RMIT NAO ROBOCUP RECOMENDATIONS

RECOMMENDATION FOR DEVELOPERS

Ryan Couper, Ran Lu, Louis Stekhoven-Smith
s3575400, s3583185, s3530225 respectively

Table of Contents

Operations Recommendations

Problem No.1

Currently the development environment is run on the developer's local computer as a virtual environment. Development using simulated robots involves running upwards of eleven robots, a game server and the monitor to view the game state. This is quite computationally expensive and unless the developer's computer is powerful their virtual environment will run very slowly. This drastically reduces productivity in developing and testing new code.

Recommended Solution

RMIT allocates server resources to running the virtual environments allowing developers to carry out work on laptops with a remote connect to said server. Exact resources required for an optimal performance to cost ratio are unknown at this time as this was descoped due to time restraints. These exact requirements will require further research.

Problem No.2

Dependencies need to be downloaded during the rUNSWift build_setup script. There is currently no long-term place to host these files. As a temporary fix they have been hosted on a student's RMIT account, but this will need to change as this account will no longer exist in 6 month’s time.

Recommended Solution

Partition some hosting space on RMIT’s server to allow long term storage of these dependencies.

Problem No.3

Dependencies are currently hardcoded in the build_setup script. When you need to add a dependency to the project or mange existing projects you must dig through the script and add/change these hardcoded values.

Recommended Solution

C++ unfortunately doesn't have a great package management ecosystem. However recently developed tools like Buckaroo may prove valuable in streamlining dependency management for this project. Research into implementing such a package management tool is recommended.

Software Development

There are two major directions development can take. Either development can move towards simulated robots or towards real robots. rUNSWift code is currently optimised for real robots however RMIT currently has only two Nao robots. The ideal solution to this would be to fork the rUNSWift code and optimise locomotion for simulation. Thus, allowing behaviour development to be done rapidly in simulations and then moved over to the rUNSWift code optimised for real robots once RMIT has enough Nao's to form a team.

AI Behaviour Development

Learning Outcomes:
Develop skills in implementing AI
Develop skills in creating high level software architecture

Programing Language: Python

Description

The rUNSWift code base currently has no real behaviours implemented. If a team is interested in developing strategy algorithms that tell the robots when and where they should walk, when they should kick and how they should work together as a team then they should work on this aspect of the code.

Benefits

Working on this problem is you are able to test your code rapidly in a simulated environment removing the bottleneck of real robots which often over heat and can only be used for an hour or so at a time. Additionally, the development guide focus on this aspect of the robot.

Problems

The main problem here is the locomotion developed for the robot has been optimised for a real robot. As a simulation environment does not match the real world perfectly, when running the robots in simulation they often fall over and have trouble getting up.

Robot Kinetics Development

Learning Outcomes:
Develop skills in applying mathematical solutions to applications

Programming Language: C++

Description

Kinetics are fully implemented in the rUNSWift code however there is always research and work that can be done to improve this code. As the simulated robots and real robots are not the same a team who wishes to work on this code should pick one or the other to focus on.

Benefits

Working on this is mostly in developing the kinetics for simulated robots. As the code is currently optimised for real robots, simulated robots tend to move poorly. Improving simulated robot's movement will greatly assist the future development of behaviours.

Problems

There is no development guide for this. A highly successful will likely Team need both a good understanding of C++ and have a strong mathematical background.

Sensor Development

Learning Outcomes:
Develop skills in computer vision
Develop skills in data processing techniques

Programming Language: C++

Description

Sensor code is already implemented in the rUNSWift code base mostly focusing on computer vision to localise itself and to detect the ball. There are currently issues in object detection and avoidance meaning they are currently disabled. While the ball and line detection is decent, more efficient and accurate ball, line and goals detection can be developed.

Benefit

Highly accurate and efficient computer vision will be critical in winning a world cup.

Problem

Short term this will have little impact on moving the overall RMIT robot soccer team towards competition as decent implementations already exist. Also, no development guide currently available for this and further development will require good C++ knowledge.