obtained from the above animation object. Please refer to slides [10]: 27-35 for information on using motion data for animating skeletal structures. The first step towards generating an animated model is the creation of a timer callback function that increments a “tick” variable from 0 to the duration of the animation (see Slide [10-33). The call-back function should also update the transformation matrices in the node hierarchy by calling a function `updateNodeMatrices()`. This function constructs the transformation matrices from the position and rotation keys of each channel, and replaces the corresponding node’s transformation matrix with the new product matrix (see Slides [10]-34,35).

If the above steps are implemented correctly, you will be able to load files in BVH format and generate skeletal animations using their motion data. A few sample outputs are shown below.

Walk.bvh

Dance.bvh

Boxing.bvh

[10]: COSC422 Lecture Slides: "Lec10 Motion Capture Data and Skeletal Animation"