# Young People's Descriptions of Computational Rules in Role-Playing Games: An Empirical Study

Judith Good, Katy Howland, Keiron Nicholson
School of Informatics
University of Sussex
Falmer, United Kingdom BN1 9QH
{J.Good}{K.L.Howland}{K.Nicholson}@sussex.ac.uk

*Abstract*— **A study was carried out which examined the extent to which young people aged 11-12, with no prior instruction in programming, are able to write computational rules which govern play in a 3D computer role-playing game. Expressing these rules required the use of common computational structures such as conditionals, sets and loops. We analysed the rules written for their structure and style, and recorded the types of errors made. It was found that although young people were able to abstract away from the game play experience, very few of the rules were error-free. The most common errors were errors of omission (leaving elements out that should have been included) rather than errors of commission (including elements which should not be part of the rule). These findings have implications for the design of the Flip language, which aims to support young people as they begin to develop computational skills through game design.**

***Keywords-novice programmers; computational rules; empirical studies***

## I. INTRODUCTION

Learning to think 'computationally' has long been recognised as an important skill [1], and the recent computational thinking drive has refocused attention on this as a significant issue in modern computing [2]. Perhaps the most fundamental of the evolving set of computational thinking skills is the ability to define clear, specific and unambiguous instructions for carrying out a process: it is the very basis of computer programming, and its applicability to almost every domain of endeavour is easy to appreciate [3].

When looking at the issue of clear and unambiguous specification of rules within the context of programming, programming language syntax could act as a potential confound. In other words, if a rule is not specified correctly, does the problem arise from conceptual difficulties at the semantic level, or is the problem a result of difficulties with syntax alone? One way of finessing this issue is to look at rule specification using natural language and in so doing take syntax errors out of the equation entirely. By removing the possibility of syntax errors we can better identify and examine any underlying semantic errors, which are often of greater significance to novice programmers as they relate to more fundamental logical or conceptual misunderstandings. Furthermore, while syntactic issues relate primarily to the idiosyncrasies of a particular programming language, semantic issues hold greater importance when considering computational thinking in the wider world.

Previous studies, most notably by Pane [4], have examined how non-programmers 'naturally' express programming concepts in terms of the language and vocabulary used. Although this research is a useful first step in designing programming languages or support tools for novice programmers, it doesn't consider the errors that arise when users try to specify clear and unambiguous rules for object behaviour. A consideration of errors seems crucial if we are to design specific support for users in the context of learning to program. We have therefore expanded the focus of previous work to include a classification of errors which occur during rule specification.

We conducted a study to look at how young people, aged 11-12, use natural language to express rules in writing, and the kinds of linguistic structures that they use. Our research questions were as follows:

- To what extent can young people use natural language to write rules in a clear, complete and unambiguous way?
- What types of errors occur in their rules?
- What types of linguistic structures and which particular words do they use to express computational concepts?
- Is there a relationship between the particular linguistic structures or words used and the frequency and type of errors that occur?

The following section describes the background to this study, including the broader context of our research, and related work. We then go on to describe the study in detail in Sections III and IV, consider results in Section V, with a general discussion in Section VI, and finally, conclusions in Section VII.

## II. BACKGROUND

Over the past 10 years, our research has focused on empowering young people (aged 10-15) to work with game-creation software to develop their own commercial quality video games as a creative and technical exercise [5, 6]. We have found game creation to be highly motivating for young users, and as such it provides an ideal context for introducing users to the often difficult topic of programming, and the related computational thinking skill of rule specification.

We work with a commercial computer game called Neverwinter Nights 2 (NWN2), published by Atari in 2006.

NWN2 is a computer role-playing game (RPG), in which players explore a large fantasy world and take part in a dramatic interactive story, with their own choices having an effect on how the plot progresses. The game is part of the Dungeons & Dragons franchise, a series of 'medieval fantasy' RPGs.

Included in the NWN2 software package is the Electron Toolset: a professional game-development application which was used by the developers, Obsidian Entertainment, to build the game itself. This means that in addition to playing through the included game, users can go on to build their own video games using the developers' own tools and art resources to design new 3D worlds and adventures. These user-created games feature new plots, dialogues, locations and challenges, but retain the same high-quality graphics and complex game mechanics one would expect from a commercial product. Although there are many complexities to working with this toolset, around five minutes of instruction is enough to empower users to begin independently working on their own 3D landscapes, complete with rivers, trees, buildings, characters, and other points of interest.

Unlike most game-based programming environments such as AgentSheets [7], Scratch [8], Alice [9], and those designed more specifically to support storytelling [10], it is not immediately necessary to start the game creation task by programming. Instead, users are able to explore the possibilities of the toolset, and use it to create something that appeals to their own sensibilities, relying in the first instance on inbuilt game behaviours. However, when they ultimately do need to engage with programming in order to specify custom behaviour for the objects in their game, their investment in their game world and the narrative they are constructing often drives them to try to achieve programming outcomes. A quest to steal the dragon's treasure is not much fun if the dragon has not been told to defend it! The behaviour of the dragon, or any other object in the game, forms a key part of the narrative the user is constructing, and so the game will be incomplete until the user can achieve at least some level of success with programming.

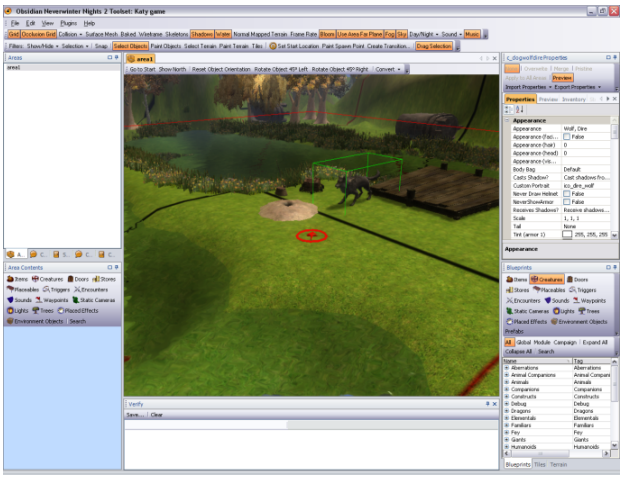

Figure 1. The Electron Toolset. Users create 3D areas in the centre panel, using the tools and resources in the outer panels

The programming language used in conjunction with the Electron toolset is a scripting language called NWScript, similar to C in its syntax and level of complexity. Young users are almost invariably intimidated and frustrated by this element of the game creation tool, and are unable to proceed without substantial support from experts. It is in this context that we seek to encourage and scaffold their attempts to learn the basics of computer programming by developing a dedicated language, Flip, for scripting behaviours in computer RPGs.

Through the numerous game creation workshops we have been involved with, we have anecdotal evidence that young people can specify computational rules whilst describing narrative gameplay elements in the context of an informal conversation. However, they often require prompting from workshop staff before they fully state the conditions and actions involved. We wanted to gather firmer evidence on whether young people can specify these rules in natural language in a clear, complete and unambiguous way. Additionally, we were interested in examining any errors present in the rule descriptions since different types of errors could suggest differences in the types of support which Flip could most usefully provide.

In designing a language which can provide support for novice programmers, we have examined the issues that such users have in trying to use possibly unfamiliar computational concepts to express their desired game behaviours. This information will help us to provide targeted support to allow users to avoid semantic errors and develop their understanding of computational and logical structures. More broadly, given the narrative based nature of the games created in this environment, it made sense to consider whether the Flip language should be based on natural language, similar to Inform 7 [11], which also operates in a narrative creation context; that of interactive fiction.

The study described here forms part of a wider process of Learner Centred Design (LCD) for the Flip language. We consider it important to design educational tools with extensive input from the target users, and followed the CARSS framework to guide our LCD activities [12]. CARSS helps researchers to conduct participatory design, i.e. design which involves end users in the design process from the outset, by considering the *context* in which the software is to be used, the *activities* which can be carried out during the design phase, the *roles* which the design team members must take on, the necessary *skills* which the team members must possess, and the *stakeholders* involved in the project. In addition to the study described here, other work has included the design and user testing of a low-fidelity prototype version of the Flip language, and asking young people to design their own visual representations for different rule elements.

## III. The Study

The study reported in this section was designed to examine the research questions outlined at the end of Section I. The study was part of a broader involvement with a high school over a period of approximately 10 lessons, in which we taught young people how to make their own computer

games. As such, appropriate ethical approval was sought from the University, and all researchers undertook Criminal Records Bureau (CRB) checks before commencing the research.

### A. Participants

The study took place in a high school in the south of England. The participants were 64 young people in year 7 (35 female and 29 male), all of whom were between 11 and 12 years old. The participants were spread across 3 classes, and were taught the same material by the same ICT (Information and Communication Technologies) teacher. The school itself consistently performed below the national average in standardised tests, but the school leaders were committed to improving results.

### B. Materials

The materials used in the study were an NWN2 game and a worksheet designed to investigate how young people expressed computational concepts when describing rules in games. The game was created by the third author using the NWN2 Electron Toolset. The game contained seven scripted encounters, each of which embodied particular types of computational structure, and which increased in complexity.

The test items were administered in the form of a worksheet containing a question referring to each encounter, and asking the student to write a rule which could have caused the behaviour they observed. In total, the worksheet comprised eight questions, the first seven of which referred to each of the seven encounters in turn. The final question asked pupils to propose their own hypothetical scenario which might occur in the game, and to write the rule for that scenario. As very few pupils attempted to answer this question, and those who did appeared to have misunderstood the question, it was excluded from the analysis. The first two questions were multiple choice, and required pupils to choose the statement which best described the behaviour they had observed. They served as a warm-up to the main exercise, and as answering them did not involve rule writing, they are not described further. The subsequent five questions asked pupils to write the rule in their own words. Table 1 shows, for each of the encounters which required writing a rule, a prototypical rule describing the encounter, and an indication of the underlying computational rule-structure for each component.

### C. Method

The game and worksheet were introduced during a normal class session (lasting 50 minutes), which took place in a computer lab in the school. All pupils had access to the demo game on their computers, and a copy of the worksheet was handed out to each pupil. The pupils were asked to play through the demo game, and in particular, to "Find the village at the heart of the forest. Follow the path to explore the game world and reach the village". For each game encounter, the pupils were asked to "write a rule" which would produce the behaviour which they had just experienced in the game. The game was available to the pupils while they were completing the worksheet, and pupils had the option of replaying the encounters and exploring different options if they desired. The worksheet was administered to the three classes of year 7 pupils, with the method kept constant across all classes.

## IV. Analysis

### A. Overview of Coding Schemes

In order to analyse the answers collected through the worksheets, two coding schemes were devised: one focusing on the errors contained in the rule descriptions (based on the error analysis used in [13]), and another focusing on the linguistic structure of the rules, extending work carried out by Pane [4]. In order to ensure reliability, the schemes were applied by all three authors over a subset of the data, and coding instructions were refined so as to reduce inconsistencies in their application.

For the error scheme, we created a set of 'model answers' which were used to assess answer correctness (see Table 1).

TABLE 1. RULE DESCRIPTIONS AND UNDERLYING COMPUTATIONAL STRUCTURE FOR EACH QUESTION

| Q | Model Answer | Code |
|---|---|---|
| 3 | *When you click on the old man* | **T1** |
| | *he speaks to you,* | **O1.1** |
| | *and then conjures a monster* | **O1.2** |
| | *to fight you* | **O1.3** |
| | *(which will also kill him).* | **O1.4 *** |
| 4 | *If you see a skeleton or lizard man,* | **C1** |
| | *fire an arrow.* | **O1** |
| | *If you see a bear, wolf or deer,* | **C2 *** |
| | *let them pass unharmed* | **O2 *** |
| 5 | *If you answer 'true'* | **C1** |
| | *the knight gives you a book,* | **O1** |
| | *if you answer 'false'* | **C2** |
| | *he gives you a sword* | **O2** |
| 6 | *While you stand on the blackened spot,* | **C1/ LSC** |
| | *the cats will all freeze in place* | **O1.1/ LB1** |
| | *and start meowing.* | **O1.2/ LB2** |
| | *When you move off it,* | **C2/ LEC *** |
| | *they'll start moving around again.* | **O2/OLB *** |
| 7 | *If you see an unopened treasure chest* | **C1** |
| | *open it.* | **O1** |
| | *When you open a chest,* | **T2** |
| | *pick up any gems it contains,* | **O2.1** |
| | *then carry on looking for unopened chests.* | **O2.2** |
| | *When you have four gems* | **T3/LEC** |
| | *return home.* | **O3/ OLB** |
| | *If you see a guard at any point* | **C4** |
| | *stop what you are doing* | **O4.1** |
| | *and run away from him.* | **O4.2** |

**T: Trigger, O: Outcome, C: Condition, LSC: Loop Start Condition, LEC: Loop End Condition, LB: Loop Behaviour, OLB: Out of Loop Behaviour,*: Non essential component.**

Each answer was broken down into the trigger/condition and the outcome of the rule. We did not insist on these elements being specified in any particular way or using any particular language in order to be correct. Similarly, we did not insist on any particular order, provided that differences in ordering did not change the meaning of the rule. The requirement was simply that conditions and outcomes were described completely and unambiguously.

Broadly speaking, we were interested in distinguishing between *errors of omission* and *errors of commission*. In the first case, rule elements which should be present are left out whereas in the latter case, elements which should not appear in the rule, because they are erroneous, are present. The full error scheme is shown in Table 2. Codes were applied to the trigger/condition and action parts of each rule. Note that more than one code may be applicable (although this was infrequent), for example, a trigger may be both "partially missing" and "partially erroneous". The descriptive analysis scheme looked at a number of characteristics of the answers: overall rule structure (if any), perspective rule is written from, tense rule is written in, keywords used, use of sets and loops and use of complex conditionals. These categories were adapted from those used by Pane [4] for our purposes (see Table 3 for the complete list of categories, and codes within each category).

### B. Inter-Rater Reliability

All of the study data was first coded by the second author. Two passes through the data were made, using the error scheme and the descriptive scheme respectively. Following this, a random sample of 20% of responses across all questions was generated for second coding. Inter-rater reliability was determined using Cohen's Kappa. For the error analysis, the Kappa value was .75, while for the descriptive analysis, the Kappa values for each category were all > .76 (indicating excellent inter-rater agreement according to Fleiss [14]).

## V. Results

Overall, worksheet responses were low, with responses less frequent for the latter questions. This is not entirely unexpected, given that the worksheets were administered in a typical high school classroom, rather than in a controlled experimental setting. It is likely that some pupils found the questions difficult, others may have run out of time, and yet others were not motivated by the worksheet format. The teacher noted difficulty with, and an aversion to, writing on the part of some pupils, particularly the boys, which may also have had an effect.

TABLE 2. ERROR ANALYSIS CODES

| | | | |
|---|---|---|---|
| **Error Analysis** | | ***C*** | All essential elements are expressed correctly and unambiguously |
| | **Omission** | ***M*** | All elements of this section of the rule are missing |
| | | ***PM*** | Some elements of this section of the rule are missing |
| | | ***I*** | This section of the rule has been left unfinished |
| | **Commission** | ***E*** | Rule section contains only erroneous information |
| | | ***PE*** | Rule section contains some erroneous information |
| | | ***V*** | Rule section contains information which is ambiguous or vague |

TABLE 3. DESCRIPTIVE ANALYSIS CODES

| | | |
|---|---|---|
| **Rule-structure** | ***NONE*** | No rule-structure |
| | ***E*** | Event-based |
| | ***D*** | Declarative statements |
| | ***I*** | Imperative statements |
| **Perspective** | ***1st*** | 1st (Refers to player as 'I') |
| | ***2nd*** | 2nd (Refers to player as 'you') |
| | ***3rd*** | 3rd (Refers to player as 'player') |
| | ***U*** | Undetermined |
| | ***N/A*** | No player interaction involved |
| **Tense** | ***CON*** | Conditional (Type 1) |
| | ***PRES*** | Present |
| | ***PAST*** | Past |
| | ***FUT*** | Future |
| | ***U*** | Undetermined |
| **Keywords** | ***AND-BC*** | AND-Boolean conjunction |
| | ***AND-S*** | AND-Sequencing |
| | ***AND-BD*** | AND-Boolean disjunction |
| | ***AND-C*** | AND-indicating consequence |
| | ***AND-O*** | AND -other |
| | ***OR-BD*** | OR-Boolean disjunction |
| | ***THEN-S*** | THEN-Sequencing |
| | ***THEN-C*** | THEN-indicating consequence |
| | ***EXCEPT-E*** | EXCEPT-indicating exception |
| | ***BUT-E*** | BUT-indicating exception |
| | ***BUT-S*** | BUT-sequencing |
| | ***BUT-BC*** | BUT-Boolean conjunction |
| | ***IF-C*** | IF-specifying condition |
| | ***WHEN-C*** | WHEN-specifying condition |
| | ***WHEN-T*** | WHEN-specifying temporal trigger |
| | ***ELSE-O*** | ELSE-meaning 'otherwise' |
| | ***TO-C*** | TO-indicating conditional |
| | ***FOR-C*** | FOR-indicating conditional |
| | ***AS-T*** | AS-specifying temporal trigger |
| | ***NOT-BN*** | NOT -Boolean negation |
| | ***ONLY-E*** | ONLY-meaning 'exclusively' |
| | ***=-C*** | =-indicating conditional |
| | ***--C*** | --indicating conditional |
| | ***IN-C*** | IN-indicating conditional |
| | ***REP-L*** | REPEAT-indicating loop |
| | ***SO-C*** | SO-indicating consequence |
| | ***NONE*** | None (if no keywords used) |
| **Sets or loops** | ***S*** | Set or subset specification used |
| | ***L*** | Loops used |
| | ***NONE*** | None |
| | ***N/A*** | Sets and loops inappropriate |
| **Complex Condition** | ***MER*** | Set of mutually exclusive rules |
| | ***GCE*** | General condition modified with exceptions |
| | ***OC*** | Listing only one condition |
| | ***NONE*** | None |
| | ***N/A*** | Complex conditionals inappropriate |


Figure 2. Errors in triggers and outcomes

*A. Error analysis*

The overall rate of correctness was quite low, with only 24 out of 113 attempted answers being fully correct (21%). In addition, there were 39 answers (35%) where one element (either trigger/condition or outcome) was correct. As shown in Fig. 2, in 41% of answers the outcome was correct and in 35% of answers the trigger/condition was correct. Errors of omission were by far the most common category (74%). The most common error made with triggers/conditions was to miss this element out entirely (27%), with the second most common error being for the trigger/conditions to be partially missing (12%). For outcomes, the most common error was a partially missing outcome (27%), followed by a completely missing outcome (19%). Vagueness was a problem in specifying triggers (16%), but not a common problem for outcomes (3%). Incomplete answers, which seemed to have been abandoned by the student partway through, accounted for a small number of errors in both triggers and outcomes (3% of all answers). Errors of commission, where participants included incorrect information in their answers, accounted for a relatively small number of errors in both triggers and outcomes (7%).

*B. Descriptive analysis*

By far the most common rule structure used was event-based (74%), which was expected given the highly event-based nature of the encounters and the game environment. Declarative structures were the second most frequent (16%). Imperative approaches were observed very rarely (4%).

The perspective used most often in the answers was 2nd person (60%). 1st person perspective was used in a small number of cases (10%). 3rd person perspective and combinations of 1st with 2nd and 2nd with 3rd were used occasionally (7%, 3% and 1% of all answers respectively). For the remaining 19% of answers, no words which indicated perspective were used.

The most common tense used in the rules was present (72%), with past, future and conditional tenses being used only rarely (10%, 4% and 4% of all answers respectively). In a small number of cases a combination of different tenses were used in a single answer (8%). In very rare cases, (3%) it wasn't possible to determine the tense.

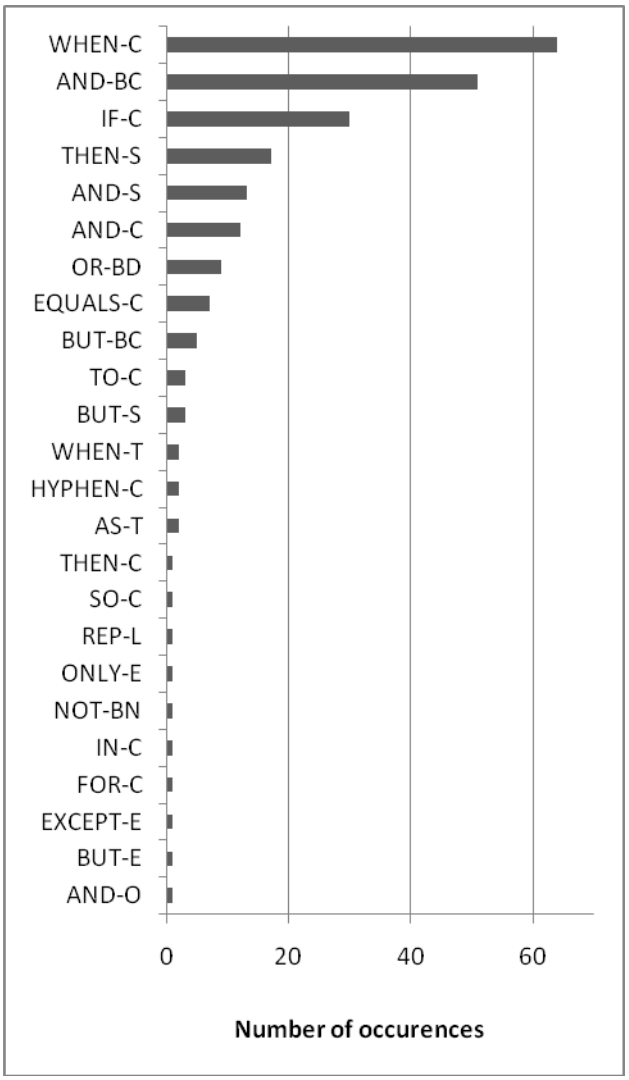


Figure 3. Keyword frequency across all answers

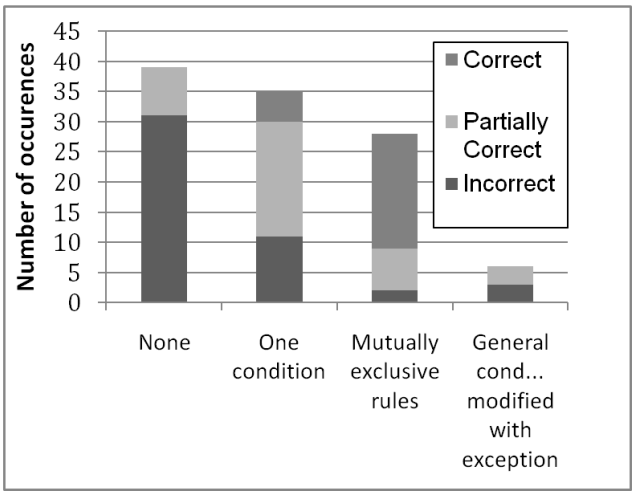


Figure 4. Complex conditionals and correctness measure

A large variety of computational keywords were used in answers. For the purposes of this analysis all words used as a keyword were counted. Different uses of the same keyword were categorized separately, as shown in Table 3. 'When' (conditional), 'And' (Boolean conjunction) 'If' (conditional) and 'Then' (sequencing) were the most commonly used keywords. See Fig. 3 for a full chart of all keyword usage.

Sets were used in 65% of answers where the use of either sets or loops was appropriate. Participants used sets rather than loops in all cases except in one answer.

Focusing on answers where the use of complex conditionals was appropriate, in a relatively high number of cases involving complex conditionals (36%), participants did not attempt to describe them at all. Where they did attempt to describe complex conditionals, the most common approach was to specify only one condition, with the other condition indicated implicitly (32%). In a number of cases participants used sets of mutually exclusive rules to describe conditionals (26%). In a small number of cases (6%) participants used a general condition subsequently modified with exceptions.

### C. *Relationship between errors and descriptive analysis*

In addition to examining errors and the language used within the rules, we were interested in whether there was any correlation between particular types of language, and correctness. In other words, are any particular patterns of word usage associated with correct answers? In order to perform this analysis, answers were firstly coded as either incorrect, partially correct (with one of the two rule elements correct) or fully correct (with both rule elements correct).

Given the categorical nature of most of our data, we used Pearson's contingency coefficient (C) measure, and (when analysing keyword frequency) Goodman and Kruskal's gamma (G) measure. Overall, there were few significant associations; those that were found are described below.

There was a significant moderate association found between rule-structure used and the level of errors in answers ($p<0.01$, $C=0.396$). Frequency of keywords used had a highly significant moderate positive association with correctness of answer ($p<0.001$, $G=0.473$). There was a highly significant moderate association between use of sets/loops and correctness of answer ($p<0.001$, $C=0.451$). There was a highly significant moderate association between complex conditional mode and correctness of answer ($p<0.001$, $C=0.616$). Notably, correctness level was much higher when complex conditionals were expressed as mutually exclusive rules as illustrated in Fig. 4.

## VI. DISCUSSION

Overall, we found that young people found it very difficult to write rules which could be considered to be complete, correct and unambiguous: many questions were simply left blank, and of those answered, only 21% could be considered to be fully correct. This is surprising given that the use of natural language excludes the possibility of syntax errors.

Secondly, errors of omission were much more frequent than errors of commission (74% were errors of omission). This suggests that there were few fundamental misunderstandings about the nature of the game encounter itself, but there were difficulties in terms of fully specifying the rule. In other words, rules were likely to be *accurate*, but not *complete*.

Taken together, this suggests that when allowing young people to express rules in natural language, they are much more likely to get it wrong than right. Although natural language eliminates syntax errors, it also allows for ambiguity: a single sentence can have multiple interpretations and meanings, unlike programming language statements. Therefore, one might have hypothesised that the ambiguity allowed by natural language could actually have worked against pupils. However, given that the most common errors were caused by omission of an important element, rather than by including incorrect information or by being vague or ambiguous, the results suggest that this was not in fact the case.

The most common tense and perspective used were 2nd and present, which suggests that despite their more generic difficulties, our participants were able to abstract away from their specific game experiences to construct a general rule which can apply in all cases (e.g. "When you talk to the knight, if you answer 'true', he gives you a sword, and if you answer "false", he gives you an old book"). In contrast, rules written in the 1st person and past tense suggested an inability to perform such abstraction, with the result that the rule would only cover what they had personally encountered, thus missing other possible outcomes (e.g. "When I talked to the knight, I answered 'true', and he gave me a sword"). This was observed only infrequently (<1% of cases), which is encouraging.

Finally, and not surprisingly, the significant associations between the correctness of the rule and the number of keywords used, the use of sets/loops, and the use of complex conditionals suggest that young people's ability to express computational rules correctly depends on their ability to firstly understand the computation, and then map their understanding to natural language expressions that successfully convey those concepts. In other words, those

pupils who do well are able to use common natural language expressions in a more principled and computational manner in order to specify rules in a complete, clear and unambiguous way. Two important points arise from this finding: firstly, the observation that the more complete solutions are also more correct. This may seem trivial, but it suggests that those learners who are writing more complete solutions are not simply including detail, but the *right types* of computational detail. One could argue that the inclusion of more detail runs the risk of increasing the chances of errors of commission, but we did not find this to be the case. Secondly, the aim of our study was to look at how to design a language which supports novices as they make their first steps in programming in the context of game design. The fact that so few are able to correctly specify rules means it is all the more important to investigating the structure of the rules of those who *are* able to do so, looking closely at issues such as ordering of statements, particular keywords used, etc. In so doing, this might allow us to better understand how we can support less able students as they learn to build and reason about computational events in their games.

## VII. Conclusions

The results obtained in this study, as well as having broader significance, have a number of implications for the design of the Flip language. Before considering these, we discuss the limitations of the study. Because the study took place in a real classroom, the reward structure in place was that of a typical school, meaning that not all students were motivated to produce the best possible answer. It may well be that the worksheet itself was not very motivating, particularly, as discussed above, for those who don't like writing. We are currently using an online survey tool for further data collection in order to investigate whether it is possible to mitigate some of these effects. Additionally, because the young people played through a game in which encounters happen in linear fashion, it is not possible to randomise the question order: this again may have influenced results, but it is difficult to see how this can be addressed, particularly when concepts of increasing difficulty are being introduced via the game.

Given the frequency of occurrence, it seems clear that the most support should be provided for errors of omission. The specification of rules in natural language did not lead to ambiguity in answers as much as it did to incomplete answers. This may have arisen because a rule written in natural language gives very little indication of its inherent structure, in other words, it is hard to look at a sentence and see, at a glance, whether part of the rule is missing. Although this could be addressed through secondary notation to some extent, we feel that the issue could be more usefully tackled by the use of a visual language interface which represents rule structure iconically, and shows clearly where information is needed. This iconic representation, which essentially provides a rule template, should allow pupils to focus, in the first instance, on providing the content for these rule structures, rather than the structures themselves.

To help pupils correct errors of commission, we will provide a read-only natural language representation of the rule. Pupils can use this to check that what they have specified in the visual language is what they intended to specify. Although a natural language representation may not be the most suitable representation for rule specification, we hypothesise that it may be better able to represent the semantics of the rules clearly, and thus be more effective for understanding and debugging. The association between correctness and specifying complex conditionals with mutually exclusive rules suggests that this would be a sensible mode of expression to employ in Flip.

The results from this study suggest that when learning to specify computational rules, although learning syntax remains a problem for novices, computational semantics is also an area of difficulty that needs to be addressed, perhaps to a greater degree than has previously been recognised. This has implications for the way in which we choose to support young people in their attempts to understand the nature and structure of computational rules.

## References


[1] S. Papert, *Mindstorms: Children, computers, and powerful ideas*: New York: Basic Books, 1980.

[2] J. Wing, "Viewpoint-Computational Thinking," *Communications of the ACM-Association for Computing Machinery-CACM,* vol. 49, no. 3, pp. 33-5, 2006.

[3] K. Howland, J. Good, and K. Nicholson, "Language-based support for computational thinking," *2009 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 09)*. pp. 147-50.

[4] J. Pane, C. Ratanamahatana, and B. Myers, "Studying the Language and Structure in Non-Programmers Solutions to Programming Problems," *International Journal of Human-Computer Studies,* vol. 54, no. 2, pp. 237-64, 2001.

[5] J. Good, and J. Robertson, "Learning and motivational affordances in narrative-based game authoring," *Narrative and Interactive Learning Environments (NILE 06)*. pp. 37-50, 2006.

[6] K. Howland, J. Good, and B. du Boulay, "A Game Creation Tool which Supports the Development of Writing Skills: Interface Design Considerations," *Narrative and Interactive Learning Environments (NILE 08)*. pp. 23-9, 2008.

[7] A. Repenning, A. Ioannidou, and J. Zola, "AgentSheets: End-User Programmable Simulations," *Journal of Artificial Societies and Social Simulation,* vol. 3, no. 3, 2000.

[8] J. Maloney, Y. Kafai, M. Resnick *et al.*, "Programming by choice: urban youth learning programming with Scratch," in 39th SIGCSE Technical Symposium on

Computer Science Education, Portland, Oregon, 2008, pp. 367-71.

[9] C. Kelleher, D. Cosgrove, D. Culyba *et al.*, "ALICE2: programming without syntax errors," *User Interface Software and Technology*, 2002.

[10] C. Kelleher, and R. Pausch, "Using storytelling to motivate programming," *Communications of the ACM,* vol. 50, no. 7, pp. 64, 2007.

[11] G. Nelson. "Natural Language, Semantics Analysis and Interactive Fiction.," http://www.informfiction.org/I7Downloads/Documents/WhitePaper.pdf. Accessed March 12, 2010.

[12] K. Howland, J. Good, and J. Robertson, "Script Cards: A Visual Programming Language for Games Authoring by Young People," *2006 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 06).* pp. 181-6.

[13] J. Good, and J. Oberlander, "Verbal effects of visual programs: Information type, structure and error in program summaries," *Document Design,* vol. 3, no. 2, pp. 120-34, 2002.

[14] J. Fleiss, "The measurement of interrater agreement," *Statistical methods for rates and proportions*, pp. 212-36, New York: John Wiley and Sons, 1981.