HIERACHICAL MOVEMENT PROJECT

By. Liangyi Hu

1. Project description

This is a project made for Lab2 of Computer Animation CS6555 in George Washington University.

The deliverables contain a video file demonstrating how hierarchical movement works, a Visual Studio project containing source code and other attachments (pictures, 3D objects). However, the library and dependencies are not included in the deliverables (relatively large), they are available upon request.

2. Features.

According to the project requirements, there are required features and additional features.

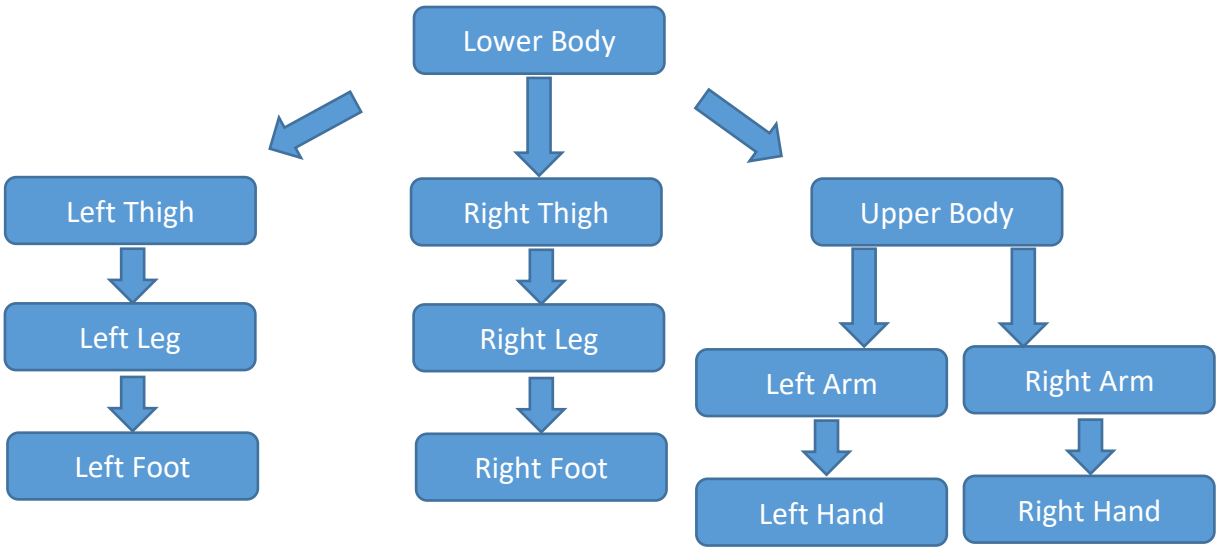
Required features:

- Moveable objects with thigh, leg and foot structure.

Additional features include but not limited to following:

- Controllable camera (First shooting game style, WSAD move camera up, down, left, right. Mouse movement control camera direction, mouse scroll to zoom in and zoom out).
- Importable multiple objects (Common 3D files with texture are accepted, e.g. File.dae, File.obj).
- Visualized control points (Control points showed in screen as polygon).
- Separate speed control (User specify speed factor for each curve).
- Configurable key call back system (E.g. press “B” to begin animation).
- Treed structure with 12 parts.
- Upper body rotates left and right when walking or running.
- Body goes up and down when running or walking.
- Arm swings when running or walking.
- Multiple state, running, walking and jumping.

Object hierarchical tree:



3. Code Structure

The project is using OpenGL as its environment, the API been used is GLFW.

Library Packages

The none standard libraries used are listed as below:

Package Name	Description
GLFW	OpenGL development tool, create Simple API for users to use OpenGL functions.
GLEW	The OpenGL Extension Wrangler Library is a simple tool that helps C/C++ developers initialize extensions and write portable applications.
GLM	OpenGL math library, provide some data structure and method in calculations
SOIL	Used to import image and convert to texture data
ASSIMP	Open Asset Import Library (short name: Assimp) is a portable Open Source library to import various well-known 3D model formats in a uniform manner.

Please check and install proper libraries if user want to run the program.

Source code header files

There are five header files written, the name and its description have showed in below table:

Name	Description
MatrixTransformatin.h	Containing almost all the functions and variables needed to build and use for key framing interpolation.
Background.h	Code construct background coordinate pictures and connected lines of key points for key framing.
Camera.h	Containing the method building View/Projection matrix, allows camera to respond from user actions.
Shader.h	Creating Shaders, source Shader code, compile Shaders. Also provide some methods to easily using shaders.
Model.h	Import 3D objects, extract information and provide some method to easily visualize them.

	Additionally, each part is a Model object that can compute its position and rotation during simulation. Lots of code related to project 2 are in here.
Mesh.h	Working with Model.h, create mesh vertices based on .obj file.

Note: Camera.h, Shader.h and Mesh.h are customized from source code written by Joey De Vries at www.learnopengl.com.

Source Files

The source file shows as below:

Source File Name	Description
Main.cpp	Main program entry
VertexShader.txt	Vertex Shader program for general object visualization.
FragmentShader.txt	Fragment Shader program for general object visualization.
bgVertexShader.txt	Vertex Shader for background pictures.
bgFragmentShader.txt	Fragment Shader for background pictures.

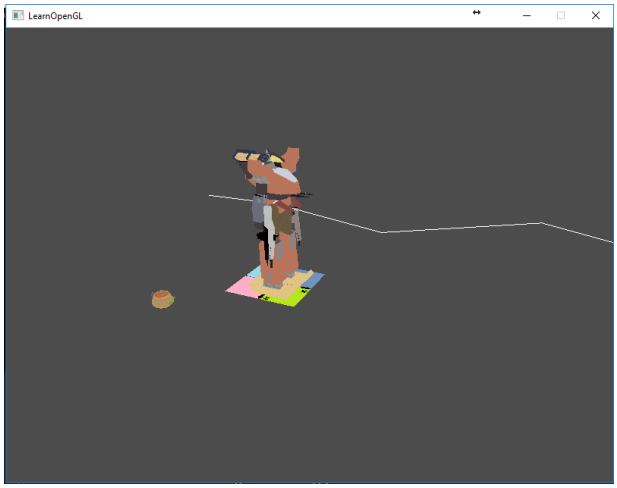
4. User Guide

The program could run under Visual Studio environment, after all the library dependencies are installed.

The steps to have it run is (in addition to Part 1):

- Define key framing points.
- Define state changing condition (a global enum variable called robotState).

When the program starts, user should be able to see following screen:



There is a robot stays on white line (key framing points), and also a background picture shows the X axle and Z axle (Y points above in this picture.

User will be able to use WASD key and mouse to control movement of the camera.

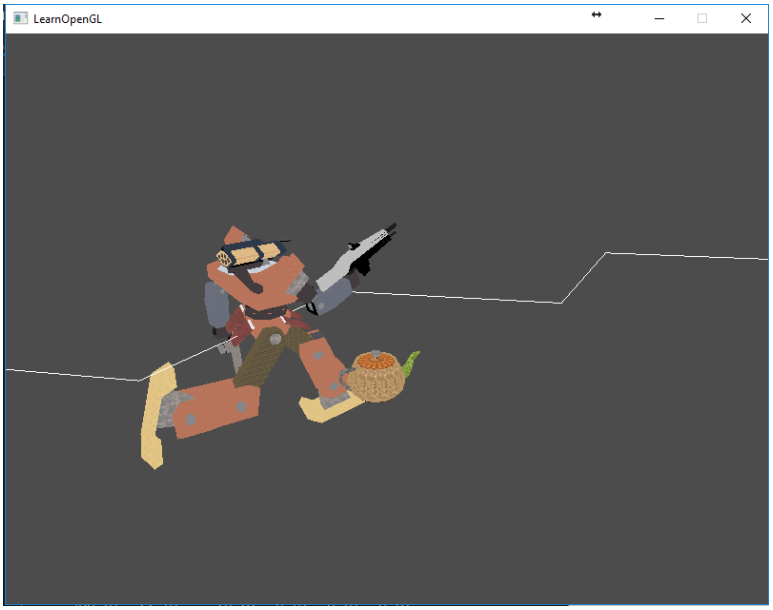
When pressing “B” (Begin) key, the robot will start move, it follows the white line, based on different interpolation method, it could have slightly different following behavior.

Currently the state change is defined as below:

Point 1 to point 9: running.

Point 9 to Point 17: walking.

Point 17 and after: jumping.



The robot will stop a little bit before the end of the line.