COMP 5823M: Animation & Simulation 2023-2024

ASSIGNMENT 2: Blending Animation Cycles **[30 marks]**

You have been provided with a simple interface derived from Assignment 1. For this assignment, you will implement animation cycles with blending. Details of user interface controls will be provided after the task description.

Code has been provided for a SceneModel that reads in four different animation cycles in .bvh format.

1. Rest pose with one frame of pose information
2. Running cycle
3. Veer left
4. Veer right

Each animation cycle contains the hierarchy (or “armature”) of the skeleton. For this exercise, we have made sure that they all share the same hierarchy. In the real world, this is not always true. We have not provided a surface model, as it is customary to animate **only** the skeleton in developing animation routines.

For full marks, you will need to complete ALL of the following:

**Task I: Rendering the Hierarchy Itself [6 marks]**

BVH files store the hierarchy first, followed by the data needed for animation. Instead of storing bones, they store joints. However, since the file specifies the child joints for each bone, this implicitly describes a bone from the origin (offset) of the parent to that of the child. Since the child’s offset is described in the parent’s coordinate system, all that is needed is to loop through the joints, rendering a cylinder from that location to that of the child.

To do this, you will need to find the offset of each joint, which is stored in the joint\_offset field of the Joint class, and redundantly in the boneTranslations array.

**Task II: Running In Place [6 marks]**

Once you can render the hierarchy, the next task is to use the data in the run cycle to animate the character. The animation data in the BVH file consists of a set of frames: in each frame are a set of “channels”, which store floating point values for position and rotation. Although quaternions are a better solution, BVH files store Euler angles, and the rotations specify degrees around Z, Y, X in that order.

You can retrieve the pose angles for a given frame from the boneRotations vector, indexing on the frame then the joint ID.

The animations are provided at 24 frames per second. To simplify life, the animation timer has been adjusted to fire 24 times per second.

**Task III: Running Forward** **[3 marks]**

While animation data can include translation, this often makes it hard to blend between animations, or to repeat them. You will need to choose a suitable amount to translate the character forward in each frame, and this location information should be kept separate from the BVH data.

**Task IV: Veering** **Cycles [5 marks]**

Two additional cycles have been provided: veering left and veering right. You will want to repeat Tasks II and III to get these animations working separately as well. These will require you to keep orientation as well as location information for the character.

**Task V: Blending Animations** **[8 marks]**

Once you have the individual animations working, you should modify the code so that when you hold a button down, the character should blend from the current cycle to the new one over a period of about 0.5s. It should then continue in that animation cycle until additional control keys are pressed. Since a rest pose is provided, you can bring the character to a stop by switching to the rest pose, and a key has been provided to reset to the original position.

**Task VI: Rough Terrain** **[2 marks]**

The terrain on which the character is running is not flat. As discussed in class, modify your code so that the character runs **ON** the surface, not through it.

**USER INTERFACE CONTROLS:**

In order to make it feasible to mark, you **MUST** use our choice of controls, even if you think you have a better solution. The following keys are defined:

WASD: Translate the camera position (like Assignment 1)

RF: Move the camera position up and down

QE: Rotate the camera position

P: Reset the character position

Up: Blend to Run Cycle

Down: Blend to Rest Cycle

Left: Blend to Veer Left

Right: Blend to Veer Right

All code should compile on the School's Linux machines without installation of any extra libraries or applications. You should include a makefile and a readme.txt file with any additional instructions.

**FILE NAMING:**

In order to make the marker’s job easier, please make sure that you rename the directory from A2\_handout to xxxxxx\_A2 where xxxxxx is your userID. For example, since my userID is scshca, I would rename it to scshca\_A2.

**PENALTIES:**

Poorly structured or badly commented code may be penalised by up to 25% of the marks available.

Code without a readme may be penalised by up to 10% of the marks available.

Code that does not compile properly will be assigned a mark of 0, but I will usually give the student one chance to correct this.

**DUE DATE:** Thursday, December 14, 2023, 10:00am