A case study on the usability of NXT-G programming language

Khuong A. Nguyen

Computer Laboratory, Cambridge University
kan30@cam.ac.uk

Abstract. The release of the Lego Mindstorms kit has carried the flexibility and creativity of Lego into the world of robotics, whilst targeting a variety of children and adults audiences. To achieve this goal, a programming language called NXT-G was developed to provide everyone full control of the Lego Mindstorms kit, regardless of their programming experience. In this project, the programming language ambition is tested through practical experiments. In a controlled experiment, twelve participants carry out four tasks using the NXT-G software and a Lego robot. Their performances are then analysed to confirm the stated claim.

Keywords: *Lego Mindstorms, NXT-G, usability, programming language*

1 Introduction

For many decades, Lego has been widely known for their creative toy bricks, which are not only addictive to children but also appealing to adults. The release of Lego Mindstorms carried the flexibility of Lego into the world of robotics, while targeting the same above users. To achieve this goal, a programming language called NXT-G was developed to provide everyone full control of the Lego Mindstorms kit, regardless of their programming experience. In this project, the programming language ambition is tested. In particular, twelve participants were recruited based on their programming backgrounds to carry out four small tasks. The experiment results were then analysed to assess the usability of the NXT-G programming language as claimed above. It was very surprising that the results suggested even some 'experienced programmers' have problems using the NXT-G software to finish the tasks described in this project. The negative responses from the participants' questionnaires mostly affirm the findings of this project. The logical progression of this report is graphically depicted in figure 1.

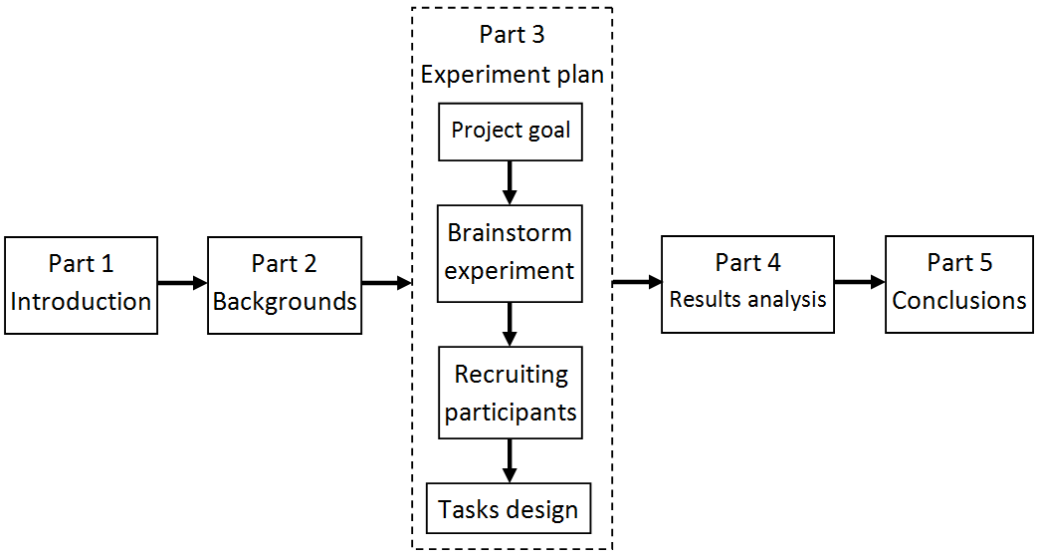

Fig. 1. Report progress

2 Lego Mindstorms robotics kit and NXT-G programming language

This section covers the background materials to be used in this report. Particularly, the Lego Mindstorms robotics kit and the NXT-G programming language are introduced, because the usability of this language is investigated using a Lego robot. This chapter also discusses several research papers which are relevant to the research topic.

2.1 Lego Mindstorms robotics kit

Lego Mindstorms NXT is a programmable robotics kit released by Lego in 2001 to replace the old Robotics Invention System kit. The NXT kit features the new Lego Technic stream, with the introduction of more sturdy beams in replacement of the traditional bricks (figure 2).

Fig. 2. A Lego Mindstorms NXT robot

In particular, what makes Lego Mindstorms excellent is the ability to interact with the outside world through the means of sensors. Compared to the old version, the new Lego Mindstorms NXT kit supports many more sensors such as the Acceleration sensor, the Gyroscope sensor, the Compass sensor, the EOPD sensor, etc.. These sensors deliver the inputs from the environment into the robot for processing.

2.2 NXT-G programming language

NXT-G is a visual programming language developed by Lego in conjunction with LabVIEW for the above Lego Mindstorms NXT kit. The language is targeted at children as well as adults with no programming background. Figure 3 shows the interface of the NXT-G 2.0 version.

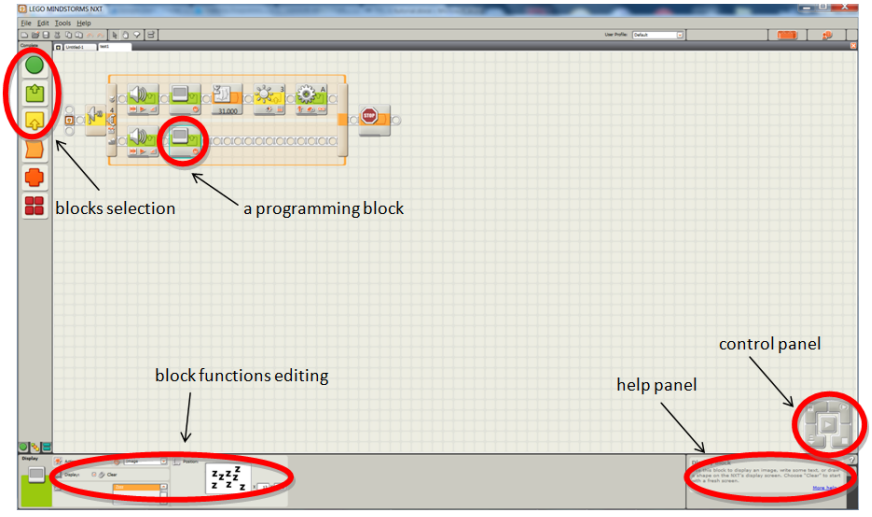

Fig. 3. NXT-G software interface

Since the latest 2.1 retail version, the language is now relatively stable, although there are still occasional crashes and bugs to be fixed. The whole concept of NXT-G is based on programming blocks. A block can be customised to perform a specific action. Many blocks can be chained one after another to perform a series of actions. Figure 4 demonstrates a simple behaviour to say 'hello' and turns the light on with only two programming blocks.

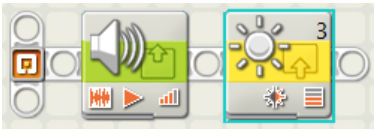

Fig. 4. Programming blocks

To provide a more flexible programming environment, two programming-related blocks are introduced in NXT-G. The 'Switch block' behaves exactly the same as the IF statement in other programming languages such as Java, C (figure 5). If the condition is true, all blocks in the upper beam are initiated. Otherwise, everything in the lower beam is initiated.

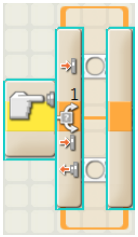

Fig. 5. The Switch block

Another important block is the 'Loop block', which behaves exactly as the DO-WHILE statement (figure 6). Any blocks being put inside the two 'square brackets' of this block are guaranteed to be performed once, and will be repeated until the condition is false.

Fig. 6. The Loop block

3 Literature review

The idea of using Lego blocks to approach programming has been implemented and tested with small children [3]. Each physical Lego block contains hidden digital implementation, and can be chained together to perform a sequence of behaviours. There are also 'sensor blocks' which receive inputs from the outside world, and transform the data into the blocks. Further, the use of Lego Mindstorms robots as a tool to convey programming ideas has been attempted [2]. Although the research mainly concentrated on the object-oriented aspect of the programming language, the use of robots provides a physical model to visually observe the results of a programming code.

4 Experiment plan

In this section, the experiment concept is formed, covering important stages in the process.

4.1 Project goal

With the release of Lego Mindstorms NXT, Lego has claimed that the NXT-G programming language requires little to no programming experience to program a Lego robot [1]. In this project, the above claim is tested through practical experiments, in which many participants are recruited to perform a series of programming tasks to control a Lego robot, as will be discussed in the next part. It is worth noting that although the Lego Mindstorms robot is involved in the experiments, it is just a tool to demonstrate the usability of the programming language. This project does not expect to investigate the usability of the robot itself.

4.2 Experiment brainstorm

To test the above claim, firstly, the overall NXT-G programming language is examined by the researcher, as an experienced 11 year programmer. This brief overview skimmed through every aspect of the programming language including the interface, the programming features, to give the researcher a feeling of the system, which will strongly orientate the design of the experimental tasks later. At this early stage, the researcher has already identified some confusion amongst the programming blocks - to be addressed later in this report, as well as some interesting features of the language such as the 'Switch behaviour', and the 'Loop behaviour' to be experimented on.

Since the NXT-G is believed to be suitable for everyone regardless of their programming background, it would be interesting to focus some of the experiments on the specific features of any typical programming language such as the IF statement, the LOOP statement, to see how a non-programmer handles them. Besides, because of the nature of a visual programming language, the layout of the components, the presentation of the programming features, the ease of navigation should also be parts of the experiment.

4.3 Recruiting participants

Once it was ascertained that the experiments would focus on the 'programming experience' aspect of the claim, two groups of participants were recruited, based on their programming backgrounds. The first group includes all programming experts. The expertise level in programming is quantified based on the number of programming languages a person knows, how often he uses it, and how long has he been using it. This information was observed beforehand, and a decision was taken if a person was to be recruited. The second group simply includes people who did not know how to program.

Due to time constraints, the original plan to recruit 20 participants could not be achieved in the given time-frame. Also, it was very hard to recruit non-programmers within the Computer Lab, University of Cambridge. Thus, the plan was modified to recruit just 12 participants, divided into two 6-person groups. The groups consist of five M.Phil students, four undergraduate students, two Ph.D students and one high school student. Everyone in the experienced programmer group knew at least two programming languages, and they had programmed very frequently in the past three years. The other group includes geography, biology and music students who had never used any programming language before. All the participants' ages varied from 18 to 24, and held at least a high school qualification. The variety of participant types plays an important role in the experiment results, since there are only 6 participants in each group. This selection also provides a moderate level of external validity, because the participants come from a variety of educational backgrounds, but they are still all students.

Also, the original plan was to recruit another two groups who are very familiar with Lego, and who never played with Lego before. This recruitment plan was abandoned, since it was virtually impossible to recruit any adult who had never seen Lego before. Although some of them claimed to have never played with Lego, they actually had seen it and knew how it worked, which destroyed the idea of the experiment. Small children are the most ideal candidate for this type of experiment.

4.4 Tasks design

Once the above recruiting process had been finalised, four tasks were carried out. For each task, a video protocol was used to capture the mouse and the keyboard activities on the screen.

Task 1: The purpose of the first task is to test the ease of recognising a programming block and the layout of the programming interface. First, a working NXT-G program is given to the participant, along with a description describing the exact behaviour of the code. Then, the participant is asked to modify that code to produce a different behaviour. The quantitative data measured in this task are the time taken (in seconds) to complete it, and the accuracy (correct or wrong) of the completed program. The qualitative data are the mouse and keyboard activities. The detailed behaviours of the code and the corresponding NXT-G program are demonstrated in figure 7.

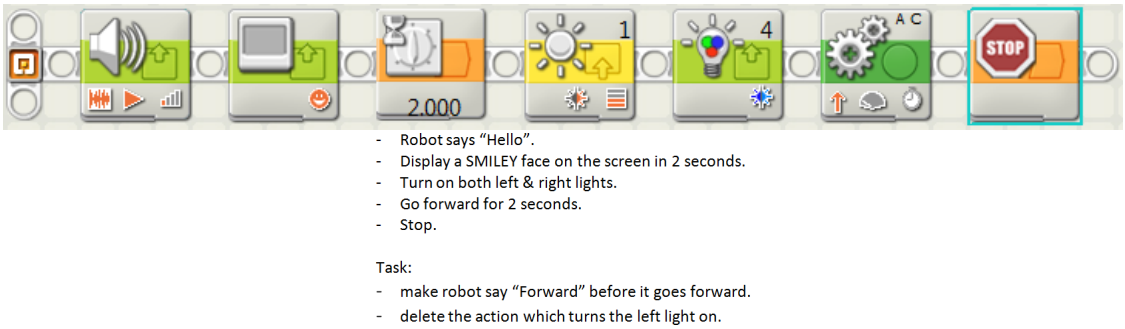

Fig. 7. Task 1 details

The part which might lead to confusion here is the selection of the 'Speaker block' and the 'Ultrasonic sensor block', since they look very similar at a quick glance (figure 8).

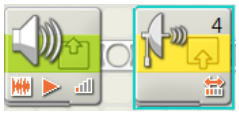

Fig. 8. Speaker block and Ultrasonic sensor block

The 'Light sensor' and 'Colour sensor' block can easily be wrongly recognised too (figure 9).

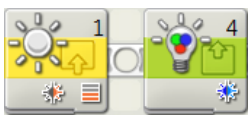

Fig. 9. Light sensor block and Colour sensor block

Task 2: The purpose of the second task is to experience the traditional IF statement, which is represented by the 'Switch block'. For this task, the same code which was completed in the first task was used. In case the participant did not perform the first task correctly, a solution was provided afterwards. The participant uses the 'Switch block' to make the robot produce a different behaviour depending on the environment condition (figure 10).

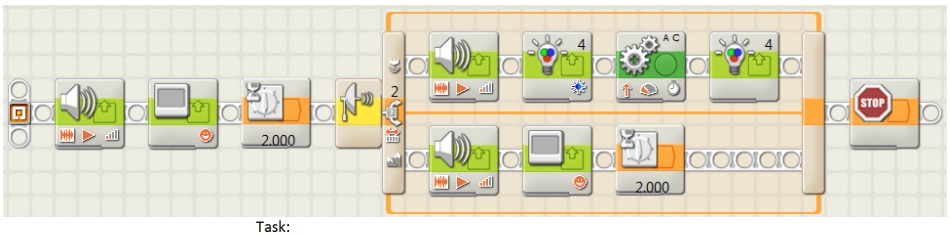

Fig. 10. Task 2 details

Task 3: The purpose of the third task is to prepare a set of duplicated actions, so that the fourth task can then use a 'Loop block' to test the effect of the FOR loop. The participant uses the previous program written in the second task, and creates many duplicated actions without using the 'Loop block' (figure 11). The solution for the second task was provided, in case the participant did not solve it correctly.

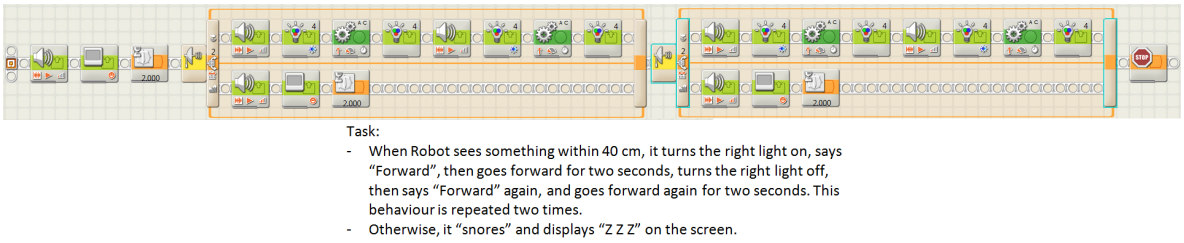

Fig. 11. Task 3 details

A small problem in this task was the navigation issue. When the code spans across the screen, due to the lack of the slider bar, the arrow buttons on the keyboard were used to shift the screen to the left or right, which was troublesome at times.

Task 4: The fourth task tests the effect of the FOR loop. It begins with the same program written in the second task, and repeats the same requirements as in the third task. However, the participant was requested to use the 'Loop block' to complete this task (figure 12).

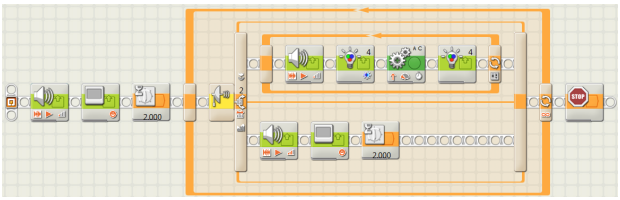

Fig. 12. Task 4 details

4.5 Experiment hypotheses

Based on the above four tasks, the following hypotheses were devised. These hypotheses are tested against the results obtained from the experiments.

- The experienced programmer group should complete each of the four tasks faster than the non-programmer group.
- Everyone should finish the fourth task faster (using the 'Loop block') than they did in the third task (no Loop block), because fewer keystrokes are required to complete the task.
- The more programming experience a person possesses, the less time is required to complete a task.

A pilot study with the above four tasks was performed by the experimenter to confirm the possible expected outcomes.

4.6 Experiment process

Once all four tasks are ready, 12 participants are contacted one by one to be interviewed individually. During the experimental session, which lasts no longer than 45 minutes to prevent the natural human fatigue effect, the participant first follows a short 10 minute online tutorial provided by Lego [7] to introduce the main concepts of the NXT-G programming language. This training session has very good external validity since the tutorial was designed for everyone, even those who had never used Lego before. Also, it was the necessary step to put all participants on the same level before conducting the experiments. After the training, the participant carries out the above four tasks. Through-out the experiments, all participants use the same laptop and robot. A webcam was mounted on the ceiling to record the user's activities, while a screen recorder was installed in the laptop to capture the mouse and keyboard movements. The use of these video protocols was agreed beforehand with the participants, and mentioned in the consent form. A quiet room was used to isolate all external noises. At the end of the session, the participant fills in a questionnaire describing his experience with the experiments thus far. All participants must also sign a consent form at the beginning of the session.

5 Results analysis

Of the above four tasks with 12 participants, the completion rate was 96%, with the exception of two participants in the fourth task, which will be explained below. The correctness of all finished tasks was also 100%.

5.1 Compare the performance of each task between the two groups

The first hypothesis claims that the experienced programmer group should complete each task faster than the other non-programmer counterpart. For the first task, which tests the layout and the interface, the Keystroke Level model shows an average of 143 seconds and 179 seconds for the experienced programmer group and the non-programmer group respectively. Although the experienced programmer group appeared to be faster (figure 13), the significant test suggests a t-value = 0.8 and p-value = 0.2, which shows small relative to the variance.

By analysing the qualitative data obtained from the video protocol, it was clear that both groups had trouble identifying the 'Speaker block' and the 'Sound sensor block', as well as between the 'Light sensor block' and the 'Colour sensor block' as suspected previously. The participants also had trouble finding the exact location of the blocks they were told to use. The P operator to point the mouse to the blocks took a considerable amount of 97 seconds and 112 seconds for two slowest persons in each group, because most participants adopted the trial-and-error approach to go through every block category from top to bottom until they found the intended block.

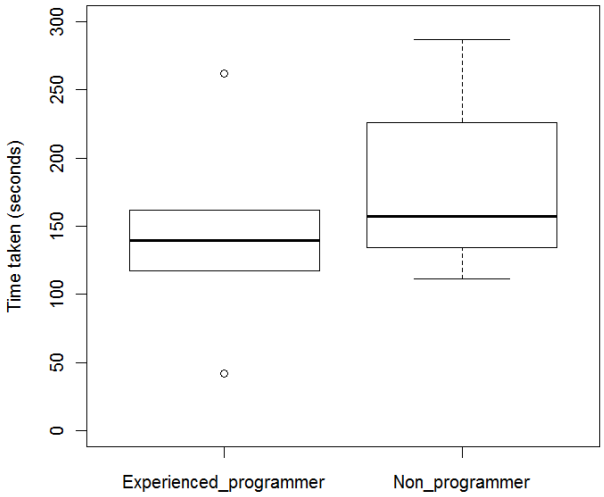

Fig. 13. Task 1 comparison between two groups

Another issue observed in this task is the layout of the 'blocks selection menu', which is not logically presented. The 'Common category' holds many duplicated blocks from other categories, with two new blocks - the 'Move block' and the 'Record/play block' which do not fit syntactically into this section. Further, there is no reason why the 'Speaker block' is considered to be more common than other sensor blocks to be put in this common category (figure 14). Neither was it explained in the documentation.

Fig. 14. Common blocks category

For the second task which experimented with the IF block, the experienced programmer group appeared to be much faster than the other group (figure 15). The mean difference between the two groups is as high as 91 seconds. The significant test also confirms a p-value $<<$ 0.05, with t-value $=$ 2.4, which re-confirms this mean effect.

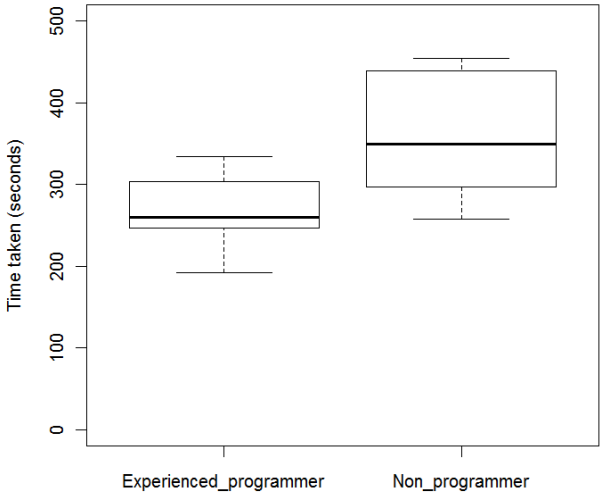

Fig. 15. Task 2 comparison between two groups

To explain this difference in performance, the activities from the video protocol of the non-programmer group were examined to confirm that 30% of the participants within this group were not familiar with the branch-idea and had problems separating the programming blocks between the two beams of the 'Switch block'.

For the third task, the experienced programmer group appeared to be faster again. Although the mean difference was just 23 seconds, the significant test with $t = 1.8$ and $p < 0.05$ suggests that the experienced programmer group did in fact perform quicker for this task (figure 16).

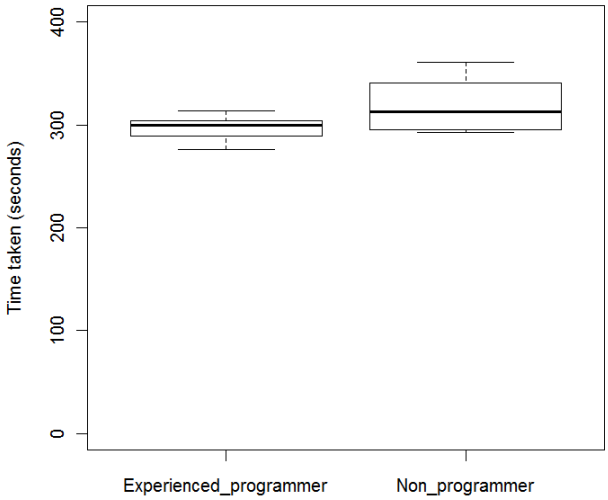

Fig. 16. Task 3 comparison between two groups

However, it was unexpected that both groups had trouble with the screen navigation when the long code spans across the screen. Mostly, the fact that the participants had to exchange between the mouse to drag-and-drop the blocks and the keyboard just to navigate around the screen, prevented the task being completed quickly.

For the final task, when the two groups experienced with the Loop block, it was clear that the experienced programmer group performed much better than the non-programmer group, with a significant effect of $t = 2.7$ and $p < 0.01$ (figure 17).

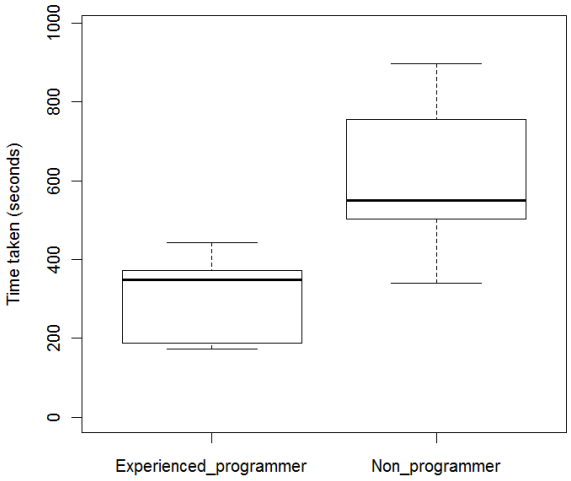

Fig. 17. Task 4 comparison between two groups

In summary, for all of the second, third and fourth tasks which involved the programming-related blocks, the experienced programmer group did perform better than the non-programmer counterpart. The first task, which only tested the interface and the layout of the programming language, received a mixed response from the two groups, with no one appearing to be clearly faster. The hypothesis, thus, appeared to be justified.

5.2 Compare the performance of the third and the fourth tasks

The second hypothesis predicts that every participant should finish the fourth task faster than the third task. This is an intuitive thought, since both tasks start with the same base program. The fourth version benefits from a shorter program with the use of loop, thus the participants can save more time from using fewer keystrokes.

Surprisingly, this was not entirely correct amongst the experienced programmer group, when 60% of the participants found it harder to implement the Loop block, which results in a longer task completion time (figure 18). The Keystroke Level model suggested an average of 296 seconds and 305 seconds for the third and fourth tasks respectively. However, the t-test reveals $t = 0.17$ and $p = 0.75$, which shows small relative to the variance for the experienced programmer group.

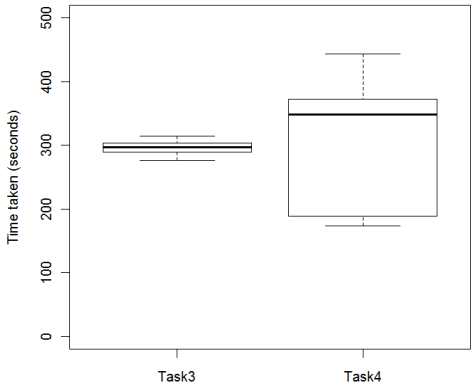

Fig. 18. Comparison between Task 3 and Task 4 of the experienced programmer group

The result was even more negative for the non-programmer group, when all participants needed much more time for the fourth task than they did for the third one (figure 19). The Keystroke Level model shows an average of 322 seconds for the third task and 609 seconds for the fourth task. The hypothesis is rejected with a strong mean effect of 288 seconds. The Mann-Whitney test reveals $w = 1.5$ and $p << 0.05$, which confirms the rejection of the hypothesis.

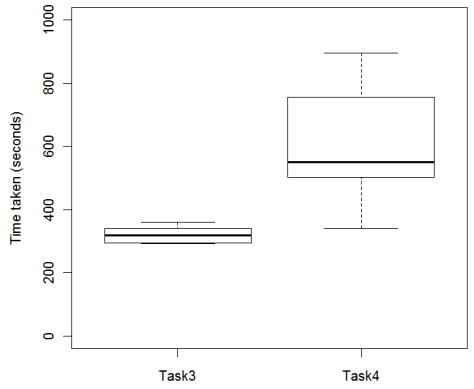

Fig. 19. Comparison between Task 3 and Task 4 of the non-programmer group

By analysing the mouse and the keyboard activities of the fourth task, it was observed that 75% of the participants had trouble with the options provided along with the Loop block (figure 20). The cognitive walkthrough method suggested that although all participants knew that they should use the Loop block, as they have been told to do so, 75% of the participants failed to recognise the correct option provided in the sub-menu. Only 25% of the participants managed to associate the correct option with the effect they expected, the rest exhaustively tried every option provided by the Loop block. This problem showed a poor representation of the block itself. However, the good thing was all participants quickly understood the feedback from the robot, and modified the code until the expected behaviour was achieved.

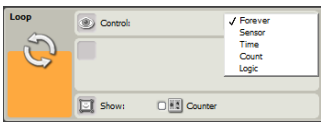

Fig. 20. Options for the Loop block

5.3 The relationship of the programming experience and the completion time

The third hypothesis states that the more programming experience a person possesses, the quicker he is able to finish all four tasks. The experience level was taken from the questionnaire. This is the number of years a person has programmed frequently. The result showed a squared Pearson correlation coefficient R^2 at 0.87, giving a p-value $<< 0.05$ which is significant at the 95% level (figure 21). This result provides a strong evidence for the above hypothesis.

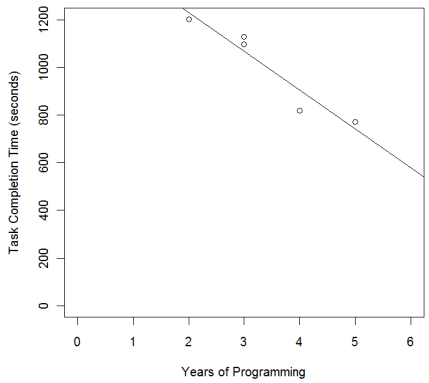

Fig. 21. Relationship of programming experience and completion time

5.4 Experiment issues

Despite the careful preparations, there were still several minor issues brought to the attention of the experimenter. First, two participants' results of the fourth task were excluded from the data analysis, because they could not finish the task. The first participant abandoned the task mid-way due to a fire alarm event. The second person had to stop in the middle of the task, because of the 20 minute constraint for each task, which had been agreed beforehand in the consent form, had been triggered. Fortunately, those two persons did not belong to the same group, so the variation was not influenced too much in the end. The second issue was that the NXT-G software crashed twice in the middle of the second task and the third task of two different sessions. The ongoing work was lost as a consequence, and the final completion time was compensated by resetting the stop-clock of these events.

5.5 Improvement suggestions

To make the NXT-G programming language more usable, some solutions for the problems identified in this report can be suggested, based on the above analysis as follows. The 'Common blocks' category should be allowed to be customised by the user, or it should automatically be updated to reflect the most frequently used blocks through-out many sessions. Some programming block icons should be re-designed to avoid confusion in recognition such as the 'Speaker block', the 'Ultrasonic sensor block', the 'Light sensor block' and the 'Colour sensor block'. The internal bugs should also be fixed so the software does not crash in the middle of the work, which was believed to be caused by poor memory management from the previous version. Finally, one of the most crucial add-ins required for the programming interface is the slide bars, which would immensely make it easier for the user to navigate across the screen in a long program.

6 Conclusions

6.1 Research contributions

In this project, the usability of the NXT-G programming language was experimented to test the Lego's claim that it can be used by everyone regardless of their programming backgrounds. Surprisingly, the experiment results suggested that it was not true. The experiments showed that even some experienced programmers had problems using the software. Some issues addressed in this report are the robustness of the software itself, the interface and the presentation of some programming blocks. Ultimately, the use of the traditional FOR loop in the visual programming language does not seem to be as efficient as in other text-based counterparts.

6.2 Future work

Due to time constraints, a limited number of 12 participants were recruited to perform the experiments. It would be more ideal to have as many as 50 participants from a variety of educational backgrounds, as well as different ages, and especially with small children. Also, the project can be expanded to survey the usability of the physical Lego robot to confirm whether it has any effect on the usage of the NXT-G programming language.

7 Acknowledgement

The author would like to thank Dr. Alan Blackwell for his insight comments, and the PPIG referees' useful suggestions on this work.

References

1. T. Griffin: *"The Art of Lego Mindstorms NXT-G Programming"* (2010).

2. P. B. Lawhead, C. G. Bland, D. J. Barnes, M. E. Duncan, M. Goldweber, R. G. Hollingsworth, M. Schep: *"A Road Map for Teaching Introductory Programming Using LEGO Mindstorms Robots"* (2002).

3. P. Wyeth, G. Wyeth: *"Electronic Blocks: Tangible Programming Elements for Preschoolers"* (2001).

4. D. Kieras: *"Using the keystroke-level model to estimate execution times"* (2001).

5. T. R. G. Green, M. Petre: *"Usability Analysis of Visual Programming Environments: a 'cognitive dimensions' framework."* (1996).

6. C. Wharton, J. Rieman, C. Lewis, P. Polson: *"The cognitive walkthrough method: A practitioner's guide"* (1994).

7. Lego Mindstorms NXT online tutorial: *"http://mindstorms.lego.com/en-us/Software/Default.aspx"*.