Assignment 3 Reflection - ArmSim

Name: Joshua Riddell
Student Number: 43947241

28 May 2015

Overall, I am very happy with how this project turned out. It has some small problems with usability and the class design but considering it is the first application I have made - besides simple scripts - it serves its general purpose quite well.

1 What I Have Learnt

- I have learnt a massive about about class design and how to use inheritance. I am very happy with my class design of the SequencerElements. The inheritance allows for easy addition of new modules with different settings and functions - following developers need only define the contents of the module, and its execute and get_value functions to get a whole new SequencerElements object.

- I spent the whole of the mid-term break trying to work out how to use OpenGL functions such as defining light sources, normals, vertices etc. I spent a lot of time on this but in the end I used an external library with the ability to import standard CAD files to OpenGL - so all of the time spent on manual vertex definitions was almost wasted. However, in the process, I have learnt a lot about how 3D graphics work on a reasonably low-level.

- I started this project with no experience in GUI design and as a result I prioritised 3D graphics and chose a library with little support for basic GUI design. As a result I have had to spend a lot of time on configuring window layout and some basic elements which are usually included in most GUI libraries. This massive amount of time wasted could've been saved by doing a little more planning and research about implementation at the start of the project - something I will definitely be doing next time. If I were to do this again then I would likely use the OpenGL bindings in MQTT and import vertices from a .obj using my own script.

2 Items To Improve

- Due to the slow update rate of pyglet_gui label elements, the sequencer modules only update with a change of 3 units. Reducing the update resolution was a poor solution which could be improved through the use of a regular scheduled callback which updates the labels to the highest accuracy every 0.5 seconds. This is again quite a poor solution however, use of MQTT would be a better fix.

- The startup speed of the program is quite poor. I suspect that this is because the .obj files contain quite a lot of vertices which are all iterated through on startup. It may be beneficial to remove features such as the screw holes and support structure to simplify the geometry, thereby lowering the vertices and increasing the startup of the program.

- Arm solving is very buggy, i.e., it will not always find the best solution and checking whether a solution is possible is a limited feature. This is a result of me attempting to use my own analytical solution using basic vector arithmetic. Although I am very happy with the time it takes to solve the arm (the current method has almost constant time complexity) the disadvantages of the limited positions which are able to be reached - and some very common mistakes in calculation - outweigh the benefits of this. A possible future implementation would be to use the *Jacobian Inverse Kinematics Algorithm* when solving for the sequencer (where accuracy is the priority), and then maintaining the current analytical solution for the simulator window (where speed is the priority).

- Variable naming conventions are not completely followed - there are quite a few variables which should really be private but due to a lack of prior knowledge and experience, I wasn't able to plan enough to know which variables would be better public and which private. My class design is very 'messy' in the sense that it is difficult to keep track of which variables are being passed in and out of classes.

- The documentation could be greatly improved by including a flow chart showing how each of the classes interact. This would have made it easy to I have not completed this due to lack of time.

- The slider interface for positioning the arm is not very intuitive. It feels clumsy to move the arm on one axis at a time, and the independent position-direction coordinates adds to this. A good feature - and one that was originally planned - would be to have the ability to drag the end-effector in the simulator window. This would make for a more intuitive user experience.

3 Known Bugs

- Attempting to initiate a connection with the Raspberry Pi when it doesn't exist will hang the program until it is connected, losing all user data. It would be good to add a timeout or make the connection initiate on a separate thread so the main application can continue running.

- When rechecking the 'Simulator' checkbox, the window reappears blank until it is resized. As of yet I have no been able to find the root cause of this. If I call the on_resize() function manually then the Simulation window content shows, but then the main window is blank until resized.

- The close button terminates the application with an error. This has only been found very recently and I have not had time to investigate it.