COSC422 Advanced Computer Graphics

Assignment 2

3D Character Animation

Due: 11pm, Friday, 19 October 2018

Maximum Marks: 25

I. Introduction:

Skeletal animation algorithms form an important part of rendering applications involving three-dimensional character models. In this assignment you will implement skeletal animation algorithms using the Open Asset Import Library (Assimp) to render scenes containing character animations. The assignment consists of three tasks, and *all three* tasks must be completed: (i) skeletal animation using only motion capture data (ii) animation of a rigged character model using a separate motion file, (iii) animation of a rigged character model using an embedded animation sequence. Each of these tasks is outlined below.

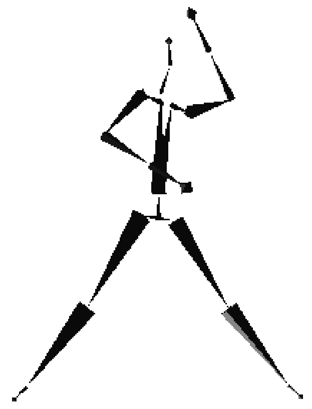	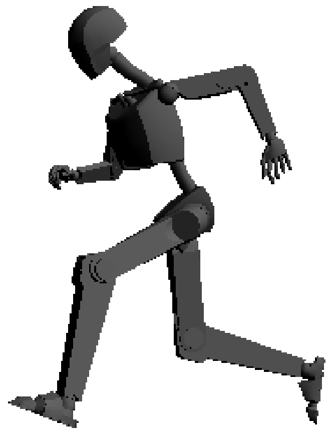	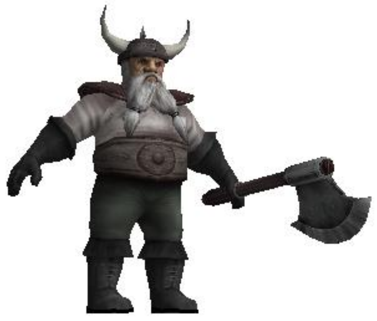
Fig. 1.	Fig. 2.	Fig. 3

II. Task Description:

Task 1: Skeletal Animation using BVH Files: BioVision Hierarchy (BVH) is a popular file format for storing motion capture data. A BVH file contains the defintion of a joint hierarchy and a series of keyframes. When a BVH file is loaded, Assimp generates a dummy model of a stick figure (Fig. 1) with associated mesh/vertex data. Your task is to create an animation of this model using the keyframes in the BVH file. Two sample BVH files (Dance.bvh, Boxing.bvh) are provided. Also included is a file (Test.bvh) which you could use for tesing purposes. This file contains a simple 3-link model, and a set of 10 keyframes which you can easily edit and use for verifying the correctness of your results. You may also use BVH files downloaded from sources on the Internet. Two such websites where motion capture data are available for download are:

Cgspeed: http://www.cgspeed.com/

Free Motion Capture Data: http://www.motioncapturedata.com/

The main requirements of this task are:

1.(a) The scene should contain at least one stick figure (dummy mesh) model generated by Assimp, and animated using the keyframes of a BVH file.

1.(b) The scene should contain a floor plane and a few other objects that are relevant to the animation sequence. For example, if the animation displays a walk sequence, the scene could include a walkway. These objects can be easily modelled in the display() function using glut objects and legacy OpenGL glBegin()-glEnd() blocks.

1.(c) The camera should follow the animated object so that the model always stays within the field of view.

It is not necessary to use texture mapping for any of the objects.

Task 2: Animation of a rigged character model using a separate motion file: Certain character model file formats (eg. FBX) store mesh data and the associated motion data in two separate files. You are required to develop a program that uses Assimp data structures and functions to load both files and animate the rigged character model using the animation sequence stored in the motion file. A sample mesh model (Fig. 2) and a set of motion files in FBX format are provided. You may use any other 3D character model files and animation data for this purpose.

You might find the following 3D model viewer useful for your work. It can display the node and joint hierarchies for character model data and motion files in various formats.

Open3D Model Viewer: http://www.open3mod.com/

The main requirements of this task are:

2.(a) The scene should contain at least one character model animated using its motion file.

2.(b) In order that lighting calculations work correctly, proper transformations must also be applied to surface normal components of the mesh.

2.(c) The scene should contain a floor plane and a few other simple objects. These objects can be generated inside the display() function using glut objects or using glBegin()-glEnd() blocks.

2.(d) The camera should follow the animated object so that the model always stays within the field of view.

It is not necessary to use texture mapping for any of the objects.

Task 3: Skeletal Animation of a rigged character model: A few character model file formats allow animation data to be embedded within the mesh file itself. Two such models in DirectX format (wuson.x and ArmyPilot.x) are provided. “wuson.x” contains two animation sequences, while “ArmyPilot.x” has several animation sequences. You are required to develop a program that uses Assimp data structures and functions to load and animate the character model using at least *two* animation sequences. The two sequences may be combined into one sequence (eg. Walk sequence followed by a run sequence). You may use any other 3D character model file that contains skeleton and animation data for this purpose.

The main requirements of this task are:

3.(a) The scene should display the character model animated using at least two embedded motion sequences.

3.(b) In order that lighting calculations work correctly, proper transformations must also be applied to surface normal components of the mesh.

3.(c) The scene should contain a floor plane and a few other objects. These objects can be generated inside the display() function using glut objects or using glBegin()-glEnd() blocks.

3.(d) The camera should follow the animated object so that the model always stays within the field of view.

3(e) Use texture mapping where necessary.

III. Report (Max. 4 pages; Max. marks: 4):

Please prepare a report describing your work, and include the following sections:

- A brief outline of the implemented methods, including any problems/challenges faced and how you attempted to solve them.
- A brief description of the background scene used for each animation.
- A few screenshots showing the outputs of both programs.
- A brief description of each animation sequence.
- The complete list of keyboard/mouse functions defined for user interaction.
- List of references to the sources of materials (textures, mocap files, model files) used.

IV. Program Development:

You may use math library functions, mesh models, and motion capture files that are available on the Internet or obtained from other sources such as books. Please acknowledge the source in your report. You may also use programs and other supplementary materials provided in the course (eg. ModelLoader.cpp).

Demo programs found on the Internet and other OpenGL resources should not be submitted as part of the assignment.

V. Assignment Submission

Submit your files using the assignment link on Learn (learn.canterbury.ac.nz) before 11pm on 19 October 2018. Your submission must contain:

1. The source codes and all supplementary files (bvh files, mesh files) needed to run your programs. Please do not include freeglut, opengl, glew, glm or assimp library files.

2. Your report in Word or PDF format.

Miscellaneous

1. Check regularly on the *Learn* system forums for spec updates and clarifications. You may submit up to one week late for a 15% penalty.

2. This is not a group project. Your assignment must represent your own individual work. In particular, students are not permitted to share program source code in any way. However, you may discuss ideas, implementation issues etc using the class forum on Learn.

3. Standard departmental regulations regarding dishonest practices and late submissions apply.