System Design Project, Group 11, Report 1

This report summarises both our progress towards the end product of a Lego robot to play football and how we have created a Lego robot that can take a Penalty Kick.

Group Details and Organisation

Our group has selected the name *Skynet Football Club*. Skynet FC does not have an overall project manager, because we have not found an elected leader to be a useful role at this time. Instead, we have split the overall process into areas of responsibility; each of these areas has a named person, who has volunteered to ensure delivery. These people keep themselves informed of progress in their area, and therefore know what remains to be done. They can then provide direction and information to the rest of the team. The areas of responsibility are Vision, Strategy, Robot Design, Simulator, and Reports/Presentation.

It is intended that all group members will contribute to whichever areas they feel competent in. This has, to date, resulted in even, steady progress. Group members are encouraged to work in Appleton Tower Level 3 as much as possible, even if they are working on another course, so they can keep up-to-date with the project.

So far we have not adopted any particular programming paradigms, but we may take a decision to follow one if we feel it will benefit our work. When a group member has had a good idea it has been recognised and tested; this has allowed us to experiment with various solutions and so increased confidence in the choices we make. We are using a Subversion (SVN) repository, which allows us to write code collaboratively and efficiently.

Robot Design

Because only one group member has worked with Mindstorms robots before, it was decided to first build the basic robot as described in the NXT instruction booklet. We then examined controlling this robot remotely using Bluetooth. As we have developed further prototypes, we have been able to test their handling characteristics with simplicity using a Bluetooth remote control, on computer or smartphone.

We initially decided to gear-up the robot as we believe speed could be an important potential advantage. However, we found control over direction was drastically affected, and the prototypes would even stall. There have also been problems with rotating the robot using Bluetooth; due to lag the angle of rotation has varied widely and the gearing has amplified the problem. We now use direct-drive from the motors and have seen a great improvement in control.

For our method of ball propulsion we debated between using a traditional kicker or a 'spinning scorpion'-type design where the robot rotates rapidly and propels the ball with angular prongs. At the moment our robot uses a kicker as we believe it is more accurate.

A decision we took some time over was the rejection of usage of holonomic wheels in our design. The possibility of sideways motion was extremely tempting. However, we felt the amount of time it would take to implement holonomic movement would stall important development in other areas and pose a risk to the completion of our robot overall. This decision has appeared to solidify a 'simple and effective' design mantra among the group. Having a simpler robot allows us to test and refine more in Real World conditions, as we should have a working robot sooner.

Our current design uses two motors to drive two axles in the centre of the robot, each with a pair of wheels to improve stability. Our NXT programmable brick is placed in the middle of our robot; this gives a good balance between removability of the brick, and keeping the centre of gravity low. The robot is supported by two ball bearings; consequently it can turn on the spot. Our kicker is driven by a third motor and is powerful enough to shoot a ball directly into a goal from the opposite end of the pitch.

Programming

The group took the decision to program mainly using Python 2.6, for platform independence, for availability of useful libraries, and for the large amount of experience group members have with this language. We are going to use Python with OpenCV, which is also cross-platform, to facilitate Vision. This combination has a proven track record in past competitions.

Our program structure is modular, with individual components communicating through a central 'Knowledge Server' module, using TCP socket connections. Separate modules may be written in separate languages and run on separate physical machines. This allows us to distribute computing resources, isolate malfunctioning modules and write computationally intense modules in lower-level languages. Modules are given a type and an importance variable. For example, the visualizer is a logger-type module of secondary importance, so it receives messages for loggers but does not satisfy any central requirement for a connected logger module. Modules that crash are restarted.

Individual modules are started as separate processes that communicate, using `stdin` and `stdout`, through a proxy module that translates this into a socket connection. They attempt to notify the Knowledge Server before they exit/crash for logging purposes. The modules we have currently are:

- Knowledge Server- Routes communication between modules. Caches messages into lists for the separate module types, then sends to all appropriate modules that are currently online and/or come online later. Restarts modules that crash.
- Vision- Reports coordinates/direction of the robots and ball. In future, will be able to handle switching sides transparently and correct barrel/perspective distortion.
- Kicker- Handles moving the kicker.
- Simulator- Translates motor commands into vision output, using a physics engine.
- Logger- Saves all the information passing through the central knowledge module.
- Visualizer- Displays the internal state of the program using live data or, in future, a log file.
- Robot comms- Sends commands to the robot, will retrieve sensor data in future.

A practical application for our modular structure is that we may use our AI with either our simulator with the visualizer or our physical robot with vision. To try different AI we need only load a new module; this can be done while the system is running. Individual modules can be tested before developing the rest of the system. Therefore, testing is more comprehensive and efficient.

To begin with we were not developing on DICE as we had trouble setting up PyBluez, a system of wrappers facilitating development of Python Bluetooth applications. Now, though, this is resolved.

Milestone 1 Demonstration

We demonstrated that our robot can take a penalty kick, that it can rotate and shoot to aim for the

side of the goal, and score from a considerable distance. We continued with an examination of our simulator and how it communicates with other modules, then concluded with an example of how the vision system will be able to correct distortion in future. Things went smoothly. We cautiously chose to use a 20-second wait in each demonstration script to ensure the Bluetooth connection was reinitialised, so we may look into maintaining a permanent connection and loading more modules in advance. We felt our presentation comprehensively displayed our accomplishments.

Future Development

The following modules are required for our final program:

- Strategy- The main AI module. Will determine where the robot moves and when it kicks, using information from the sensor and vision modules.
- Movement- Calculates the commands to give to the robot through robot communications, to accomplish the determined strategy, using information from vision and sensors.

The purpose of this duality is to simplify the AI programming into the 'thinking' and 'doing' parts.

By Week 5 we aim to have reliable vision, then the robot will be able to navigate to a static ball, dribble or kick it into the goal, and react to collisions intelligently. By Week 7 we want to be able to intercept a moving ball and defend the goal appropriately. This will require the use of the simulator to predict the movement of the ball. For Week 9 we wish to have the ability to keep control of the ball while dribbling around a static opponent. To do this we will need to calculate smooth motion.

Our physical robot will require a kicker with larger sides, to be able to control the ball while turning. This would exceed permitted dimensions with the current chassis. Hence, we will need to refine our robot's design further. The kicker also needs to be implemented in the simulator.

Strategy Ideas

There has been eager discussion with regard to strategy, and the team are determined to produce a winning robot. We have reviewed online material, such as videos of robot football matches, to source inspiration for effective strategies. Strategies we have considered implementing include:

- Dribbling the ball against the side of the pitch, to make it more difficult for the opponent to steal
- Shooting at the earliest opportunity, either directly at the goal or against the wall, as dribbling may be less effective if the kicker is powerful enough
- Retreating to a defensive position to reduce shooting opportunities for the opponent, when it is considered unlikely we will reach the ball first, or if the opponent is already in control
- Utilising the 'back' of the robot instead of taking the time turning to bring the kicker to bear

As development continues, we will test these suggestions and judge their effectiveness.

Conclusion

Our project is demonstrably ahead of Milestone 1. As a group we decided early to meet as often as we could, this being especially important presently, while we don't have as many assignments in other courses. This has paid dividends in progress and we are eager to maintain our momentum.