Computer Graphics Project 3: Bvh Viewer

Handed out: May 15, 2023

Due: 23:59, June 4, 2023 (NO SCORE for late submissions!)

- *LMS course home > Lecture Contents (강의콘텐츠) > Week 5> Project 1 > "Submit Assignment" Button> Upload your zip file*
- *Compress your files into a zip file as in the following example. The zip file can be named whatever you want.*

```
+ submission.zip
  - main.py
  - ...
  - report.pdf
```

- *Your program may consist of several python source files. But the main module should be in **main.py**. That is, your program should be executed with the following command:*

```
  python main.py
```

1. Implement your own bvh file viewer.

   A. You have to implement all requirements in a single program. This project DOES NOT require each requirement to be a separate program

   B. The window size doesn't need to be (800, 800). Use the larger window that is enough to see the details of the viewer.

   C. **Your program must use OpenGL 3.3 Core Profile**, meaning that …

      i. Your python code must include:

```
  glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3)
  glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3)
  glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE)
```

      ii. Your shader code must start with:

```
  #version 330 core
```

      iii. **You will not get any score for this project (except report) if you do not use OpenGL**

3.3 Core Profile.

D. Total points: 125 pts

2. Requirements

A. Manipulate the camera in the same way as in Project1 using your Project1 code (10 pts).

i. Also draw the reference grid plane.

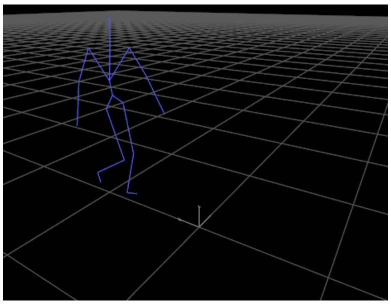
B. Load a bvh file and render it (110 pts)

i. Open a bvh file by drag-and-drop to your bvh viewer window **(10 pts)**

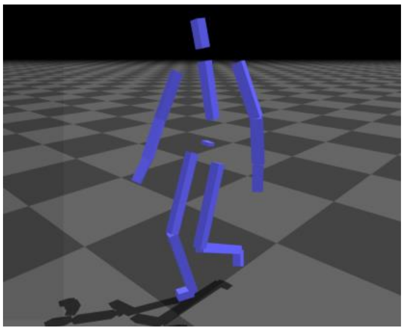
1. Google *glfwSetDropCallback* to see how to do it
2. The viewer should render only one bvh file at a time. If a bvh file B is drag-and-dropped to the viewer while it is rendering another bvh file A, the viewer should only render the new bvh file B.
3. **This feature is essential for scoring your assignment, so if not implemented, you won't get any score for "Load a bvh file and render it (100 pts)"**

ii. Provide **two rendering modes** – line rendering and box rendering **(30 pts)**

1. Line rendering (when you press the key '1') **(10 pts)**

A. 

2. Box rendering (when you press the key '2') **(20 pts)**

A. 

B. Render all boxes using Phong Illumination and Phong shading in this mode.

3. You should be able to change the rendering mode by pressing the above key **at any time during the execution of the program**, and the newly rendered model **should be displayed as soon as the mode is changed**.

iii. Read the bvh file and **render the "skeleton" (rest pose) of the motion when you load the file by drag-and-drop (30 pts).**

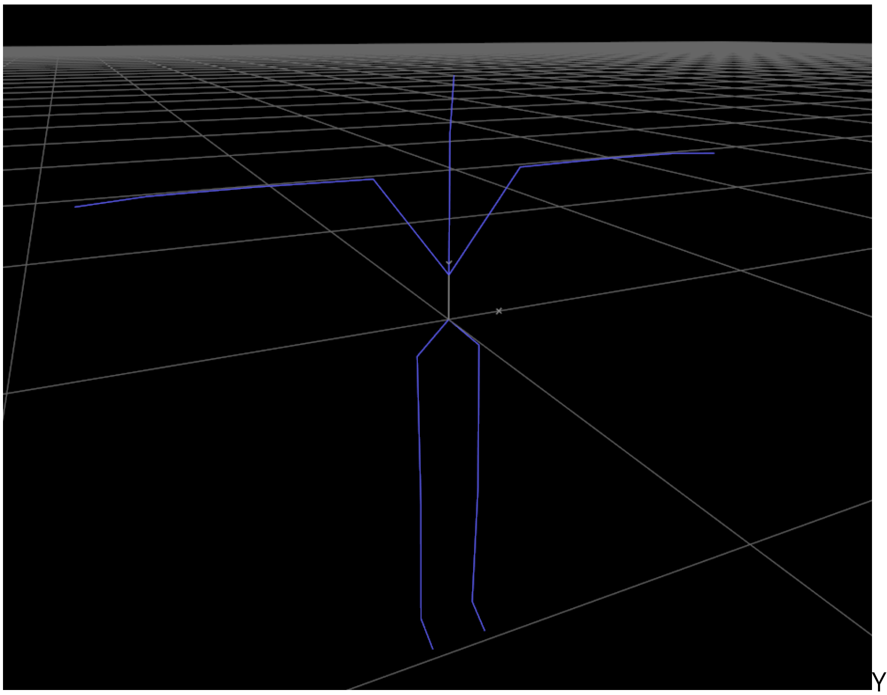
1. **Do not automatically animate the character.** Just draw its skeleton (rest pose) when you open a file.

2. Just draw joints with offsets between them only using the HIERARCHY section of the bvh file.

3. In other words, draw a pose of the motion with zero translation (0,0,0) and no rotation (identity matrix) being applied to transitional joints and rotational joints, respectively.

4. For end-effector joints such as foot, a line segmented should connect the end-effector joint and the "end site".

5. An example screenshot for sample-walk.bvh (in line rendering mode)

A.  YO

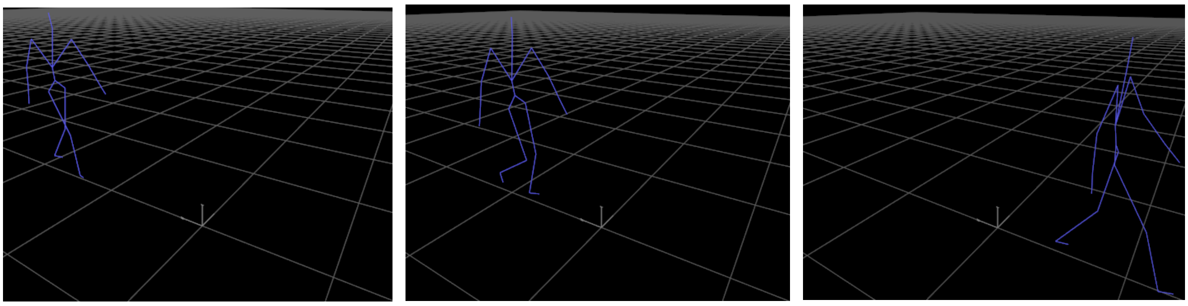
iv. **Animate the loaded motion if you press the <spacebar> key (30 pts)**.

1. As time goes by, draw each pose of the motion using each line of frame data in the MOTION of the bvh file, from the start frame to the end frame.

2. Progress one frame at a time whenever the **frame time** specified in the BVH file **elapses**. (If progressed one frame per iteration of the rendering loop, the animation would play too quickly, making it difficult to perceive.)

3. After drawing the last frame, just **replay** the motion again (draw from the first frame to the last frame again).

4. Example screenshots for sample-walk.bvh (in line rendering mode)



v. When open a bvh file, print out the following information of the bvh file to stdout (console) **(10 pts)**

1. File name

2. Number of frames

3. FPS (which is 1/FrameTime)

4. Number of joints (including root)

5. List of all joint names

3. Report (15 pts)

A. Submit a report of **at most 2 pages** in docx file format (MS Word). Do not exceed the limit.

B. The report should include:

i. Which requirements you implemented **(5 pts)**

ii. A hyperlink to the video uploaded to Internet video streaming services (such as YouTube and Vimeo) by capturing the animating hierarchical model as a video **(10 pts).**

1. Download bvh files from the Internet and open them with your viewer. For example,

A. http://motion.hahasoha.net/

B. http://mocap.cs.sfu.ca/

C. http://dancedb.eu/main/performances

2. Choose one of the best looking motion plays using your viewer.

3. **Do not use the sample bvh files for this requirement.** You must use other bvh files downloaded from internet to get score for this requirement.

4. **The uploaded video MUST be publicly accessible.** Otherwise, you won't get the 10 pts.

C. You do not need to try to write a long report. Just write down the required information. Use either English or Korean.

4. Runtime Environment

A. Your program should be able to run on Python 3.8 with only NumPy, PyOpenGL, glfw, PyGLM installed. Do not use any other additional python modules.

B. Only **glfw** is allowed for event processing and window & OpenGL context management. **Do not use glut functions for this purpose.**

C. If your program does not meet this requirement, it will not run on TA's computer so you will not get any score for this project (except report).

5. Test your viewer with sample bvh files.

A. sample-walk.bvh: A walking motion file

B. sample-spin.bvh: A walking while spinning around a vertical axis

6. What you have to submit: A zip file including

i. **.py files -** *Your program may consist of several python source files. But the main module should be in **main.py**.*

B. **.pdf report file**

7. Additional information

A. *drop_callback* in glfw python binding is slightly different from that of original glfw written in C. *drop_callback* in python takes only two parameters, *window* and *paths*. *paths* is a list of dropped file paths.

B. Python provides powerful string methods helpful for parsing a bvh file. Among them, split() will be most useful.