EXECUTIVE SUMMARY

Learning computer programming is never a simple task for novices. Many undergraduate students experience several challenges in learning programming especially their first programming language. There are chances of learning and progressing in programming but there are also high chances of not learning and failing it. Several techniques are employed by many lecturers today in efforts to ensure that atleast the basic knowledge of programming gets across to the students. C-Jump Computer Programming Board is a new game in the game industry that teaches basic computer programming statements to its players. It educates the basic commands of a programming language, such as the concept of variables, “if”, “else”, “switch”, “while” and “continue”. Players will be able to see how the real computer program looks like. Though it is targeted for young players of age 11 and above, it’s potential to be used as a tool to aid undergraduate students to learn basic computer programming was investigated. Both qualitative and quantitative techniques were employed to collect the necessary data for the study. Twenty fresh undergraduate students were divided into 5 groups to play the game. The game sessions were closely observed to note their reactions and gestures, their comments were tape recorded, field notes were taken and upon completion of the game, all participants were encouraged to answer a set of questionnaire. Besides obtaining input from the students, 5 computer programming lecturers were interviewed. The Likert Scale technique was employed in analyzing input of the students. From the finding, it was clear that there was positive response from the students. Most of the interviewed lecturers too liked the idea of using games to teach programming but have suggested several improvements for C-Jump to be used for tertiary level students. Several other issues were encountered while the C-Jump sessions were closely observed. Therefore, a few main recommendations were suggested to overcome some of the issues in concern. Nevertheless, the finding strongly indicates that there is some amount of basic programming knowledge to gain from the game.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

1.1 Introduction

“The demand for programmers is likely to increase steadily in years to come. Furthermore, the programs that needs to be written will continue to increase in complexity, requiring higher levels of software skills” (Barr and Tessler, 1996). Research has shown that the demand for software specialists in the modern age continues to increase day to day. This is due to the fact that many organizations are seeking new applications for computers and improvements to the software are already in use.

Educators are facing tremendous challenges in generating quality programmers to meet the high expectations of the computer industry. IT undergraduates endure immense pressure to become skillful programmers for the software industry. Many programming languages are being created for different purposes. The ability to program in different languages is often a prerequisite for employment in many reputable companies.

The process of learning new programming languages becomes much easier if there is a good foundation on the fundamentals of computer programming.

“Once you learn one programming langauge the others are fairly easy to pick up as well” (Dragontamer, 2005). Therefore, it is necessary to ensure that the undergraduate students strictly meet the objectives set for their programming languages courses especially the first programming course taught at the high school. The introductory programming courses at universities are often comprehensive due to the fact that there are many programming concepts, statements and syntaxes to be covered. Upon completing the introductory course, students are usually expected to demonstrate their understanding of the programming concepts and statements by writing complete programs. Many students usually do not meet the expected outcome. Novice students often complain that programming is difficult to learn, irrelevant to their lives and boring.

This project is focused to investigate how games can be used as a mechanism to support teaching and learning of programming at universities. There are lots of research attention to the use of games especially on both video and computer games in educational settings but there is least exploration on the use of physical board games in learning.

Board games are usually played for a variety reason such as social gathering, past time and for educational purposes. There aren't much board games in the market that are solely created and aimed to teach basic computer programming. Although, computer programming board games like Programmer's Nightmare and RoboRally do exist, but these games require sufficient programming knowledge to play (Cohn, 2005). C-Jump is a recently released computer programming board game, which is aimed to provide basic computer programming exposure to its players. This game could be used as a tool to teach basic programming to tertiary level students. Therefore, the objective of this project is to investigate the possibility and appropriateness of using the C-Jump board game as a tool to introduce computer programming to tertiary level students.

The study will be conducted by collecting the necessary data using suitable research methods and then interpreting the results to reach the consensus of whether the game should be used as a tool to aid students to learn basic programming. Firstly, the participant observation technique will be employed by getting groups of students to play the board game, while their verbal communication being tape recorded. Despite that, the field notes technique will be employed by getting the observer to note significant comments made by the participants. At the completion of the game, students will be given a questionnaire to evaluate the C-Jump board game as an educational tool. Apart from obtaining responses from the students, a few lecturers that have taught programming courses will be interviewed to obtain their opinions of using the board game to introduce basic programming to students. These research techniques are suitable to gather both students and lecturers views about employing the C-Jump to provide initial programming exposure to novice students.

1.2 Background

Teaching and learning computer programming especially the first programming language is a challenging task for both the lecturers and students respectively. Research has proven that educators contemplate various methods to make learning the first programming language a less painful experience. Novice computer programming learners are often bombarded with lots of new programming terminologies, concepts and the rigid syntaxes that scare them away.

One possible way to reduce the difficulty of learning the first computer programming language is to introduce the fundamentals of programming in a smart and fun way before the targeted group of students undertakes the first programming language course. C-Jump is a new innovative board game constructed to build the basic understanding of computer programming. It educates the basic commands of a programming language, such as the concept of variables, “if”, “else”, “switch”, “while” and “continue”. Learners will be able to see how the real computer program looks like. Although the C-Jump is targeted for learners of age 11 and above, it has potential to be used as a tool to introduce computer programming to tertiary level students. However, knowing that C-Jump is a new invention, issues involved in playing the game for educational purposes are to be investigated.

It is assumed that the board game would be great help to the novice students when they attend the formal lectures and labs of their first programming language course. There would be fewer jargons as the board game covers variety of programming commands and basic syntaxes. Students may not be able to write complete programs, but they would have a good understanding about the common statements used in writing programs.

1.3 Special Terms and Words

C-Jump – Refers to the computer programming board game being studied to determine if it could be used as an aid to teach basic programming to tertiary level students.

Skiers (or pawns) – Refers to the games pieces of the C-Jump board game. A game piece represents a player on the game board. In C-Jump, players can control one or two game pieces of the same color.

Game Board (or board) – Refers to the surface where the players play the game. Players would place the game pieces on specific position of this board, depending on the rules of the game.

Space (or square) – Refers to a physical unit of progress on a gameboard delimited by a distinct border. In C-Jump, each of the squares contains a programming programming.

1.4 Aims of the Study

The main aims of the study were to assess the practicality and suitability of using the C-Jump board game as a tool to introduce computer programming to tertiary level students and to make appropriate recommendations concerning any flaws indicated by the findings. The specific objectives were:

- To examine whether the rules of the board game are clear to the players.
- To examine the appropriateness of the content presented in the board game.
- To examine whether the players enjoy the game.
- To examine whether the players gain any computer programming knowledge while they play the game.
- To examine the presentation of the board game.
- To examine whether changes are necessary to the game's design and to make suggestions to overcome any problems encountered in the game.

1.5 Outline of Subsequent Chapters

Literature Review

The contents of this chapter can be divided into 2 broad categories: board games in general and the C-Jump computer programming board game. First half of this chapter provides basic information about board games, i.e. what is a board game, history of board games, nature of board games and modern board games. The other half reveals specific details about the board game being studied, i.e. history of C-Jump, board game design, look and feel, game rules, scope and people's views about this game.

Methodology and Findings

As the title explains, this chapter reveals information about the methodology and the findings. First, it provides details about the techniques used in gathering the necessary data for the study, i.e. methods of data collection, objective and the reason why these methods were chosen. Then, the collected data is systematically presented in tables.

Analysis

This chapter of the report provides details about the interpretation of the raw data which was collected and presented in the previous chapter (Chapter 3: Methodology and Findings). First, this chapter discusses students' view about the C-Jump board game and then it provides lecturers' opinion about using C-Jump for tertiary level students.

Issues, Recommendations and Summary

This chapter is divided into 3 different sections: issues emerging from the study, main recommendations and summary. The first section reports several issues about the game which was discovered throughout the study, the second section provides a few recommendations to enhance the C-Jump board game and finally, there is a short summary about the whole project.

LITERATURE REVIEW

2.1 What is a Board Game?

Board game, refers to any game played with a premarked surface, with counters or pieces that are moved across the board. A History of Board Games defines board games are defined as “any game played primarily on, but sometimes just near, a board of some kind” (2003).

2.2 History of Board Games

Board games have been consistently popular, and the origins of board games date back to ancient times. Over time board games spread to ancient Egypt, Greece and Rome, through Europe and eventually to the colonies of the New World.

The History of Sports and Games provides that the Senet (see Figure 1) of the Egyptians was one of the earliest board game**s** known and it was being played before 3000 BC. Senet was played by both the common and noble group of people and eventually it was believed to have taken on ‘religious significance’ (CanBooks, 2003).

Figure 1: Senet (CanBooks, 2003)

The Royal Game of Ur (see Figure 2) is the most famous board game known, dating from about 2500 BC (CanBooks, 2003). This ancient board game was discovered by a famous archaeologist, Sir Leonard Woolley, in the tomb of the royal cemetery of Ur (CanBooks, 2003). Wikipedia reveals that most of the games excavated by Leonard Woolley can be found at the British Museum in London (2006).

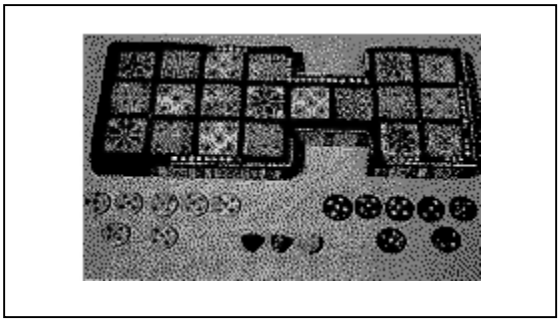

Figure 2: Royal Game of Ur (CanBooks, 2003)

The Go board game or "Wei-qi" in Chinese (see Figure 3) is another classic game known, as far back as 2300 BC that has maintain the same rules for longer than any other board game out there (A History of Board Games, 2003). The Go game have also spread into Korea, Japan sometime around the year 700 A.D. (A History of Board Games, 2003).

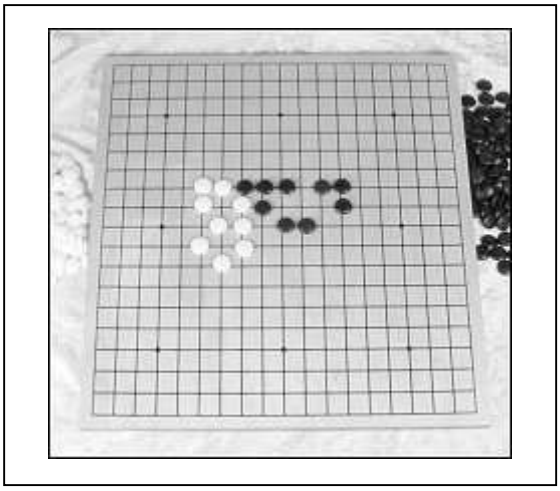

Figure 3: Go (A History of Board Games, 2003)

2.3 Nature of Board Games

Board games usually consist of boards, pieces, a system of rules and players. They have been used as a pastime as well as a tool to teach children and enlighten adults over a thousands years. There are many different types and classifications of board games.

The Histroy of Board Games provides that there are two categories of board games: strategy games and racing games (2002). The object of the strategy games that are to gain control of a larger board by using pieces to block or capture an opponent's pieces (The Histroy of Board Games, 2002). Chess, Risk and Monopoly are examples of strategy board games. The racing games are aimed to begin at a specific point on the board game and race along one or more paths to reach a specific goal finish line before your opponent (The Histroy of Board Games, 2002). Chutes and Ladders, Life and Parcheesi are examples of racing board games. Other board games do not fit in these categories. However, most are variations. For example, the objective of the board game Clue is to be the first player to solve a specific puzzle. Although the players are not trying to get to a specific location on the board, this is still a type of racing game.

Roland G. Austin in Greek Board Games claims that board games are generally based on the 'three primitive activities of man': the battle, the race, and the hunt (1940). The object of the battle, the race and the hunt game is to chase the opponents from the game board, bring all game pieces to a specific end position on the board and to escape from the opposing team respectively (Austin, 1940).

Wikipedia have different classifiication of board games: games that simulates aspects of real life and games that do not imitate reality (2006). Popular games that simulate aspects of real life include *Monopoly,* which is a rough simulation of the real estate market; *Cluedo*/*Clue*, which is based upon a murder mystery; and *Risk*, which is one of the best known of thousands of games attempting to simulate warfare and geo-politics. Other games that are not based on reality are abstract strategy games like chess and checkers, word games, such as *Scrabble,* and trivia games, such as *Trivial Pursuit*.

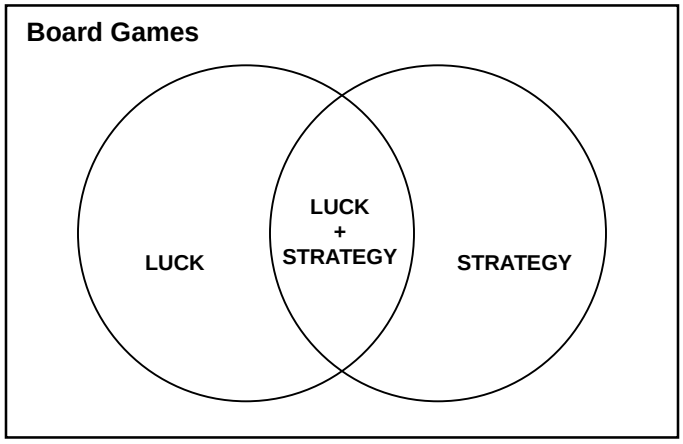

Figure 4: Types of board games

Board games are normally designed to involve luck, skill or both luck and skill. Figure 4 shows the 3 classifications of board games. The classic *Snakes and Ladders* is a pure luck based board game that doesn't require the players to make any decisions during the play. These sorts of games are often targeted for children. There are various methods of introducing luck in board games. The most common is using the dice (usually six sided). In the *Snakes and Ladders*, the number on the dice is used to determine how many steps a player move his or her token. Games like *Sorry!* use deck of cards to create randomness and *Scrabble* involves luck by requiring players to pick randomly pick letters.

The popular *Chess* board game is solely based upon skill. Most of the board games involve both luck and skill. *Monopoly* comprises both luck and skill as it uses the dice and it requires some thinking. Players may be losing in a game like this due to luck involvement, but a player with a superior strategy will win more often. Adult game players usually find purely luck based games quite boring and prefer games that require them to make some decisions. Table 1 gives a clear comparison of luck, skill and luck and skill based board games.

Table 1: Types of board games (Board Fun, 2005)

Luck	Luck and Skill	Skill
Usually involves spinners, dice, and cards, and doesn't involve strategy.	These games might include dice, spinners, and cards, but also include strategy and choices.	These games don't use dice, spinners, or cards, and have nothing to do with luck. Skill games take lots of practice and include strategy.
Easiest to develop	Harder to develop	Most difficult to develop
Parcheesi, Candy Land, Life	Monopoly, Sorry, Backgammon	Chess, Checkers, Go, Othello
These games usually are the simplest to play, because they don't require much thinking. Most of the time they turn out as a race.	Anything more complex than spinning a spinner, rolling the dice, or picking a card, would fall under the catagory of luck and skill. A luck and skill game must require some luck or it will fall under the category of a skill game.	These games consist of no luck at all. Skill games usually have many different rules, and give you many choices on where to move, which piece to move, and how to win. Skill games usually include many pieces, involve strategy, and require concentration to play

2.4 Why Board Games are better than Electronic Games

Board games are generally more educational than video games. Johnson in one of the Virginia Cooperation Extension's news release reports that Thomas Sherman, a professor of teaching and learning, suggests that parents should consider buying the traditional board games, books and activity equipments over electronic games for their children (2001). The authors further reports that there are several guidelines to ensure positive video gaming with young children and teens. Some of these guidelines include setting up the game equipment in a social setting, ensure the games are suitable for the desired age level, limit the game play time to two hours and have "regular quite time" for other activities.

Many publications allege that computer games especially simulation games have high educational value over other types of games. Nonetheless, these games often are too complex. Saari (2004) states that "board games have one advantage over computer

games: players know the rules". This implies that board games are simple as there are no hidden rules. Whereas, in computer games the players are expected to enter some input to obtain some output "based on rules they can't be sure of" (Saari, 2004). In other words, nobody knows exactly know how the game engine is being designed unless one has access to the source code.

BoardGameGeek is an online resource providing lots up-to-date information about games and it contains an active discussion forum. *Why Board Games are Better than Video Games* is one popular subject of discussion in the forum. Below are selected opinions about board games from various game enthusiasts.

- Cheaper than video games
- Increase in value with time than video games that decrease in value after some time
- Wireless or it can be played in the complete absence of electrical power unlike video games
- Easier to be designed than video games as it does not involve programming
- Bring people together, 2 or more players play the game together
- Non-violent
- Do not cause payers to strain their eyes in front of their monitors
- Compatible ever unlike other games that involve serious compatibility issues
- They are real as players can touch the components, deal cards and feel the tokens
- They are mostly quite
- Accessible to everyone as they do not come with specific ratings like U and 18SX
- Do not cause Attention Deficit Hyperactivity Disorder (ADHD)
- Players to easily change the rules to suit a group, fix a flaw in gameplay, or just make a game more fun easily.
- Handy and easily transported
- They are unique or completely different every time

2.5 Modern Board Games

In this modern era of technology, computers have profoundly impacted the traditional board games, as most of these board games are now computerized. Like any other computer games, board games can be downloaded, installed and played on personal computers. Similar to other online games, these board games can also be played online against other players located remotely.

Many researchers seem to support the idea of computerizing existing board games for several reasons. Morrison explained that computer relieves tedious task of calculating the results of a battle (2005).

"By speeding up these tedious offline tasks, a computer allows players to focus more on their strategy and actual game playing, and less time on the accountant bean-counting at the end of a turn" (Morrison, 2005).

In older board games, players might have read all the cards and the game never change (Learn to Love Board Games Again, 2005).

Modern board games are designed differently. In modern board games, players compete until the end of the game unlike older board games, where players drop out of the game before it ends and wait for other players to finish (de Boer and Maaten, 2004). Nevertheless, most modern games are designed closely to original board game. Often, the physical look and rules of the game are well preserved but how the players interact in the game differs. For instance, instead of having the players to throw the dice, they are now required to click at the dice on the screen. The dice is programmed to randomly display a number ranging form 1 to 6.

The use of Artificial Intelligence (AI) allows modern board games to be more intelligent. Some of the board games specifically *King Arthur* of designer *Rainer Knizia* has gone to

the extend of incorporating “intelligent electronics” that give feedback depending on decisions made by player (de Boer and Maaten, 2004).

Clim J. de Boer and Maarten H. Lamer in their study of Electronic Augmentation of Traditional Board Games (2004) have proposed a conceptual framework for the development of modern board games called *self-conscious game board*. The proposed concept of a self-conscious game board is built around the idea of incorporating the ability to recognize the state of the game and to provide appropriate feedback (de Boer and Maaten, 2004). The framework was tested by electronically enhancing an existing board game, *Settlers of Catan* and the outcome showed positive results.

“Through a case study in which *Settlers of Catan* was electronically enhanced, it was proved that the proposed concept of a self-conscious gameboard is viable and capable of heightening a board game’s appreciation, particularly through dynamic changing of the game board. The case study also showed player’s positive reception of the unexpected new possibilities to customize a game to their own liking. (de Boer and Maaten, 2004)”

The next generation of board games will blend the coolness of video games and the social approach of video games. Currently, research efforts are focused on the development of multi-touch interaction which will allow multiple players to touch the game board instead of moving the physical game pieces or pawns by hands.

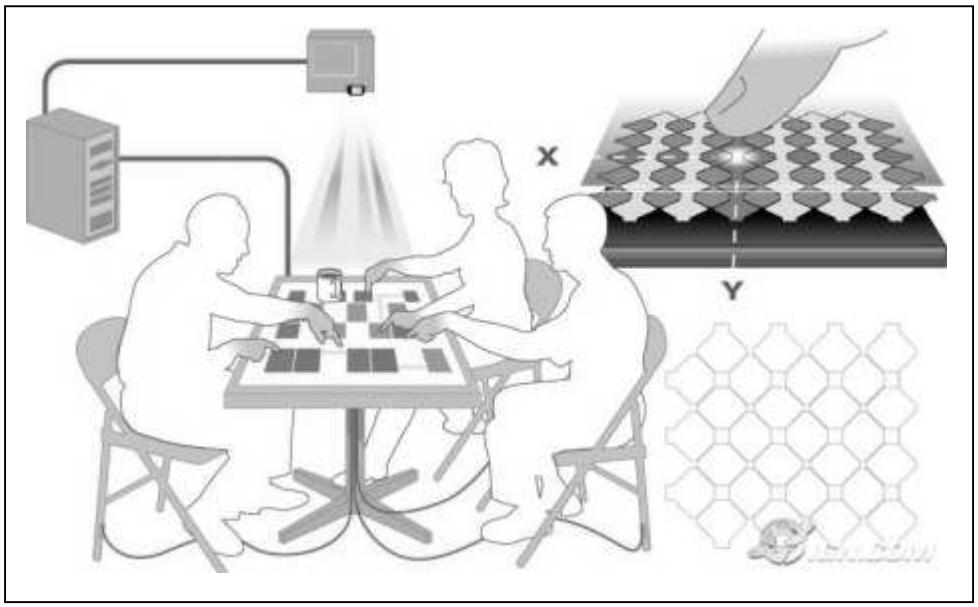

Figure 5: DiamondTouch by Mitsubishi (Block, 2006)

Mitsubishi have released an invention called *DiamondTouch* (see Figure 5), which is not really a touch screen because the display is coming from a video projector, but several players can touch the screen simultaneously (The Future of Baord Games, 2006). The Mitsubishi Electric Research Laboratories as quoted by Block, 2006 claims that:

"DiamondTouch is front-projected and uses an array of antennas embedded in the touch surface. Each antenna transmits a unique signal. Each user has a separate receiver, connected to the user capacitively, typically through the user's chair. When a user touches the surface, antennas near the touch point couple an extremely small amount of signal through the user's body and to the receiver. This unique touch technology supports multiple touches by a single user (e.g., two handed touch gestures) and distinguishes between simultaneous inputs from multiple users. DiamondTouch tables are available in two sizes (32" diagonal and 42" diagonal display), while custom sizes and shapes are available on spec."

Figure 6: Entertaible by Philips (Entertaible, 2006)

Besides DiamondTouch, there is another promising creation which is still a working prototype by Philips called *Entertaible* (see Figure 6). Entertaible is a "30-inch LCD screen embedded in a table displays the board and uses infrared sensors to detect how the players move their pieces" (Board Games of the Future, 2006). Players are required to roll the virtual dice and move the clear cubes (pawns) around the screen according to the arrows that shows where they can move (Entertaible, 2006).

2.5 About the C-Jump Board Game

2.5.1 Introduction

C-Jump is a new released educational board game in the board game industry that introduces the fundamentals of computer programming to its players. The theme of C-Jump is *Skiing and Snowboarding Race* and it has a catchphrase that impart "*Race down a mountain, think like a computer programmer!*" The goal of the game is to find the most efficient way to "ski" down a mountain. Players are to image themselves as either skiers or snowboarders, racing with each other to reach the finish line. The catch is that the player must make decisions based on common computer programming syntax, such as "if(X==1)" you can go down a certain path. The first player to move all pieces past the finish line would be the winner.

The C-Jump was created and released in 2005 by a Computer Programmer named Igor Kholodov. The initial idea of creating such game was triggered to Kholodov in 1999 when he wanted to teach the basics of programming to his son. "I was fired up by my son's interest and went to Toys "R" Us and the Discovery store hoping to find some educational toys and was surprised there was nothing out there," said Igol Kholodov (Cohn, 2005).

The game is manufactured by the C-Jump Factory, a company based in Braintree, Massachusetts. The company specializes in the manufacturing of educational games for children, college students, and adults. Today, C-Jump is being commented and discussed in many articles in popular websites such as Wired News, MSDN, Popgadget and BusinessWeek.com.

The game is designed for learners between the ages of 11 and above, who are interested to learn the basics of computer programming. The game can be played with a minimum of 2 players to a maximum of 4 players. It takes about 30 minutes to complete a single

game session with 2 players and an additional player would incur about 5-10 minutes of extra time (C-Jump, 2005).

The game helps to develop the basic understanding of a complete computer program, formed by logical sequences of commands. Players will indirectly grasp the basic computer programming commands as they make their moves in the game. The game teaches the players basic commands of programming languages such as the basic concept of variables, if, switch and while. Regular players will become familiar with these commonly used commands used in programming languages like as C, C++ and Java.

"By moving around the board, entering loops, branching under conditional and switch statements, the players gain physical experience of a complete program. Understanding of the internal action of a computer is essential to understanding what software is. Static program causes dynamic process in the computer. By playing the game, players see this process as physical and spacial motion. (C-Jump, 2005)"

The C-Jump website claim the following facts about the game: "This game is not only about teaching and learning: it's fun and entertainment for the whole family; skiing and snowboarding is a perfect programming analogy; C-Jump game is ideal for home school education; the game is based on the code of a real computer program. (C-Jump, 2005)"

2.5.2 Board Game Design

The C-Jump is designed based on the classic racing Snakes and Ladders board game where a player's game piece follows a track from start to finish. The players race with each other to reach the finish line. It is built using the similar concept of having the players to roll the die and then move the game pieces to some specific location on the game board. C-Jump comes with a *Start* space at one corner of the game board and a *Finish* space at the other far corner of game board. Spaces on the board are shown as squares. Instead of having a number in the spaces of the game board, each space has a statement of a rule, borrowed from the C programming language such as the "if", "else",

“switch”, “while” and “continue”. These statements are sequenced to form a complete computer program.

The *Snakes and Ladders* contains snakes of different lengths, which could slow down players from finishing the game. The “goto” statement in C-Jump acts in similar way in the sense that it changes position of the game piece from better to worse.

It is vital to understand how the players should make their moves in the C-Jump. Most of the spaces in C-Jump contain statements using the “x” variable. So, before the player proceeds, he or she is required to calculate the number of steps by replacing “x” with the number rolled on the die. Lets us assume that the player’s game piece is located on a space containing the “x + 2” statement and the number shown on the die is 4. Since the result of the addition is 6, the player would need to position the game piece counting 6 locations from the previous space. If at all the piece ended on a space containing conditional statement like the “if”, “switch” and “while”, the player would need to roll dice again to decide the next path. So, different statements have different instructions for the movement of the game piece. Therefore, players are expected to read the board rules carefully in order to advance the game correctly.

2.5.3 The Look and Feel

The game set comes with 1 game board, 1 die, 8 playing pieces (of 4 different colors: red, green, blue and yellow) and a copy of game rules. Since it comes with 8 pieces, players are required to decide the number of pieces to be used in the game. They have a choice to use either 1 or 2 pieces of the same color to represent each skier or snowboarder.

Figure 7: Front cover of the game box (C-Jump, 2005)

The game box comes with an attractive and promising look that at a glace sets a good impression towards the game (see Figure 7). The game box cover clearly illustrates that theme of the game, which is Ski and Snowboard Race. The top right portion game box is presented with larger picture of a snowboarder in a race. The top left portion of the game box is depicts a simple computer program that explains that the game is related to real computer programming and it has an educational purpose. It provides all the necessary information about the game: the title, targeted players, equipment and the website address.

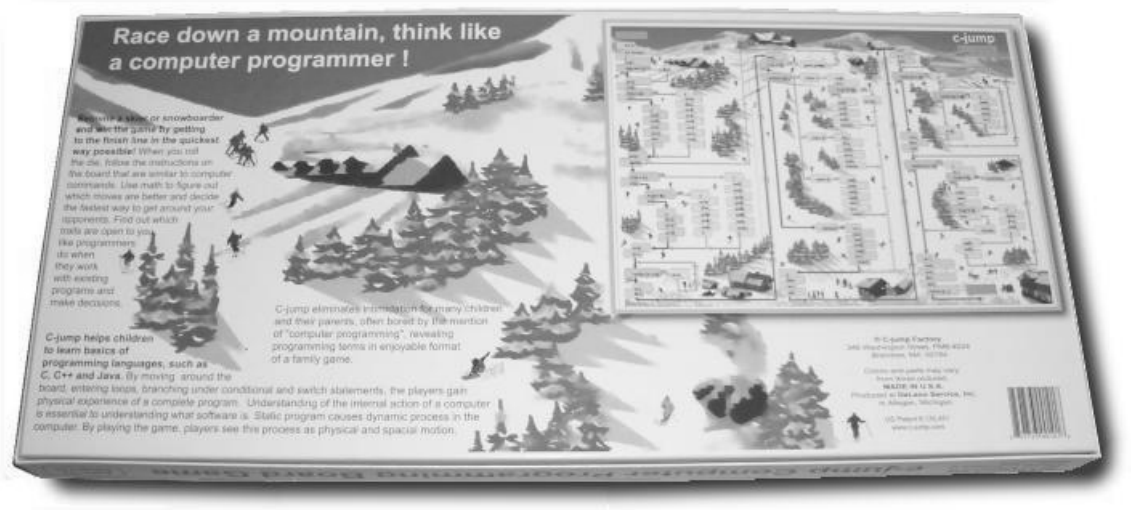

Figure 8: Back cover of the game box (C-Jump, 2005)

The back cover of the game box (see Figure 8) provides a general idea about the board game. Specifically, it dictates the basic rules of the game and the educational benefits of playing the game. The motto of the game “Race down a mountain, think like a computer programmer!” is clearly presented at the top left corner of the game box and a snapshot of the whole game board is displayed at the right portion of the back cover.

2.5.4 The Game Board

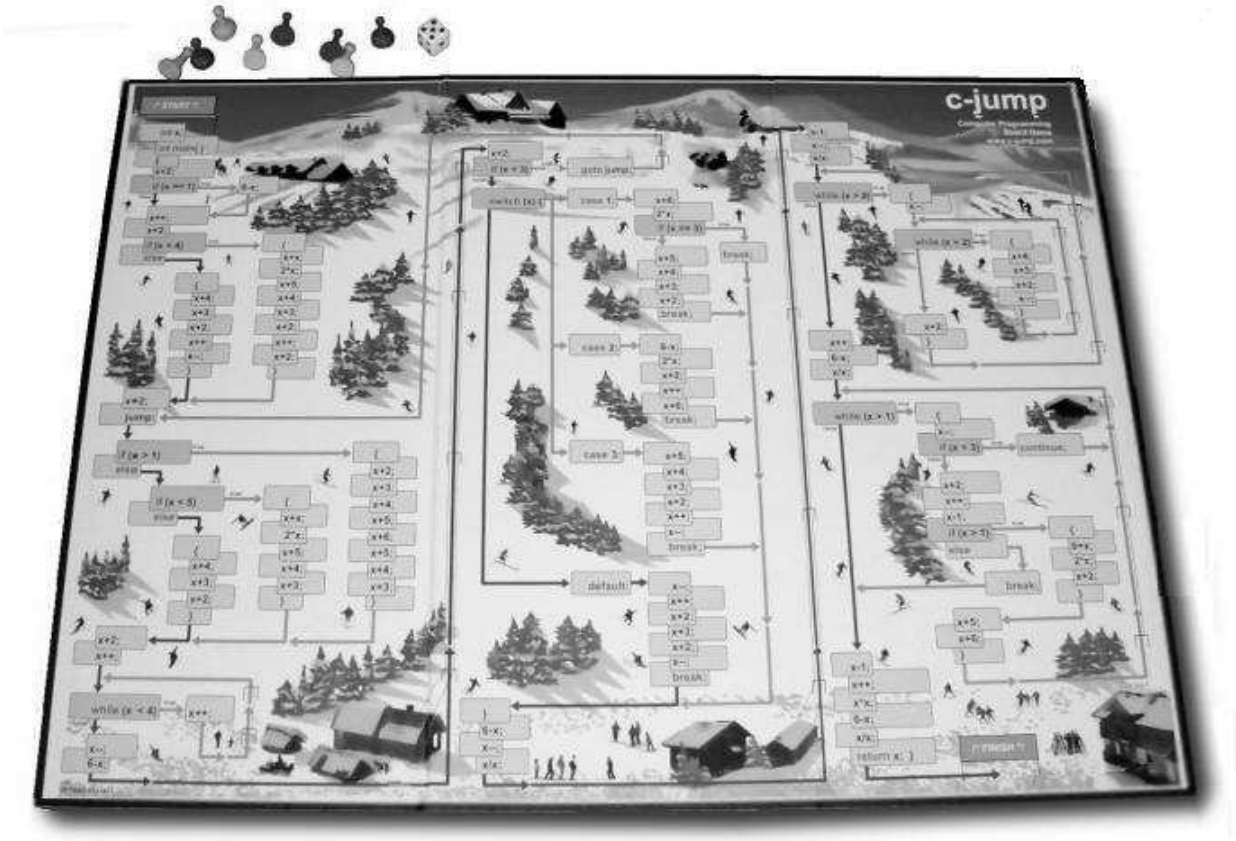

Figure 9: The game board (C-Jump, 2005)

As you can see in Figure 9, the background of the game board depicts a large view of mountains filled with snow. The mountains are being used for the purpose of skiing and snowboarding. In the board, there are many people caring out the activity of skiing and snowboarding from the top till the bottom of the mountain. There are added sceneries such as the pine trees and mountain ranch that makes the desired atmosphere to suit the theme, a ski and snowboard race adventure.

The game board is presented with 145 spaces containing various programming statements. These statements are linked with arrows that will guide players to reach the finish line. The statements actually form a large computer program. Players are supposed to navigate their game pieces to the finish line by placing the pieces on the designated spaces based on the number shown on the rolled die and the instruction of the previous statement.

The spaces containing the programming statements come in 3 different colors: blue, orange and gray. The orange spaces contain conditional statements such as the “if”, “switch” and “while”. If the players end their movement of pieces at orange spaces, they are awarded a free roll to determine the next path, which could either be the *true* or the *false* path. There is no significant difference between placing the pieces in the blue or gray spaces. However, the blue spaces are sequenced to be the direct path to the finish line and the gray spaces are statements on the *true* path of the conditional statements. The arrows connecting the statements come in 2 colors: blue and orange. The blue arrows represent the direct or shortest path to the finish line whereas the orange arrows are other paths in the game.

2.5.5 The Rules

Skiers and snowboarders line up at the start location in the game board. Player rolls the die and moves one of his or her skiers to new position by counting off the number of spaces. Players can play the game with 1 or 2 skiers for each player. If each player is represented with 2 skiers, players may choose any of their skiers to move. The first player to move all his or her skiers past the finish line would be the winner.

Ultimately, it is best to follow the blue trail, which is represented by blue arrows. The blue trail is the direct path to the finish line. However, the “if”, “switch” and “while” statements on the game board may cause the players to follow the orange trail that may cause delay in reaching the finish line.

The game board spaces contain different types of programming commands. So, there are different ways of counting the number of moves to be made to determine the new position of the skier. Some statements require the player to replace the 'x' in statement with the number rolled on the die to determine how far the skier should be placed. Certain statements merely require the player to move according to the number shown on the die. The conditional statements like "if", "switch" and "while" statement requires the player to dice again, test the condition and move accordingly. Other statements such as the "goto", "continue" and "return" statement forces players to position their skiers at specific locations on the game board. The complete set of instructions that comes along with the C-Jump board game can be found at Appendix section of this report. An animated application of the game rules can now be downloaded from website of C-Jump (www.c-jump.com). Table 2 summarizes the game rules as to how the player should move his or her skier for a particular type of statement.

Table 2: Types of movements in C-Jump

Programming Statement	Example of Statement	Type of Movement
if	if(x==1)	The player is awarded a free roll; replace 'x' with the number rolled on the die and then test the condition.
while loop	while(x>0)	
switch	switch(x)	
variable declaration	int x	Move downhill accordingly to the number rolled on the die.
function declaration	int main()	
open curly brace	{	
close curly brace	}	
label	jump:	
case	case 1:	
else		
break		
default		
plus operator	x + 2	Replace 'x' with the number rolled on the die, perform the operation and move accordingly.
increment operator	x++	
minus operator	6 - x	
decrement operator	x--	
multiply operator	x*x	
divide operator	x/x	
goto	goto jump	Place the skier to the square labeled 'jump:'
return	return x	The skier moves pass the finish line.
continue		Place the skier back to 'while'.

2.5.6 Depth of C-Jump

Table 3 illustrates types of programming statement found in C-Jump. It consists of 21 types of programming statements. In total, there are about 145 statements on the game board. The table also shows the number of occurrence of a particular type of statement.

Table 3: Programming statements in C-Jump

No	Programming Statement	Example of Statement	Number of Squares on the Game Board
1	if	if(x==1)	8
2	while loop	while(x>0)	4
3	switch	switch(x)	1
4	variable declaration	int x	1
5	function declaration	int main()	1
6	open curly brace	{	10
7	close curly brace	}	10
8	plus operator	x + 2	52
9	increment operator	X++	11
10	minus operator	6 - x	9
11	decrement operator	x--	10
12	multiply operator	x * x	6
13	divide operator	x / x	4
14	goto	goto jump	1
15	label	jump:	1
16	case	case 1:	3
17	return	return x	1
18	else		4
19	continue		1
20	break		6
21	default		1
		Total	145

The following chart (Figure 10) clearly illustrates the frequency of programming statements in C-Jump. It is obvious that many of the statements are arithmetic statements involving the plus (+) operator.

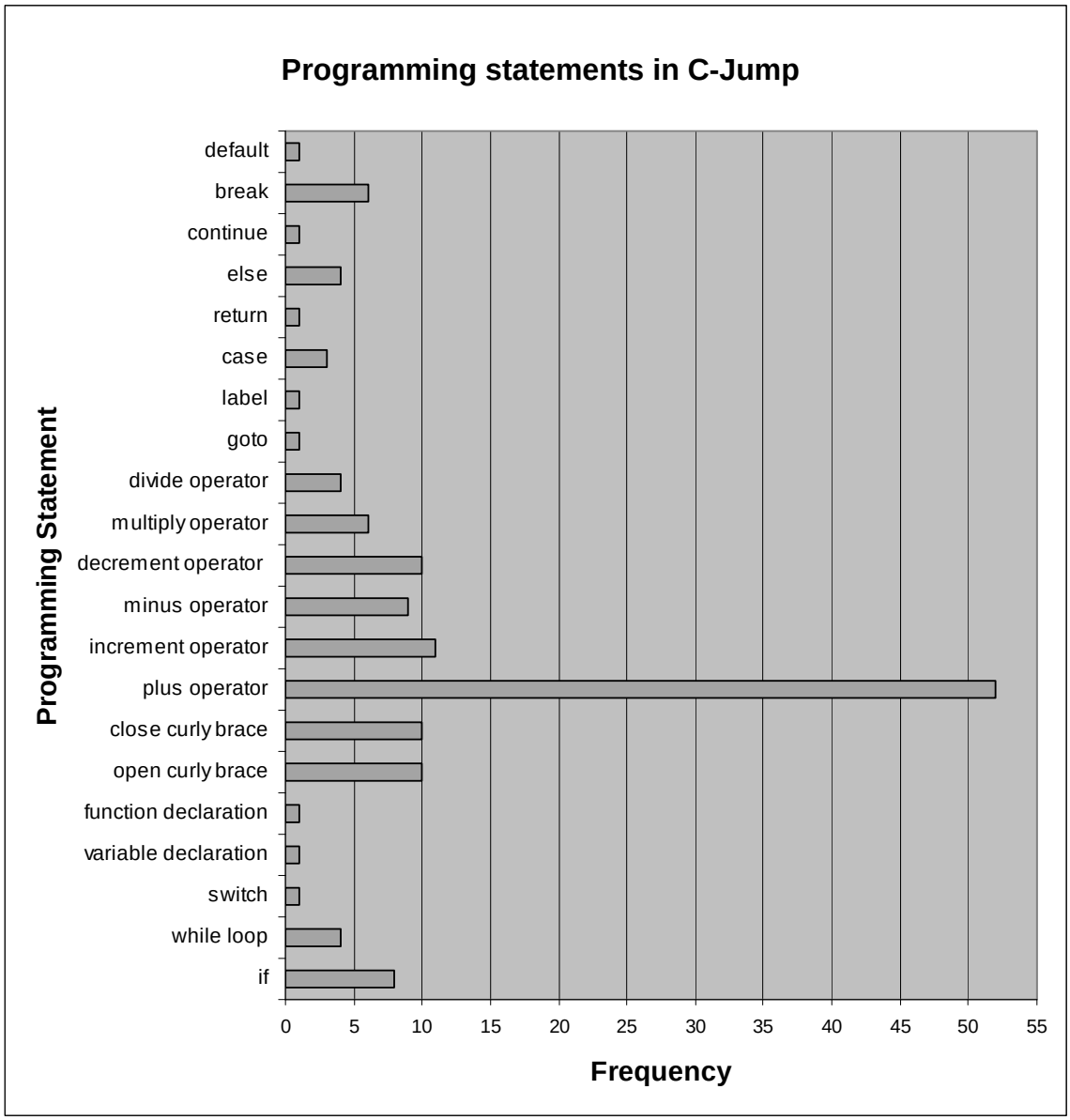

Figure 10: Programming statements in C-Jump

2.5.7 People's Views about the C-Jump

There seems to be contradicting views about the educational use of C-Jump. Some people find it as a worthwhile innovation whilst others do not see the point of having one. The following opinions are quoted from various sources on the Internet.

"This is a smart and interesting way to teach 'simple computer programming syntax' to students. (Smith, 2005) "

"I don't really see a whole lot of value in teaching children through this method as it introduces the concepts pretty abstractly and in a way that is not representative of a real computer program. Furthermore you're not *doing* anything other than adding and subtracting from the die roll so I do not believe many children will really see any point to it (at least in Monopoly you're bankrupting your friends and in Mouse Trap you get to set off the cool Goldberg device). (BoardGeek.com, 2005) "

"Wow, I've gotta say, as a programmer, this seems like a great way to introduce my non-geeky friends and wife to the joy of programming, and kids too. (Deanh, 2005)"

Below are selected concerns and opinions by some commentators of C-Jump quoted from JWZ (2005):

- "X isn't quite a variable. Take a look through the rules."

- "Great idea, but not well done. Look at the numbers in the if and switch statements here. The first two cases can't be reached."

- "Where's the danger? A game without danger is no fun at all. A race is all well and good, but what players really want is to avoid going to jail or to score the community chest or pawn the most valuable properties. Introduce some security

concepts. Get kids used to the idea of a buffer overrun (skiing off a cliff?), poor input validation (forged ski ticket?), and other things …”

- “Great idea, but not well done. Look at the numbers in the if and switch statements here. The first two cases can't be reached.”

- “Where's the danger? A game without danger is no fun at all. A race is all well and good, but what players really want is to avoid going to jail or to score the community chest …”

- “How does "roll a die and do whatever the game tells you to" qualify as a fun game?”

- “Does the game ever halt?”

- “The scheduler worries me: With multiple skiers of the same color on the board, players may choose any of their skiers to move."

- “… i think i just fell asleep reading the rules.”

METHODOLOGY AND FINDINGS

3.1 Introduction

Several research techniques were employed to gather the necessary information for the study of C-Jump. Data was collected in various forms by observing students playing the game, tape recording their comments, making own field notes, distributing questionnaires to get feedback from the players and also interviewing educators that have taught programming courses.

3.2 Participant Observation

Twenty IT undergraduate students who were enrolled for the first programming course, CSEB114 Principles of Programming at Universiti Tenaga Nasional (UNITEN) were scheduled to play the game while their reactions and gestures are being closely monitored. Objective of the observation was to perceive any interesting issues involved in playing the game. There was only 1 C-Jump board game available for testing. Due to this, a simple schedule was prepared to get different groups of students to play the game. The C-Jump could only engage 2 to 4 players in one single game session, and therefore the participants were divided into 5 groups with each consisting of 4 players.

During the exercise, actions and gestures of each of the group member was carefully observed. Each group observation result was tabulated into 3 parts: early game, mid game and end game (see Table 4). Indirectly, the observation was useful to notice any observable pitfalls of the game.

Table 4: Observation results

Group 1	Early game:	- Only 1 skier per player. - Players directly started to read the instructions. - Repeatedly read the instructions. - Continuously added the value of the previous x. - No clear idea on how to start the game, restarted the game several times. - Players looked frustrated.
	Mid game:	- Players played very inconsistently, sometimes evaluating the conditional statements halfway through every move and sometimes only when the skier was placed there. - One of the players did place his skier outside the provided squares. It seemed that the orange arrow was not noticeable.
	End game:	- Players counted backwards when the number of steps was big, thinking it is similar to the classic Snakes and Ladders. - Skiers already at “return x” but player still continued to playing the game.
Group 2	Early game:	- Only 1 skier per player. - After placing the skier on the squares, players made attempts to understand what the statements meant by reading the instructions.
	Mid game:	- Player kept evaluating the conditional statements halfway of every move. - Players look excited.
	End game:	- Players not sure how to proceed when their skier was placed on the “return x” statement. They were confused about where is the “ski base” when they referred the instructions for guidance.
Group 3	Early game:	- Only 1 skier per player. - First, players continuously added the value of the previous x. - Then, players made moves according to the number shown on the dice, disregarding the statement in each square.
	Mid game:	- Player kept evaluating the conditional statements halfway of every move. - Players look excited.
	End game:	- Players counted backwards when the number of steps was big, thinking it is similar to the classic Snakes and Ladders.
Group 4	Early game:	- Only 1 skier per player. - Players immediately read the instructions to know what the statements meant.
	Mid game:	- Player kept evaluating the conditional statements halfway through every move.
	End game:	- At the “return x”, players still continued to roll the dice to obtain a 1 to finish the game.
Group 5	Early game:	- Only 1 skier per player. - No clear idea on how to start the game, restarted the game several times. - Players look puzzled.
	Mid game:	- Player kept evaluating the conditional statements halfway through every move.
	End game:	- Players repeatedly read the instruction to know what “return x” means.

3.3 Tape Recording

Participants' verbal communication while they were playing the game was tape recorded. The purpose of tape recording the game session was to ensure that every single comment made by each player was noted without fail. Certain comments that have frequently occurred were taken as more important than if it only rarely occurred. The number of times a particular comment or similar was counted and comments that turns out to be the greatest was then be identified as the greatest problem in the game. Player's verbal comments while playing the C-Jump board game are showed in Table 5.

Table 5: Players comments

- Why do we have 2 pawns of the same color?
- I am ready to bet what we did is all wrong.
- The instructions are not clear.
- The instruction doesn't explain how we should move.
- We don't have any clue about how we should play the game.
- Do we continue adding the x value?
- What do return x actually means?
- I feel something is wrong somewhere.
- Where is ski base?
- How do we start, any tips for beginners?

3.4 Field Notes

This method is about making own notes against the comments made by members of each group while they play the board game. The objective of making own field notes while the participants played the game was to identify any significant comments that may not be the most frequent but gives an insight of the game studied. The field notes results are shown in Table 6.

Table 6: Field notes results

- The game is teaching us something, before this we never know what is x+2.
- We only learn basic arithmetic from this game.
- Players have limited things to do in the game, so there should be more options.
- This game is boring.
- We can't see the motive of playing this game; we are not sure where are we heading.
- How does skiing relates to programming?
- How can we learn C++ and Java from this board game?
- This game is useful for people who are in the process of learning programming. It will help beginners to visualize how certain programming statement work. I.e. how the x++ statement works.
- The arithmetic statements of C-Jump should be replaced with other types of statements.

3.5 Questionnaires

Upon finishing the game, every participant was requested to complete a set of questionnaire consisting of several statements relating to five general areas: playability, rules, learning value, fun factor and aesthetic appearance of the game. The objective of the questionnaire was to obtain students' feedback towards the C-Jump board game specifically on the 5 mentioned areas. These statements were used as the main instrument in obtaining a good insight about what students feel about the game. This technique was also useful in obtaining personal views from the participants especially from those that were less expressive during the game session.

The following statements were given out to the participants after completing the C-Jump session. Participants were asked to evaluate the extent to which they agreed with the statements on a scale from 1 to 5 (1, strongly disagree; 2, tend to disagree; 3, neither disagree nor agree; 4, tend to agree; 5, strongly agree). In addition, a general comment section was included for any additional suggestions or comments.

- *Statement 1: C-Jump is playable board game.*

The above statement of the questionnaire deals with the playability factor the game. The objective of this statement was to determine if C-Jump is accepted as a game that can be played by people.

- *Statement 2: Rules of the C-Jump board game were clear enough for players to understand how the game should be played.*

The above statement of the questionnaire relates to clarity of rules of the game. The objective of this statement was to determine whether the rules of the board game were unambiguous and easily understood by the players.

- *Statement 3: I have learnt the basic programming concepts and syntaxes from playing the C-Jump board game.*

The above statement refers to the educational value of the game. This statement was acquired to measure the programming knowledge gained from playing the game.

- *Statement 4: C-Jump is an interesting and fun board game.*

The above statement of the questionnaire is about the fun factor the game. This statement was aimed to investigate whether the players were having fun while they were playing the board game.

- *Statement 5: C-Jump board game looks attractive and appealing to me.*

The above statement refers to the aesthetic appearance of the game. The objective of this statement was to determine the consensus about look and feel of the board game.

- *Statement 6: Please write in anything else that you would like to tell us about the C-Jump board game.*

The above is an added statement in the questionnaire, which was aimed to take note of any other information related to the C-Jump board game that the participant wants to highlight.

Responses for the first 5 statements are tabulated in Table 7. The mean score and standard deviation are also calculated and displayed accordingly. These results is referred and discussed in great detail in the following chapter (Chapter 2: Analysis). Other comments made by the respondents are displayed in Table 8.

Table 7: Response for the five evaluation measures (Question 1 to 5)

No	**Question**	**1** (strongly disagree)	**2** (tend to disagree)	**3** (neither disagree nor agree)	**4** (tend to agree)	**5** (strongly agree)	**Mean Score** ($\bar{x}$)	**Standard Deviation** (σ)
1	C-Jump is playable board game.	2	1	4	8	5	3.65	1.19
2	Rules of the C-Jump board game were clear enough for players to understand how the game should be played.	3	7	4	4	2	2.75	1.22
3	I have learnt the basic programming concepts and syntaxes from playing the C-Jump board game.	0	2	5	11	2	3.65	0.79
4	C-Jump is an interesting and fun board game.	0	2	5	8	5	3.8	0.93
5	C-Jump board game looks attractive and appealing to me.	1	2	5	9	3	3.55	1.02

Table 8: Comments from respondents (Question 6)

No	Question	Participants Response
6	Please write in anything else that you would like to tell us about the C-Jump board game.	- C-Jump should be made more comprehensive and complicated if it will be used by undergraduate students. - The game rules are confusing but it is good for students to start learning computer programming. - C-Jump features the basic code programming knowledge which will definitely help to increase the knowledge of students. - The theme “skiing” won’t attract much attraction. - An interesting game that teaches programming but it isn’t suitable for people in Malaysia, as we are far behind compared to others. - It is not suitable for children below 11 and it is suitable for student age 12 above and for those have programming knowledge. - C-Jump board game must have a complete set of rules so that it can be played by both children and adults. - At first, I couldn’t understand the game and it appeared boring to me but later when I understood it, its fun! - Need to make this game more thrilling and interesting with various options. - The game finishes very fast. - I don’t understand the main concept of the game. - A good and challenging game. We didn’t know at first how to play it but after been taught, it was fun and interesting. We should play these games more in the class routines. - The game was enjoyable but would be more interesting if the instructions were clear. - The game is like a maze for those who just started learning programming. - The game is fun but the instructions are confusing.

3.6 Expert Interviews

The C-Jump board game is being carefully examined to see the possibility of using it as tool to aid students to learn basic computer programming. Besides gathering feedback from the novice students, it was fairly important to obtain what the lecturers teaching the programming course feel about using the C-Jump in their classrooms. Five lecturers teaching programming languages at UNITEN were interviewed. During the interview, the C-Jump board game was first played by the lecturers and then the following questions were answered. Lecturers' responses for the five questions are presented in Table 9.

- *Question 1: What are some of the frequent problems faced by students taking the first programming course?*

The above question was aimed to identify common problems experienced by students taking the first computer programming course.

- *Question 2: What is your opinion about using a board game to introduce basic computer programming to students?*

The above question was aimed to determine lecturers' general opinion about using a board game as tool to support the traditional methods of teaching programming to students.

- *Question 3: What is your opinion about using the C-Jump board game to introduce basic computer programming to students?*

The above question was posed to determine lecturers' personal opinion about using the C-Jump to introduce programming to tertiary level students.

- *Question 4: What are your comments about the learning content (programming statements and the syntax) provided in the C-Jump?*

The above question was aimed to determine the appropriateness of the basic programming content covered in the game.

- *Question 5: What are your suggestions to improve the C-Jump board game so it could be used as tool to introduce programming to tertiary level students?*

The above question was raised to determine lecturers' suggestions to improve the C-Jump board game as an effective tool to introduce basic programming to IT undergraduates.

Table 9: Expert interviews' results

No	Question	Response from the Lecturers
1	What are some of the frequent problems faced by students taking the first programming course?	L1: Students face problems to understand logics and syntax. L2: Students face problems in understanding the programming concept. They don't know how to learn programming languages. They just read their notes and they don't try the exercises. They can't apply what was taught in the class when they are given problems to be solved. L3: Students' main problem is to solve problems and to come up with the algorithm. They also cannot visualize how the syntax works, for instance how a loop executes in a program. They do not know where to start in programming and they have problems in memorizing the syntax. L4: The students do not understand about problem solving. Students don't have good knowledge of how to solve the problem. Some of maybe good at the syntax but knowing the language doesn't help them to solve any problem. We also have a group of students who cannot learn the syntax easily; they find the language itself is difficult. They are carried away with notations, semicolons, quotations, punctuations. Currently, a lot of our students have difficulties with the syntax. L5: We are responsible our selves as educators to follow the syllabus, and we right away try to teach them coding instead of programming. As a result, the students with the characteristic of being so visual are forced to imagine something abstract i.e. the coding system, syntax of the language, declaration of the variable, how the computer organization works, why you need declaration and etc. Most of the programming books start with "Hello World!" and students tend to wonder what that actually means. Instead of explaining programming, the book explains that the printf statement prints, scanf reads data and then it will print "Hello World!" at the end of this. The students are then lost into syntactical events i.e. semicolon, braces, quotes, etc. The students are confused between programming and coding which makes them to dislike programming. I believe it is because of our approach, we must teach them programming before coding.
2	What is your opinion about using a board game to introduce basic computer programming to students?	L1: Game is an interesting tool to attract the interest of students to learn programming. It is a good start to introduce anything learning process. It is a good tool to improve learning. L2: It can be done outside of formal lectures hours or we can have it as a quiz or something in groups, who wins faster will be the winner. It can be used to make learning more interesting but it is just a supplementary and optional thing, which can't replace the formal lectures. It can probably get the interest of students who dislike programming to entice them in learning. L3: Game is a good alternative for teaching as compared to the

		PowerPoint presentations. It would be more fun to use a game to teach programming. L4: It is a good idea but we must have proper tools and infrastructure. We must make sure that the game has good regulations or rules. Anyhow, we can't standardized the way we teach because some students like games and some may not like to play games to learn things; students who do not like games may prefer the formal education. Games are only meant for a different target group who love to play games. So, you can use that advantage of people who like to play games to instill knowledge that will be beneficial to them in the education system. However, students may not go through certain sequences or they may take shortcuts to reach the final finishing line. Students maybe keen in moving because they want to win and probably they did not really observe the steps. Students may just want to win the game but not to learn from that game. L5: I haven't seen a good game which can teach programming. I don't play game anyway so I am totally illiterate in terms of games. I am from the older generation, I never believe that and I don't like it.
3	What is your opinion about using the C-Jump board game to introduce basic computer programming to students?	L1: Generally, I would very much encourage the use of this game to introduce programming. Perhaps, the C-Jump could be used during the first week of the programming course. Usually, we will not cover much in this week. This is to create the interest in the students despite giving them basic idea of what programming is all about. If C-Jump is played regularly, each statement presented in the game will automatically be installed in the players mind. For instance, the player will know how a switch statements work. Students will learn the basic programming statements by playing the game. L2: The person who loses in the game would learn more than the person who wins the game. C-Jump doesn't meet the objective of the game, which is to get all players to learn the fundamentals of computer programming by playing the game. Players could possibly escape all the conditional statements in the game. Students playing this game must be guided by a teacher. These students may play the game wrongly. They might also interpret the programming statements incorrectly and may then carry the wrong interpretation with them. L3: The game is presented very well. After playing the C-Jump, I realized that it is not a good game to introduce programming to students. It will confuse the students even more. Students will only learn basic arithmetic i.e. calculating x+1 and x-1. Even the if statement doesn't teach much. I can't see the flow of evaluating the conditional if condition as true or false. In both ways, the players get to proceed in the game without thinking much about the difference of evaluating the 'if statement' as true or false. There are only a few while statements and if the players crosses over these while statements, he or she will not learn anything about the while statement. The learning of

		programming concepts comes from the game instructions and not from the game itself. L4: C-Jump might not be suitable for students at the university. We got to ensure that they have more understanding about the programming. Students may end up playing the game without going through certain syntax at all. For instance, someone who never got to play a while loop. Students may be playing this game several times but coincidently never got to learn 'while' and therefore the lecturer cannot penalized him for not learning it because he simple never got the chance. So, it is important to make sure that everybody learns everything in a standard way. So, this board game might not be so appropriate. Nevertheless, this game can be played by students before they take the programming course. It cannot be part of the programming course or syllabus. It could be a pre-introductory programming to students. Students will be able to expose themselves about how a computer may behave, what it means by branching, what it means by looping, and what does a particular statement mean. So, C-Jump is useful in giving the new students a good start on programming before they actually attend the formal lectures. L5: I don't recommend it because C-Jump is enforcing coding.
4	What are your comments about the learning content (programming statements and the syntax) provided in the C-Jump?	L1: C-Jump is specifically focuses on the C language. The game covers C programming syntax. C-Jump will help students to understand the logic (flow of the statements) i.e. if(x==1) and operators i.e. x++. x++ could be a trivial arithmetic statement but someone who has zero knowledge on programming will take time to digest it. L2: The statements are simple and would suite the beginners. There lots of arithmetic statements. L3: C-Jump consist of very basic programming statement i.e. int declaration, if else, switch and while. C-Jump should introduce functions to its players. Majority of the statements covered in C-Jump are basic arithmetic and many statements only require players to move down accordingly to the number rolled on the dice. L4: It comprises of programming syntaxes. Players are required to identify what the syntax means. I.e. 'x--' is a C program syntax and not a normal arithmetic symbol. By learning this syntax, the players will be able to know what is meant by --. It will actually help players to learn some syntax. Players will never see these statements like x++ anywhere else except in the C program. So, players will be exposed to this ++ symbol even before he or she go into the programming class. Anyhow, players will only learn about the syntax and not about any concept. There is some amount of learning for a new comer. The purpose of this game is to introduce basic programming concepts. It is quite comprehensive for a game that teaches basic programming concepts. It covers variables, arithmetic

		operations, the three control structures (sequence, selection and repetition) and jump statements (goto, continue and break). If students who really play and explore it with proper understanding of the correct rules, then they should be able to learn something. Programming is about both arithmetic and logic, and so it is C-Jump. L5: It is very primitive and very simple to learn. Each time you throw the dice, it will change the variable and that is confusing and does not really happen in the context of programming. Association of throwing the dice is an assignment statement to assign variable x, and there is only one x. Once you declared an initialized variable, change of value of that variable will be depending on the course of the program. So, once the variable is initialized to a value, statements like x+1and x-- should change the value and not the dice.
5	What are your suggestions to improve the C-Jump board game so it could be used as tool to introduce programming to tertiary level students?	L1: This game should first be tested with 2 different groups of students, i.e. students having some programming knowledge and students without any programming knowledge. After that, see if they have mastered the logics or the syntax. Maybe we should add different syntax in the game. For instance, adding nested if statement, for loop, do while, etc. L2: Need to rearrange the statements on the board to ensure every player gets a chance to evaluate some conditional statements. Add more statements i.e. assignment statements, nested statements, etc. Having a computerized board game that could be monitored and regulated. If students play it wrongly, then it should hint the students by giving a beep sound. If the students do not follow the correct flow of the program, then we could inform them. The game should be guided to ensure the students get to understand the concepts correctly at the first time itself or not it would be staying with them forever. L3: First we need to improve the instructions. For instance, it is unclear of what the colors of the squares should mean. I would suggest rearranging the game instructions by separating them into two parts, how to play the game and what the syntax means. Then we need to reduce the arithmetic statements in the game and replace them with other types statements. Add function calls in the game. Have 1 small main function that calls other functions that returns a value. L4: Not a bad game but it is just that the rules are not clearly stated, maybe they should state with more examples. Anyhow the virtual tour which is downloadable from the website is helpful in understanding how to play the game. If we were to use this game for university tertiary level, then your assessment cannot be based on exams. For tertiary level education, we normally use exams to assess if the students have learned something. When we have an exam, it is difficult to make sure that the student has learned everything using a game. For instance, how do you ensure that the student have really learned about the ‘default’ statement? There is a possibility of

		not getting to place the skier on the 'default' statement. This may be possible by having a computerized board game where somehow every player is maybe forced to go into at least 1 sequence structure, 1 selection structure and 1 repetition structure. The game should be able to keep track of the number of times each player has entered a loop, if else structure, etc. Having done this, you can somehow program it in such a way that the player has experienced all the basic aspects of programming while playing the game. L5: We have to think of some other game which will be able to teach programming instead of coding.

ANALYSIS

4.1 Students and the C-Jump board game

4.1.1 Students' views on the ability to play the game

Figure 11 shows students responses towards the ability to play the C-Jump board game. The mean score of this statement is 3.65 ± 1.19. Clearly, the majority of the students (78%) agree that C-Jump is a playable game. From the observation, some of the students immediately started to play the game and some were repeatedly reading the instructions to learn how to play. Anyhow, the majority of the groups could play the game although most of them were not playing correctly. Sixteen per cent of the students claimed that they neither disagree nor agree that the game is playable. Only 6% disagree that it is playable. Perhaps, these were the students who looked very puzzled during the C-Jump session and patiently waited for other participants to start the game. These finding indicates that generally students would be able to play the game even if they were to play it incorrectly.

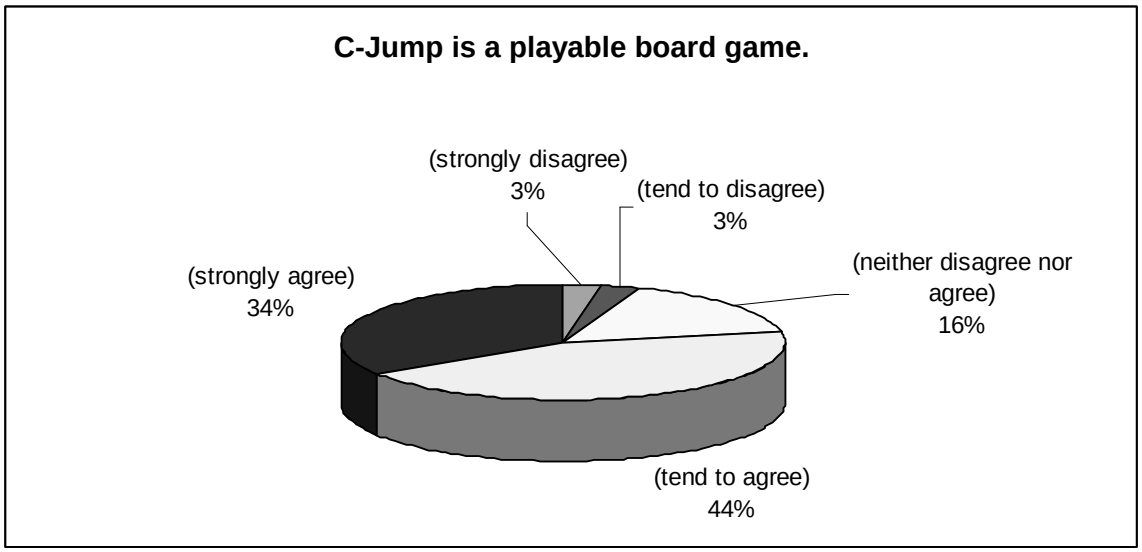

Figure 11: Playability

4.1.2 Students' views on the clarity of the game rules

Figure 12 shows students response towards the clarity of the game rules. The mean score of this statement is 2.75 ± 1.22. About half of the students (48%) agree that the rules were clear enough for them to understand how C-Jump should be played. Thirty per cent claim that the game rules were ambiguous. One of the students commented that:

> *The game rules are confusing but it is good for students to start learning computer programming. C-Jump board game must have a complete set of rules so that it can be played by both children and adults.*
>
> Student

Some students (22%) neither disagree nor agree that the rules were clear. Perhaps, these students find that are some parts of the game rules are clear and others need to be improved. From the observation, it was clear that some of the instructions are rather confusing. For instance, almost all the students were not sure about how to proceed with the game when their skiers were stopped at the "return x" statement. The instruction were misleading due to the students interpreted it differently. Some of them restarted the game and some just finished the game.

Despite that, most of the comments noted from the recorded tape are negative comments related to the games instruction. Although majority of the students find that the game rules are clear, this finding reveals that some of the instructions are rather confusing and should be revised.

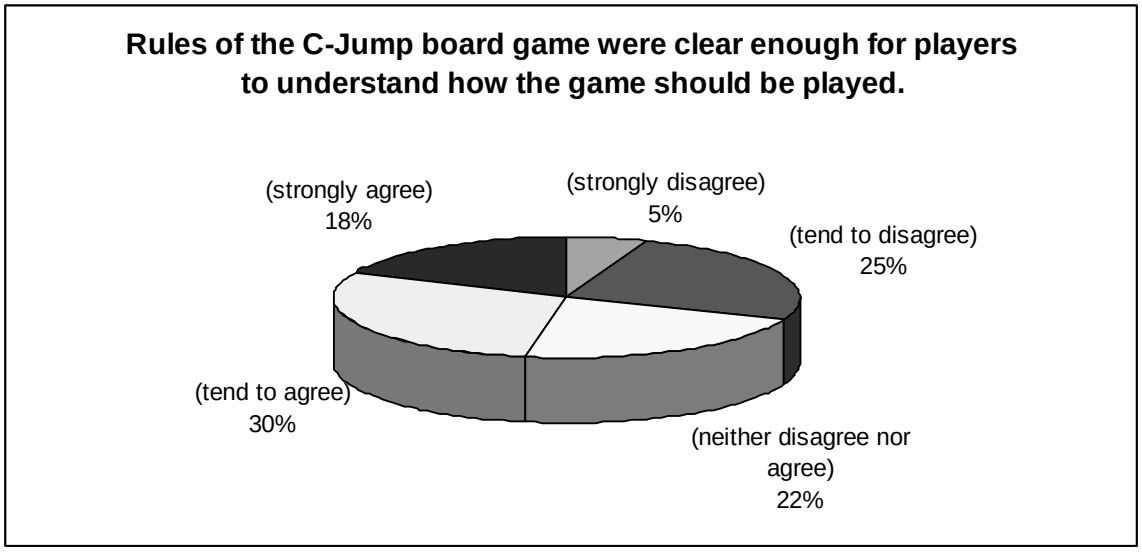

Figure 12: Clarity of rules

4.1.3 Students' views on the learning aspects of the game

Figure 13 depicts whether the students have learned the basic programming concepts and syntaxes from the C-Jump. The mean score of this statement is 3.65 ± 0.79. Majority of the students (74%) agree that they have either directly or indirectly learnt the basics of programming from the game.

Very few students (5%) claimed that they didn't gain any basic programming knowledge from the game. Perhaps, these students were just playing to win in the game or the game covered only very trivial programming content. Twenty one per cent of the students neither disagree nor agree that they learnt anything from the game. At the general comment section of the questionnaire, a student highlighted that the C-Jump should be adjusted to suit the university students.

> *C-Jump should be made more comprehensive and complicated if it will be used by undergraduate students.*
>
> Student

From field notes taken, one of the students commented that:

> *We only learn basic arithmetic from this game.*
>
> Student

This finding provides that C-Jump is able to stimulate some basic programming knowledge while the students play the game. Anyhow, it is apparent that it is quite primitive and should be enhanced further to match the expectation of undergraduate students.

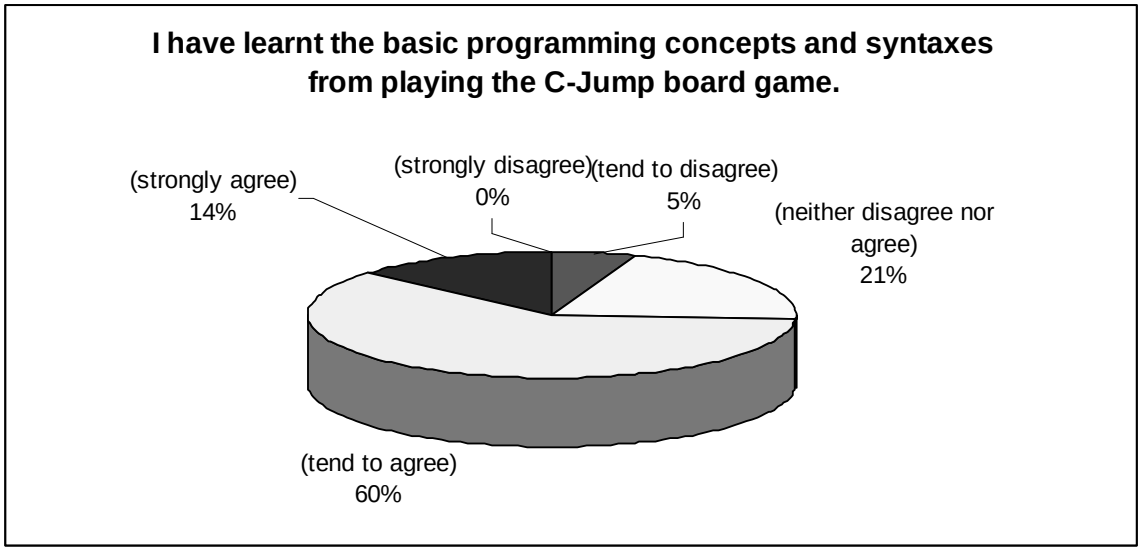

Figure 13: Educational value

4.1.4 Students' views on the fun factor of the game

Figure 14 depicts whether the C-Jump was interesting and fun to the students. The mean score of this statement is 3.8 ± 0.93. Seventy five per cent of the students found the game interesting and was fun to play with. During the observation, it was noted that a number of the students were excited throughout the game session.

At the general comment section, one of the students commented that the game is fun only if the players are able to understand it.

At first, I couldn't understand the game and it appeared boring to me but later when I understood it, its fun! We should play these games more in the class routines.

Student

Only 5% of the students disagree that it was interesting and fun to them and 20% neither disagree nor agree with the statement. The results of the finding indicates that C-Jump is generally an interesting and fun game to its players.

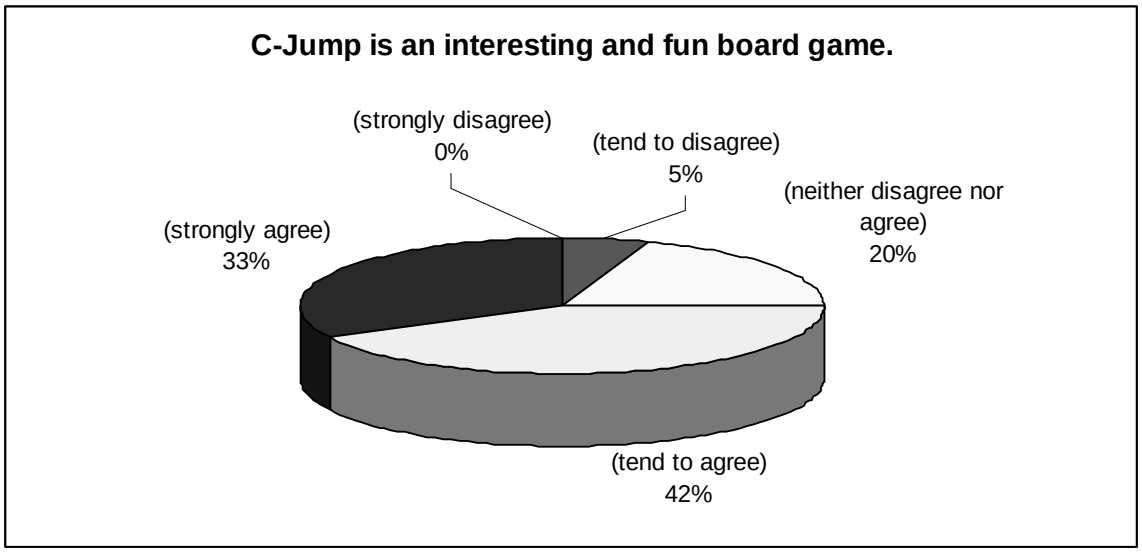

Figure 14: Fun factor

4.1.5 Students' views on the aesthetic appearance of the game

Figure 15 represents students' opinion about the aesthetic appearance of the game. The mean score of this statement is 3.55 ± 1.02. It is clear that majority of the students (72%) found the game attractive and appealing. Only a small number of the students (7%) found it otherwise. Twenty one per cent of them neither disagree nor agree that it was attractive and appealing.

Even though C-Jump is targeted for younger players (age 11 and above), results of the survey reports that it is attractive and appealing to undergraduate students (adults) as well.

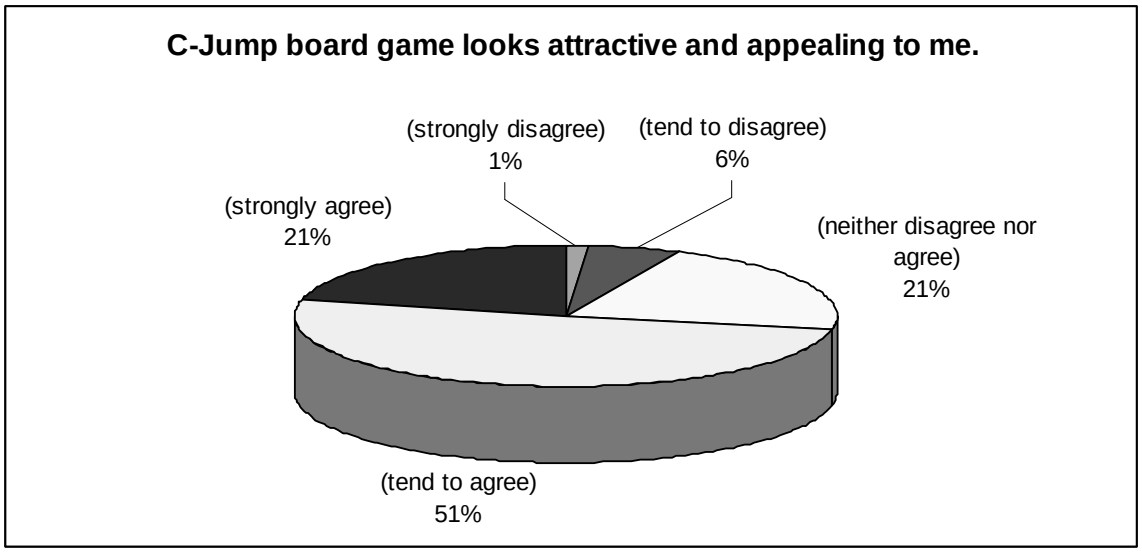

Figure 15: Aesthetic appearance

The following chart (Figure 16) provides that mean response for each of the category of evaluation by students. Overall, the responses were quite positive. The fun factor category were rated the highest, with a mean score of 3.8 ± 0.93. The lowest-scoring category was clarity of rules, with a mean square of 2.75 ± 1.22. These results generally indicate the best aspect of the game is that it is fun and interesting to play with and the worst aspect of the game are the game rules which are rather confusing or incomplete.

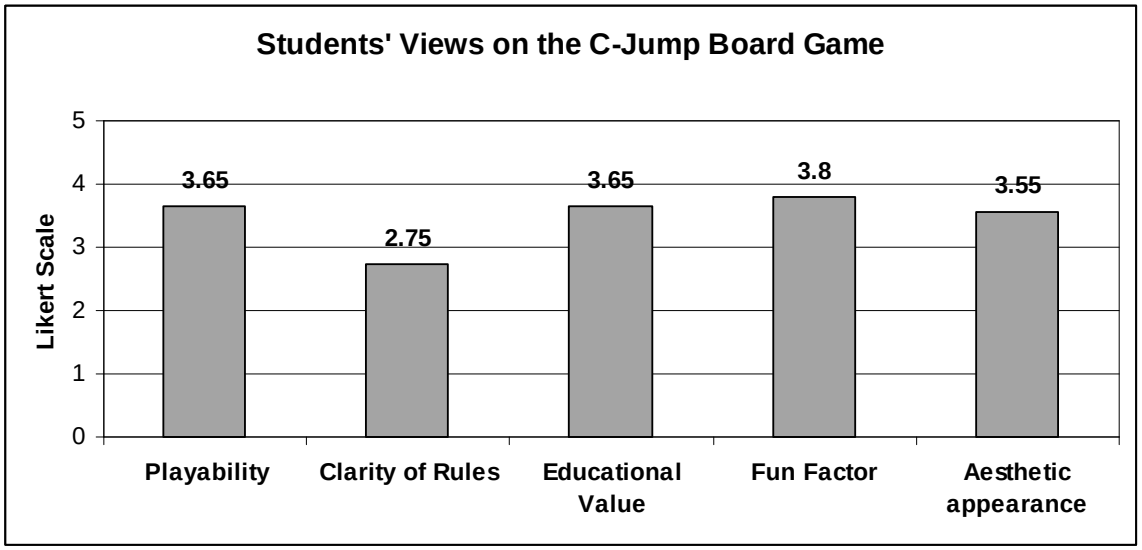

Figure 16: Overall response of the five evaluation measures

4.2 Lecturers and the C-Jump board game

4.2.1 Lecturers' remarks about common problems experienced by students learning programming

The lecturers generally agreed that the students face several challenges in learning their first programming course. Interestingly, they have different opinions on this issue.

Some of the lecturers explained that the students face difficulty in visualizing what was taught in the class. The students are expected to imagine something very abstract, which they have never experienced before.

> *They cannot visualize how the syntax works. For instance, how a loop executes in a program.*
>
> Lecturer

Most of the lecturers claimed that the students do not know how to do *problem solving*. In other words, they can't breakdown a given problem into smaller parts. The students are carried away with notations, semicolons, quotations and punctuations.

> *Students don't have a good knowledge of how to solve the problems.*
>
> Lecturer

Lecturers' comments also indicated that many students face difficulties with the syntax or coding. They have problems in memorizing and applying the syntax. One of the lecturer reported that many students merely read and memorize the notes without doing much practice on writing programs. As a result, they can't apply their knowledge when they are given some problems.

One lecturer generalized that the students are confused between programming (problem solving) and syntax (coding). He strongly suggests that the lecturers should first teach

and focus on programming instead of coding. It seems that, by doing so, students will not be lost into syntactical notation like semicolon, braces and quotes.

4.2.2 Lecturers' views about the use of games in teaching programming

Most of the lecturers generally agreed with the idea of using games as an alternative tool to teach programming. Many of them claimed that the game is an effective tool to attract the attention of many students to learn and gain something.

Majority of the lecturers feel that the game is just a supplementary to the lectures even though games can be used to make learning interesting. Formal lectures should still be the main medium of instruction and the educational games should be played outside of these formal lectures hours. A lecturer commented that it is a good tool to attract attention of students who dislike programming.

> *It can probably get the interest of students who dislike programming to entice them in learning.*
>
> Lecturer

One lecturer has stated that we can't formalize the idea of using games to teach something. This is due to the fact the some students may not like games and may still prefer the formal education over learning while playing.

Another lecturer responded that he hasn't seen any good game that can be used to teach programming to students. Moreover, he totally disagrees with the idea of using games in education. He never believed that it would be an effective approach to enforce learning.

4.2.3 Lecturers' comments about using C-Jump to teach basic computer programming

The lecturers had different opinions about the use of C-Jump as a tool to aid learning programming at tertiary level students. Some of the lecturers recommend the use of C-Jump at the university because it is a game, and games can attract the interest of students to learn something. Anyhow, they propose it to be played by students before the formal lectures. Perhaps, it can be used during the first week of the lecture, i.e. when students are being introduced to programming. Students playing the C-Jump before attending the formal lectures will be able to expose themselves about how a computer behaves, branching, looping, and what certain programming statements mean.

Some of the lecturers encourage the use of C-Jump upon believing that if the students were to play it quite frequently, they will somehow learn some programming statements. For instance, they will learn how a switch statement works. One lecturer somewhat agreed it to be used at tertiary level but highlighted that the game session must be guided by someone. This lecturer is concerned that the students may not be playing it correctly or they may be interpreting the statement wrongly. The students may end up confusing themselves even more if they are not assisted in playing C-Jump.

Other lecturers discourage C-Jump for undergraduate students. They claim that it will confuse the students. Moreover, it doesn't teach much. It mainly covers basic arithmetic like x+1 and x-1. One lecturer explained that we can't use C-Jump for undergraduate students because we can't assess them due to the possibility of not getting to learn all statements in the game. There are only a few different statements. For instance, there are very few "while" statements and if the player crosses over these statements, he or she will not learn anything about the "while" statement. One lecturer feels that the students would be learning the programming statements by reading it from the game instructions which maybe equivalent to reading from notes.

The learning of programming concepts comes from the game instructions and not from the game itself.

Lecturer

Another lecturer objects the use of C-Jump for tertiary level students because it enforces coding rather than programming. The game focuses more on the syntax instead of problem solving.

4.2.4 Lecturers' comments about learning content covered in the C-Jump

Majority of the lecturers claimed that the C-Jump mainly covers basic arithmetic statements. Anyhow, one lecturer did stress that the "x++" might be trivial statement, but it is something very unique and would be a worthwhile knowledge for someone who has zero knowledge on programming.

Some of the lecturers reported that the game covers enough for a game that teaches basic programming statements. It covers variables, arithmetic operations, the three control structures (sequence, selection and repetition) and jump statements (goto, continue and break). The lecturer added that if students really play and explore the game with proper understanding of the game rules, they should be able to learn something.

One of the lecturers commented that C-Jump specifically focuses on C programming language. Players playing this game will only learn the C language. Another lecturer claimed that the players of this game will only get to learn about the syntax and not about any programming concept. He also added that there is surely some amount of learning for a new comer.

Another lecturer claimed that C-Jump is very primitive and very simple to learn. Interestingly this lecturer has given an insight into how this game is contradicting with real programs. Each time the player throws the dice, it will change the contents of variable "x" which actually does not happen in the context of programming. In C-Jump,

association of throwing the dice is an assignment statement to assign variable “x”. In real programs, once you have initialized a declared variable, change of value of that variable will be depend on the course of the program. So, once the variable is initialized to a value, statements like x+1and x-- should change the value and not the dice.

4.2.5 Lecturers’ views on improving the C-Jump

The most common suggestion for improving the game was to improve the game instructions. The instructions need to be rephrased so that it is less ambiguous and the players could easily understand what they are supposed to do. The instructions also need to be reorganized so that the basic rules of the game are stated first and only then to state what each statement means and how the player should proceed.

Lecturers’ comments also included suggestion to replace the many basic arithmetic statements with other distinctive programming statements. For instance, there should be other statements like the nested if, for loop and do while. One the lecturer suggested that the game should include functions as well. They could be several other functions, each containing a different control structure (sequence, selection and repetition) which should be called by the main function. The new functions should return value to the main function.

Many of the lecturers suggested that the physical C-Jump board game should be transformed to a computerized board game. One lecturer explained that by having the game automated, we will able to keep track of the players’ progress in the game. It seems that currently the players can easily disobey the game rules and it is difficult to monitor every player’s moves on a physical board game. There should some beeping sound if the player has moved his or her skier incorrectly. Another lecturer did recommend that the computerized board game should be intelligent to ensure that every player gets to place the skier on a control statement on the board. This is to reduce the possibility of not getting a chance to learn the “if”, “switch” and the “while”. The lecture explained that if the player is almost finishing the game but has not placed the skier on any of the control

structures, then the player should be forced to enter into the final “while” statement on the board.

The last interviewed lecturer totally disagreed with the idea of using this game for undergraduate students and therefore refused to suggest any improvements.

ISSUES, RECOMMENDATIONS AND SUMMARY

5.1 Issues Emerging from the Study

The finding provides that most of the students do not play the game correctly. They do not play according to the rules specified in the game instructions. Many of the students do not read the rules carefully and are very impatient to start the game directly. They seem to apply the rules of the classic Snakes and Ladders board game in C-Jump.

Majority of the students do not understand how to play the game. They try various ways to play the game and restart all over several times. Quite often, they keep on replacing the "x" of the subsequent statement with the addition of the earlier statements. In other words, they seem to roll the dice only once.

Most of the students do not understand how they should proceed the game when their skier is located on the "return x" statement. The game instruction provides the following:

> *" "return" statement returns skiers to the ski base. Regardless of a number rolled, the skier moves past the FINISH line. "*
>
> C-Jump

The first sentence of the above instruction is interpreted as the need to restart all over. The second sentence of the instruction explains that the player finishes the game. The students are confused whether they should start all over or they finish the game. The students seem to wonder where is the "ski base" actually.

From the observation, it is apparent that the game finishes very fast. Perhaps, it is because every group played C-Jump by using only 1 skier per player. Not even a single group chose to use or attempted to use both skiers to represent the players.

Many students seem to evaluate the “if” statements, “switch” statement and “while” statements halfway in their moves. The rule of not evaluating these statements in the middle of the move is not stated in the instructions given in the game box but is explained in the animated tutorial which can be downloaded from the website.

C-Jump requires the players to replace the every ‘x’ with the number shown on the rolled die. There is no assignment statement on the game board but the players are required to assign a new value to the “x”. This is different from real programs, where once the variable is given a value, the value remains unchanged unless there is an assignment statement causing that action or there is some unary operator (++ or --) which will increase or decrease the original value by 1. Hence, the students may be confused when they see the real programs in the future.

Students will not learn every distinctive statement available on the game board. They may not get the chance to place their skiers on every single statement. Some of the statements could be crossed by the player counting number steps rolled on the dice. As a result, they student will not get to learn the “switch” statement if he or she coincidently did not get place the skier on the “switch”.

C-Jump comprises of too many arithmetic statements. Nearly 65% of the statements presented on the game board are basic arithmetic statements involving operators like plus (+), minus (-), multiply (*), divide (/), increment (++) and decrement (--). Therefore, most of the time, the player will be engaged in performing arithmetic operations instead learning other types of statements. Other than arithmetic operations, the game consists of several “if” statements, a few “while” statements, a “switch” statement and few other statements.

Statements which are not arithmetic or a control structure (if, switch and while), merely requires the players to move downhill accordingly to the number rolled on the dice. When the player places his or her skier on the following statements (see Table 10), the player has nothing much to think or to learn but simply move on by counting the number

steps rolled on the die. In other words, the students don't learn anything from these statements because they are not required to do anything special.

Table 10: List of statements

Programming Statement	Example of Statement
variable declaration	int x
function declaration	int main()
open curly brace	{
close curly brace	}
label	jump:
case	case 1:
else	
break	
default	

Most of the students do not attempt to understand what the statements on the game board means but are only interested to know how they should proceed from the current statement. The instructions seem to explain 2 things: first, the meaning or purpose of the statement and then how the player should move from there. It seems that majority of the students pay less attention to understand the statements in concern as they just want to know how to play the game. So, students just want to win the game and not learn from the game. This indicates that the students will not learn much if their motive is just to win the game. Winning the game does not necessarily mean that the students have gained more knowledge than the other players in the game.

5.2 Main Recommendations

Results of the study indicates that certain aspects of the C-Jump board game should be enhanced further. Below are some recommendations for consideration that can be used for the future development of the game particularly as an effective tool for teaching computer programming to tertiary level students.

5.2.1 Improvements to the game instructions

Most respondents mainly suggested the game instructions to be revised. First, the instructions should be made to be unambiguous. By reading the whole instruction text once, the players should easily understand each rule of the game. There shouldn't be any words which would make the players misconstrue its meaning. The instructions should be restructured in a manner that will help players to easily understand the game. First, general rules of the game should be informed. For instance, it is a general rule that players need to roll their dice again if their skier has stopped at the orange square. This sort of basic rule must be explained much earlier so that players get to have an overview of how the game should be played. Only then should there be rules concerning cases of placing the skier on different programming statements.

Each rule of the game should be clearly defined in the instruction text. The finding clearly provides that almost every group of students evaluated the orange "if", "switch" and "while" statement in the middle of their moves. It seems that this rule was missed out from the instruction text.

Some of the respondents suggested the important rules of the game to be placed on the game board itself. The finding reveals that that many new players of the game read the game instructions quite frequently. There are about 24 different types of statements on the game and each of the statement may require a different move. The suggestion is to use a small portion of the large game board to provide information about the

programming statements and how to precede the skier from these statements. Perhaps, this will reduce the time involved in reading the rules from a separate paper.

5.2.2 Modifications to the programming content

The second most prominent recommendation is to reduce the number of basic arithmetic statements on the game board. Many respondents feel that these squares or space on the game should be replaced with other distinctive programming statements like the nested if and for loop, do while loop. Players should be able to learn a lot from this game.

Some of respondents did highlight the idea of including functions in the game. Perhaps, the game board could be divided into a few functions. Each function should do an important task like finding the average value. There should be several functions calls and return values in the game. This will definitely help players especially the undergraduate students to visualize how functions work in real programs. They will also be able to understand the concepts of returning values from other functions.

5.2.3 Transformation to a computerized board game

Some of the respondents strongly suggested the transformation from the physical board game into a computerized board game. By having the board game computerized, various intelligent features could possibly be added.

There could be 2 modes of play: having the skiers to be placed at the designated statements automatically and having the players to position the skiers on statements by themselves. The first mode is to allow players to observe how the game should be played. The player will be required to click at their respective buttons alternately. This will cause their respective skiers to be repositioned at a designated statement depending on the rules of the earlier statement.

Once the players have mastered the rules and the movements, the players can then choose to play the second mode, in which they will be placing their skier on the square containing the statement by them. Every movement of the skier will be closely monitored. For instance, a beep sound will be triggered for any false positioning of the skiers. In other words, the players will be guided to play the game correctly. This is essential because it will help to ensure that the players learn to move the skier as expected or explained in the instruction. Perhaps, this will help them to visualize how certain programming statements work in real programs.

It is an issue that many of the important statements in the game could be coincidently crossed over by the players. Hence, they might not learn anything about these missed statements. By having the board game automated, the players can possibly be forced to enter into “if”, “switch” or any possible statement. This is because the game board can be programmed to keep track of the number of statements experienced by the players. For example, the board game can be regulated to keep track of the number of “if” statements experienced by a particular player in a single game session. Having that tracked, if the player is almost ending the game but he or she did not place the skier on an “if” statement, then perhaps the board game can be programmed to force the player to experience the last “if” statement of the game.

5.3 Summary

The C-Jump board game was studied in great detail to determine at the possibility of using it as useful tool to aid undergraduate students to learn programming. C-Jump is generally a good game to introduce basic programming statements to its players. It is actually target for players of age 11 and above but it was interesting to determine if it could benefit undergraduate students as well.

Several techniques were employed to evaluate the game. Mainly, data was gathered from 2 categories of respondents, i.e. the undergraduate students and the lecturers. Inputs and comments from both categories were analyzed accordingly.

Generally, the C-Jump was well appreciated by the students. Though the game covered only basic statements, the finding indicates that there is surely some amount of programming knowledge to gain from this game. However, several issues were encountered while the game sessions were being closely observed. These issues were elaborated in length and certain recommendations to possibly overcome some to those issues were highlighted. It is great idea of attracting student to learn basic programming by using a game. Perhaps, the adjusted version of this game would greatly benefit many novices at tertiary level.

REFERENCES

A History of Board Games. (2003). Astral Castle.
< http://www.ccgs.com/games/ (15/11/2006) >

Austin, R. G. (1940). “Greek Board Games”, Elliott Avedon Museum and Archive of Games University of Waterloo, Canada.
< http://www.gamesmuseum.uwaterloo.ca/Archives/Austin/index.html (29/11/2006) >

Barr, A.& Kessler, S. (1996). “Good Programmers Are Hard To Find: An Alternative Perspective on The Immigration Of Engineers”, Press Briefing, Stanford University Computer Industry Project.
< http://www.stanford.edu/group/scip/avsgt/immigration1096.pdf (24/11/2006) >

Block G. (2006). “Mitsubishi R&D's WarCraft III Panel”, ign.com.
< http://gear.ign.com/articles/698/698241p1.html (26/2/2007) >

BoardGameGeek , “Why Board Games are Better than Video Games”
< http://www.boardgamegeek.com/geeklist/13879 (5/5/06)>

BoardGameGeek (2005), Game: c-jump Computer Programming Board Game, Forum.
< http://www.boardgamegeek.com/thread/79632 (10/1/2007) >

Board Fun, (1998). “Types of Board Games”, Drawing Board.
< http://library.thinkquest.org/4377/drawingboard.html (14/1/2007) >

Board Games of the Future (2006). “Computer Engineers Bring a Bit of Virtual Reality to a Holiday Tradition”, Science Daily.
< http://www.sciencedaily.com/videos/2006-11-11/ (26/2/2007) >

CanBooks. (2003). “Ancient Board Games and the Nabataeans.”
< http://nabataea.net/games3.html (29/11/2006) >

Cohn, D. (2005). “C'mon Kids, Let's Play Programmer”, Wired News.
< http://www.wired.com/news/technology/0,68872-0.html (24/11/2006) >

Deanh. (2005). “C-Jump: Computer programming board game”, Make: technology on your time
< http://www.makezine.com/blog/archive/2005/09/cjump_computer.html (26/2/2007)>

de Boer, C. J. & Lamers, M.,H. (2004). “Electronic Augmentation of Traditional Board Games”, Leiden Institute for Advanced Computer Science, Leiden University, Netherlands.
< http://www.clim.nl/personal/docs/SP1-DeBoer-Clim.pdf. (10/1/2007) >

Dragontamer. (2005). “Talk:AP Computer Science”
< http://en.wikibooks.org/wiki/Talk:AP_Computer_Science (24/11/2006) >

Entertaible. (2006). “LCD-Based Board Gaming from Philips”, Gizmodo.
< http://gizmodo.com/gadgets/ces/entertaible-lcdbased-board-gaming-from-philips-146788.php (26/2/2007) >
History of Sports and Games. History World.
< http://www.historyworld.net/wrldhis/PlainTextHistories.asp?historyid=ac02 (29/10/2006) >

Johnson, M. A. H., (2001) “Computer and Video Games Are "Not So Bad" But Books, Board Games, Activity Equipment Are Better”, News, Virginia Cooperation Extension.
< http://www.ext.vt.edu/news/releases/121701/games.html (5/10/2006)>

JWZ. (2005). “C-Jump: Computer Programming Board Game”
<http://jwz.livejournal.com/572429.html (20/3/06)>

Kholodov, I. (2005). C-Jump.
< http://www.c-jump.com (16/9/2006) >

Learn to Love Board Games Again. (2005). “Learn to Love Board Games Again: 100+ Ways to Rejuvenate the Games You Already Own”, Yehuda.
< http://jergames.blogspot.com/2006/10/learn-to-love-board-games-again100.html (27/2/2007) >

Morrison, P. (2005). “Peter Morrison's Board Gaming Philosophy Overview”, Morrison Games.
< http://www.morrisongames.com/peter_morrison's_board_gaming_philosophy_overview.htm (10/1/2007) >

Saari, M., (2004), “Simulation & Computer Integration in Board Games”, The Games Journal.
< http://www.thegamesjournal.com/articles/ComputerIntegration.shtml (5/10/2006) >

Smith, S. E. (2005). “Clickers, C-Jump”, Educause Connect.
< http://connect.educause.edu/taxonomy/term/825,826 (10/1/2007) >

The future of board games (2006). TomSoft.
< http://blog.landspurg.net/the-future-of-board-gaming (26/2/2007) >

The History of Board Games. (2002). Essortment.
< http://w.v.essortment.com/historyofboard_rjyw.htm (15/11/2006) >

Wikipedia contributors. (2006). “Board Game”, Wikipedia, The Free Encyclopedia.
< http://en.wikipedia.org/w/index.php?title=Board_game&oldid=90538486 (27/11/2006) >

APPENDIX: RULES TO PLAY THE C-JUMP BOARD GAME

HOW TO PLAY

Introduction
Discover the fundamentals of computer programming by plying a game!

c-jump is a fun family game, benefiting learners of programming languages, such as C, C++ and Java.

By moving around the board, entering loops, branching under conditional statements, the players gain physical experience of a complete game. Understanding of the internal action of computer is essential to understanding what software is. Static programs causes dynamic process in the computer. By playing the game, players see this process as a physical and special motion.

Players
2 to 4 players.

Age
Age 11+.

Equipment
One game board, one die, and sets of colored pawns representing skiers and snowboarders for each player.

Object Of The Game
First player to move all skiers past the FINISH line is the winner.

Setup
Skiers and snowboarders line up at the START location and race along the ski trails, according to each player's roll of die and board rules.

Spaces on the board are shown as squares. Each square has a statement of a rule, borrowed from programming language. Semicolons ";" separate rules from each other.

`int x;`	Keyword "int" creates integer variable "x". in the game, "x" represents the number rolled on the die. For example, if player rolls 5, then x becomes equal to 5. From this location, skiers move downhill accordingly to the number rolled on the die.
`int main( )`	"Main" is a name of the blue ski trail on the board. All computer programs have function named "main". Functions define computer operations. The skier can move downhill number of steps rolled on the die.
`{`	Opening brace "{" indicates beginning of a ski trail. Closing brace "}" ends the trail. The braces require no special calculation, and can be counted as free landing space.
`}`	

Playing the Game
Player rolls the die and moves one of his/her skiers, counting off the number of squares. The game can be played with one or more skiers of the same color on the board, players may choose any of their skiers to move.

Before the move, if skier starts at a space with an arithmetic statement, players should calculate the number of steps by replacing "x" with the number rolled on the die. For example,

x+2;

Means "add 2 to x". The player must replace "x" with the number rolled on the die and add 2. if the player rolls 5, then number of steps becomes 7: 2 + 5 = 7.

Same rule applies to other statements with arithmetic expressions:

"6-x;" means "Subtract x from 6".
"2*x;" means "2 times x".
"x+x;" means "x plus x".

x/x

means "x divided by x". a number divided by itself equal one. Therefore, the player always gets to move one space from this location.

x++

means "increment x by one". The player should add one to the number rolled on the die. For example, if the number rolled is 4, the resulting number of steps is 5: 4 + 1 = 5.

x--;

means "decrement x by one". The player should subtract one from the number rolled on the die. If the number rolled is one, it becomes 0: 1 - 1 = 0. If the player rolls 1, the skier cannot move on that turn.

if (x==1)

means “if x is equal to one”. A double equal sign “==” compares two numbers for equality.

The condition “(x == 1)” is true when the number rolled on the die equals one. In all other cases this condition is false.

When this condition is true, the skier enter orange ski trail on the right side of the “if”. After entering the “if” pathway, the player is awarded a free roll and can only move the same skier, when playing with more than one piece per player.

When this condition is false, the skier must continue downhill, following the blue trail.

Similar rules apply to all other “if” statements on the board:

“if(x > 1)” means “if x is greater than one,” which is true for 2, 3, 4, 5, 6 and false for 1.

“if(x < 5)” means “if x is less than five,” which is true for 2, 3, 4 and false for 5 and 6.

else

The “else” keyword indicates a pathway that should be followed when condition of the previous “if” statement was false. From this location, a skier moves accordingly to the number rolled on the die.

while (x < 4)

means “while x is less than 4”. Keyword “while” test the condition the same way “if” does. An orange arrow at the end of the “while” pathway points back to the “while” space, allowing skier to make a loop.

When condition “x < 4” is true, the skier enters the “while” pathway, counting off the number of steps accordingly to the number rolled on the die. The player is awarded one free roll and should move the same skier again.

When the condition is false, the skier must continue downhill along the blue trail moves.

while (x > 0)

The same rule applies to other “while” statements on the board. For example, “while (x > 0)” means “while x is greater than zero”. Since any number on the die is greater than zero, this pathway must always be entered by skiers starting at this location.

When exiting from any loop, skiers should continue downhill, following the blue ski trail.

goto jump;

jump

Keyword “goto” points skiers to the square labeled “jump:”. “Jump” is a label that gives a name to a particular location on the board. Labels allow “goto” statements to point to various places in a computer program. From both of these locations, a skier moves accordingly to the number rolled on the die.

switch(x) {

case 1:

case 2:

case 3:

default:

Starting at the “switch” statement location, skiers move to one of its labels. If number rolled on the die is 1, 2, or 3, the skier should move to the square labeled “case 1:”, “case 2:” and “case 3:” respectively. The player is award one free roll and moves the same skier again. If the player rolls 4, 5, or 6, the skier follows the “default:” pathway.

break;

Keyword “break” creates an exit from a loop or a “switch”. From this location, a skier moves the number of spaces rolled on the die.

continue;

Keyword “continue” forces the skier back to “while”. The skier moves accordingly to the number rolled on the dice. If there is more than one step in the move, the skier exits the loop and follows the blue trail.

return x; }

“return” statement returns skiers to the ski base. Regardless of a number rolled, the skier moves past the FINISH line.

Finishing the Game

To complete their goal, skiers must cross the FINISH line by exact number of steps, counting FINISH location as a square. If the number of steps is too big, the player must choose another skier, or skips the turn.