Code Ninja: A Game for teaching Basic Programming Concepts using Unity 3D

By

Janine Maasin

John Pritz Belleza

COMPUTER STUDIES

ATENEO DE DAVAO UNIVERSITY

MARCH 2016

Code Ninja: A Game for teaching Basic Programming Concepts using Unity 3D

An Independent Study

Presented to

The Faculty of the Computer Studies Cluster

Ateneo de Davao University

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science in Information Technology

By

John Pritz G. Belleza

Janine M. Maasin

SCHOOL OF ARTS AND SCIENCES

ATENEO DE DAVAO UNIVERSITY

MARCH 2016

Table of Contents

ACKNOWLEDGMENTS

We would like to express our sincere gratitude to our adviser, Ma’am Michelle P. Banawan for the continuous support and encouragement. Her guidance and immense knowledge helped us in our research to successfully produce the final output of our thesis.

Besides our adviser, we would like to thank the rest of our thesis committee, our panelists: Mr. Antonio Bulao II and Ms. Katrina Cecilia L. Corro, for their insightful comments, suggestions and questions that challenges us, and for recognizing our efforts to complete this study. In particular, we are grateful to Mr. Bernie M. Jereza for enlightening us and who offered much advice and insight all through our work. To our classmates Colleene May Marie Escarlan, Bianca Sambrano, Perkeen Li and IT4A for supporting us, giving us courage and for helping us in our testing. And to Mr. Emmanuel Cabuloy and his students for giving us time and for allowing us to do the testing in his class CS 111.

Last but not the least, we would like to thank our parents, Mr. and Mrs. Belleza and Mr. And Mrs. Maasin for the love that they show to us and for supporting us spiritually and emotionally throughout our life. And to God for being our strength to hold on and to believe in him and to ourselves in times that we almost gave up.

Without these people, our research and thesis project wouldn't be possible without their support and help and they kindheartedly helped us in the preparation and completion of this study.

Code Ninja: A Game for teaching Basic Programming Concepts using Unity 3D

JOHN PRITZ G. BELLEZA, ATENEO DE DAVAO UNIVERSITY
JANINE MAASIN, ATENEO DE DAVAO UNIVERSITY

ABSTRACT

Game-Based Learning are games that for learning purposes yet are fun. Games have been a popular approach in teaching concepts to motivate students to learn. Since computer programming has been found to be a difficult subject to teach and to learn, games to help novice programmers better understand

programming is deemed to be useful. This project intends to contribute to the teaching and learning process of computer programming by the creation and development of a game that will help students understand the underlying concepts. Upon evaluation, the game received positive ratings by novice programmers / students in terms of their better understanding of computer programming concepts, and usability.

1. INTRODUCTION

1.1 Background of the Study

Most games today are purely based on entertainment. These games are a distraction, irrelevant, and time-wasting. According to Park (2014), it can cause young people to become violent, based on the scores of aggressive behavior and violent tendencies test it revealed that kids who play more hours of video games a week scored higher than those who play less. It would be beneficial if a game is educational or a game can help improve the player's knowledge. According to Malykhina (2014), video games are playing an increasing role in school curricula because teachers seek to teach core lessons like math, reading or even the computer programming in the best interest of the students. A game should have an impact on people's knowledge and skill base. An educational game is a useful learning tool. It can enhance skills needed for success in school. These games are both designed to help people to learn a certain lesson or to assist them in learning a skill while they play and have fun at the same time. Education should be fun and challenging to make students motivated to learn. The learning should

be inherent to the gameplay, and the gameplay should be inherent to the learning. The programming lessons that is taught to college students is essential and gives them an advantage if they are pursuing it or if the course they will choose is related to Computer Science. That is why the proponents wanted to motivate the students to learn the basics programming concepts. The proponents are proposing a computer game that is developed to teach college students programming concepts. This game will assist them on developing their skills on programming while having fun at the same time.

1.2 Problem Statement

Learning computer programming concepts is difficult for most students (Jenkins, 2002). Not only is it difficult to learn but it is also difficult to teach, and to address this the use of illustrative examples to teach and learn concepts is a pedagogical solution that is employed (Miliszewska, & Tan, 2007). This proposed research intends to develop a game that helps students learn programming concepts. Will the students be able to learn programming concepts with the help of a game that is designed for this purpose? Will the students be able to relate the game to the programming concepts? Will the students be able to learn and apply those concepts learned from the game?

1.3 Objectives

The main objective of this study is to create a game that teaches programming.

Specific Objectives are:

- To test whether the students learn from the created game
- To interpret the results based from the tests given to the students
- To find out whether the players had fun in playing the game

1.4 Significance of the Study

This significance of this study is to develop a game for novice programming students that would teach them some introductory programming concepts like the top-down flow of structured programming, control flow of execution, procedures or functions and the dynamics of repetition in loops. This game would teach them how to break down problems, have a visual representation of the solution, and have a better understanding of the underlying concepts of introductory programming. This study would also investigate if games are effective learning tools for novice programmers.

1.5 Scope and Limitations

This project is motivated on developing a game that focuses on developing the player's understanding of the basic programming concepts by giving them visual puzzles that gets more challenging in every level and illustrates the underlying concepts of the programming skill that is being introduced in the level. The game is intended for students in order for them to understand introductory programming concepts.

The game is developed using Unity 3d. The specific concepts/ topics are : basic flow of control, concepts regarding functions or procedures, and the concept of repetition in loops.

2. Review of Related Literature

2.1 Video Games and Learning: Teaching and Participatory Culture in the Digital Age

Kurt Squire addresses a question in the first chapter of his book Video Games and Learning: Teaching and Participatory Culture in the Digital Age (Squire, 2011) which is “can games actually be used to learn?”. In the chapter, Squire presents a few responses to questions such as, “Why study video games? Aren’t they a waste of time?”. He concluded that people learn academic content like names of people, historic places, damage calculation, and terminology through games regardless whether the game is designed for educational purposes or not. Popular video games like Pokemon, Dota, and League of

Legends, diehard fans of these franchises retain an in-depth knowledge about the worlds of these games so if a player can learn and absorb this type of deep information based on a fictitious world, one could argue that educational content could be delivered to players and absorbed in the same manner. He also said that games that makes players more engaged are the games that give players the ability to manage their goals and complete them in a timely fashion. Squire concludes that video games create participatory situations for players to learn and hone their skills. This is the exact atmosphere that any educator would want from a classroom.

2.2 Light-bot

According to Gouws et.al. (2013), computational thinking is an important skillset that should be acquired by students but it is not really taught through lectures but learned through experience or through deep thinking. Light-bot an educational platform puzzle game which aims to complete certain tasks by issuing the sequence of commands.

Learning programming concepts will transfer well into actual coding. In Light-bot completing one level unlocks the next, but the challenge escalates quickly. Players who continue to play and accept the challenges of the game will develop a pretty deep understanding of programming concepts. Gouws'

approach is to make a computational framework be used as a planning and evaluative tool.

2.3 Cargo-bot

In Cargo-bot, the learning approach is to make players learn recursion by making them learn the step-by-step computational logic that teaches them to tackle a bigger problem by breaking it into pieces and solving it one at a time this will help them in programming or even tasks in the real world. After mapping Cargo-Bot games to a set of learning goals, we devise a lesson plan that uses Cargo-Bot game playing to scaffold key concepts used in writing recursive Java programs. (Lee et. al, 2013).

Recursion is a fundamental concept in computer science that novices often struggle to understand. In particular, students often fail to understand the passive flow of recursion, which is the backward flow of control that can take place after reaching the base case (Scholtz et. al., 2010). Cargo-Bot, an educational video game in which users solve puzzles using a simple visual programming language. The players are to use recursion in solving every puzzle.

Lee and his group had a test wherein to evaluate the learning, they handed out the test on a controlled group. One group is to be taught on a lecture while the other group is to play the game. The game Cargo-bot where they

understood recursion better than the group who is lectured but the gap were not that large, but the result favored the group who played the game.

Since our project is designed for novice programmers, recursion is not part of the scope of the content. Instead, repetition, as the more basic and introductory concept, has been given the focus for level three.

2.4 Scratch

Scratch is a tool that can be used by students or teachers for educational or entertainment purposes from math projects, science projects, programming projects including simulations and visualizations of experiments, lectures with animated presentations, interactive art and music. It is a visual programming language that has no specific language to teach, but it lets the user create a program through manipulating a program by dragging the graphical elements rather than typing it manually. Scratch allows the user to use event-driven programming that has sprites objects. Event-driven programming is a fundamental style of computer programming which serves as a way of assembling the structure and elements of computer programs in which the control flow of the programs is determined by events such as actions like mouse clicks and key presses, sensor outputs, or messages.

For our project, however, event-driven paradigm is not the focus. Instead, we focused on using the imperative paradigm as the simpler alternative and the

more appropriate paradigm for a gaming environment that teaches novice programming concepts.

3. Methodology

3.1 Conceptual Framework

3.2 Methodology

3.2.1 Content Creation and Design

The user-interface will have a toolbox wherein the available commands are displayed. Then there will be a main box where the commands that are clicked are executed. Commands are Move Forward, Turn Right, Turn Left, Function Call and Loop. The main interface will be the puzzle to be solved.

3.2.2 Game Design and Development

3.2.2.1 Learning Objectives / Level Design

Programming concepts will be associated in each level. Different puzzles will be in every level and will be more difficult as the player progress. There are 11 levels which is categorized into three which is the introduction, basics, functions, and loops and each

levels has goals which is to collect all the gems and reach the door which serves as the finish line. Introduction is where the player learns about the basic mechanics of the game like forward, turning left and right. After introduction is the basics where in there is basic conditionals. After 3 level of basics, they will be given a new mechanic wherein they will have the ability to use functions. New mechanics in functions includes function panels where the player will put the commands and call the function name in main panel and after 3 levels they will be able to use the loop command. Loop has also its own panel where players can put the commands and set the counter according to the solution and call the loop in main panel.

The TUTORIAL Level

In this level, the user will be able to learn the mechanics of the game. The different objects and actions that can be done in the game will also be presented here. The tutorial will also present the storyline and the goals of the game.

Level 1 – Introduction

The Introduction level is comprised of puzzles that will allow the user to perform sequences of actions that follow the natural top-

down structure of the imperative programming paradigm. There is also a puzzle that will use the TURN action/object to make the user understand and appreciate the concept of control flow.

Level 2 - Basics

Level 2 presents puzzle on the concept and behavior of IF statements

Level 3 – Functions

Level 3 presents puzzles that illustrate the behavior of functions, e.g. how commands embedded inside this smaller units of procedures can be visualized and how they behave when invoked.

Level 4 – Repetition

Level 4 presents puzzles that illustrate the concept of repetition or looping. Here the user will have a visual experience of how each repetition or iteration happens.

The Game Objects:

- Gem – A crystal, wherein players need to collect

- Door – Serves as finish line of the game

The Game Actions:

- UP ARROW (or Forward) - command to move forward

- LEFT ARROW (or Turn Left) - command to turn left

- RIGHT ARROW (or Turn Right) - command to turn right

The Storyline

A ninja was in his mission to collect all the gems that can be found in an old dungeon. Unfortunately the ninja was trapped and the only way to get out is to solve the puzzles in each level wherein each level has different type of difficulty. The puzzle will be more difficult as the ninja progress. The player will need to command the ninja using functions and loops in order to solve complex puzzles. The ninja must collect all the gems in each level before reaching the finish line in order to proceed to the next level. After finishing all the levels, the ninja can now get out of the dungeon safely.

The Game's Goal

The player should command the ninja to collect all the gems and reach the door which is the finish line for each level.

3.2.2.2 Development using Unity

The game is developed using Unity script in Unity 3D, the game engine by Unity Technologies (Blackman, 2011). Unity is a flexible and powerful development platform for creating multiplatform 3D and 2D games and interactive experiences. It's a complete ecosystem for anyone who aims to build a business on creating high-end content and connecting to their most loyal and enthusiastic players and customer. We also used MonoDevelop as the Integrated Development Environment. MySQL was also used for the game's backend requirements.

3.2.3 Implementation

3.2.3.1 Implementation of the Game to a Class (Control Group)

Because of the need to test the game with a target audience, the CS111-Fundamentals of Programming class of Mr. Emmanuel Cabuloy was chosen as the control group. This class is composed of first year Bachelor of Science in Information Technology (BSIT) students. The class was given a pre-test to see what they already know before playing the game.

After the pre-test, they were asked to play the game for at least 45 minutes.

During gameplay, we observed their reactions to the game and made notes. After completing the levels, the participants were again asked to answer a post-test.

The pre-test and post-test questions were comprised of programming exercises that the participants can answer using the C programming language. Although the game was not specifically teaching the C programming language, but only teaching underlying concepts of basic programming, the students – being enrolled in a CS111 class and a C programming class – are able to translate this understanding to actual code.

Pre and post test scores were computed to solve for the post-test learning gain to answer the research question if there was indeed learning because of the game .

$$\text{Normalized Gain} = \frac{\%\text{ of Post test score} - \%\text{ of Pre-test score}}{1 - \%\text{ pre-test score}}$$

3.2.3.2 Validation of The Application by Mr. Cabuloy- CS111 Instructor.

Mr Cabuloy, the class instructor of the control group, was present during the whole duration of the implementation/deployment experiment. He also read the questions in the pre and post tests and agreed that the level of difficulty for the pre-test questions matches the level of difficulty of the post-test questions. He also gave feedback and points to improve on the game, its design and contents

His suggestions were:

.1 To limit the commands in main panel so that students are forced to use functions and loops.

.2 To see if the player collected enough gems to finish the level.

The game was then revised to incorporate the inputs of Mr. Cabuloy.

3.2.3.3 Revisions of the Original Design of the Game

The following changes were made to the game:

1. The instruction is shown by telling the students how each icon works and they are shown screenshots in order for them to follow. The instructions were made clear so that students can easily understand how the game works.

2. Added gem counter in order to know if the player collected enough gems

3. The main window now has limited slots so that the players are forced to use functions and loops.

3.2.3.4 Evaluation of the Revised Game

After incorporating revisions and making updates to CodeNinja, novice programmers were randomly selected. A total of 23 novice programmers (please see appendices for their profiles) were requested to evaluate the game for learning and gaming elements (Thomas, Schott, & Kambouri, 2004).

Pre-tests and post-tests were no longer conducted this time but qualitative comments and self-report were used to gather evaluation feedback for the game.

The participants ages ranged from 16-19 years old. All had introductory programming courses, and considered themselves as novice programmers.

They were asked to play the game for at least 45 minutes or until they finish all levels of the game.

During gameplay, we again took notes for our observations on the players. After gameplay, the players were asked to answer the evaluation tool.

The tool aims to answer the questions: Will the students be able to learn programming concepts with the help of a game that is designed for this purpose? Will the students be able to relate the game to the programming concepts?. In addition, the tool also included questions on usability and heuristics for an educational game as discussed in Thomas, Schott, & Kambouri, 2004.

EVALUATION TOOL

Learning Metrics

	Strongly Disagree 1	Disagree 2	Neither Agree or Disagree 3	Agree 4	Strongly Agree 5
1. I am able to understand the top-down concept of control flow as illustrated in the game					
2. I am able to understand the conditions and branching concepts as illustrated in the game					
3. I am able to understand the concept of function definition as illustrated in the game					
4. I am able to understand reusing a set of commands multiple times in functions as illustrated in the game					
5. I am able to understand the concept					

of repetition as illustrated in the game					
6. I am able to understand the constructs of loops, specifically the concepts of iteration, actions and counters as illustrated in the game					
General Comments/Suggestions:					

Usability Metrics

	Strongly Disagree 1	Disagree 2	Neither Agree or Disagree 3	Agree 4	Strongly Agree 5
1. I think that I would like to play the game frequently					
2. I think I would need the support of a technical person to be able to use this game					
3. I found the game easy to use					
4. I felt very confident using the game					
5. I am able to understand the concept of repetition as illustrated in the game					
6. I am satisfied with the game					
General Comments/Suggestions:					

The measures for evaluation to be used in analyzing the results of the tool will be Median and the Inter-quartile range (IQR). For a likert scale, the median will show the median ratings of the respondents. The IQR will determine where the middle 50% of the respondents are, or what their ratings are. For example if the IQR is zero, this means that the middle 50% of the respondents all gave the same results or were unanimous. The advantage of using the IQR as a measure for evaluating is the outliers will be removed. Unlike average or mean, where even extremes or outliers are included.

Pre test Questionnaires

- **FUNCTIONS**

- **LOOPS**

Post Test Questionnaires

FUNCTIONS

LOOPS

4. Theoretical Background

4.1. Game-based Learning

4.1.1. What is Game-based Learning?

Game-based learning is a type of game that has learning outcomes. Digital Game-Based Learning is precisely about fun and engagement, and the coming together of and serious learning and interactive entertainment into a newly emerging and highly exciting medium (Prensky, 2007). Educational games are learning tools that have game elements and educational purposes. Educational games are designed to help people understand a certain subject and help them to learn new skills but at the same time it gives people fun, excitement, joy and motivation that games usually have.

4.1.2. Advantage and Disadvantage of Game-based Learning

According to Keesee et. al., serious games allows players to focus, make decisions about a certain scenario and learning about the consequence of choice in the situation. As people become more committed to playing and finishing the game, they begin to care about learning more about the topic on how to solve a certain problem that the game offers. Games like this allow people to become active participants discovering new ideas and solutions to problems while allowing them to feel an excitement in every level. Educational games are also cost-effective and low-risk. Well-designed games allow learning experiences that are not possible in real life. The disadvantage is that the result is not a guarantee. The game may not give out the expected results as it may not make the player learn something. The uncertainty makes it hard for developers to develop a certain kind of game.

4.2. Learning in Games

4.2.1 Why use games as the learning tool?

Most of the learning tools are less entertaining than games and this is the reason why most of the learning tools are ignored. Making a game as a learning tool makes people motivated to play since games are fun and exciting to play. Since games are more welcoming than learning tool many

people will try to engage and play the game which means the learning that people will learn from the game is not wasted since the game is played, unlike the learning tools that are seldom used.

4.2.2 Learning Programming in games

Programming is a subject that is hard to master and requires you to think abstractly. To learn to program in games, you must make players think algorithmically to get a solution. Have game mechanics that make players apply the basic programming practices. First is planning wherein a player should plan out a way in order to solve each puzzle. Second is programming where the player will make a solution to the problem. Then the player will test the solution and if there are errors to the solution then the player will debug and goes back to step 1.

5. Results and Discussion

5.1 Concept Phase

5.1.1. Framework used for Content Design

During the concept phase, we patterned the content and pedagogical design for the game using Gouws, Bradshaw & Wentworth (2013)'s framework in their work entitled "Computational thinking in educational activities: an evaluation of the educational game light-bot". According to these authors, computational thinking relates to programming and taught concepts that have a "direct-link" to programming. These concepts are Control-Flow concepts

which are comprised of sequences of instructions, procedures (or functions), and loops. Understanding these control flow concepts have been highlighted and were recognized as important in learning (or teaching) programming.

Hence, for content, we have placed the following: Level 1 – Introduction (to include sequences of instructions), Level 2 - Basics(to learn the basics of conditionals), Level 3 – Functions (to understand the transfer of control to smaller units of tasks inside procedures or functions), and Level 4 – Repetition Concepts in Loops.

5.1.2 Conceptualization of Game Elements

The plot, setting and game mechanics were also designed. CodeNinja was developed to relate the challenging environment that the gamer will feel in relation to programming as also a challenging task. The game's environment is simple so as not to complicate the gamer's objective of understanding the desired programming concepts.

5.2. Game Development and Implementation

5.2.1 Creation of Local Database

Filename: SetUpDatabase.cs

When the user starts the game, the game automatically creates its own local database.

5.2.2 Register and Login

File name: AddUser.cs

The students will only need username in order to register and will only need to select their username in order to login.

Data Model for Persistent Data Requirements (User Profile, Levels and Scores)

The database saves the player's name and it saves the time the player finishes a level and the number of commands the player used.

5.2.3 Game Implementation in Unity3D

The game objects were created using Adobe Photoshop. Then the bulk of the work was with Unity 3D, the chosen game engine. All the models were imported to Unity 3D, structured into different game layers and appropriate scripts were written in MonoDevelop.

5.3 Results

5.3.1 Results of the Control Group Test

With the computed post-test learning gain, the overall normalized learning gain is -0.08% (with a standard deviation of 0.6) (almost 0%) which means that, overall, there was no increase in the post-test scores of the students. There were some students, however, where post-test learning gain can be seen, but because

since the average showed that there was no overall learning gain. The conclusion that the game increased the learning of the students after one session of gameplay cannot be made.

5.3.2 Usability Test 1

To validate the usability of the application in terms of HCI criteria, such as ease-of-use, interface design, and satisfaction (Perlman, 2009*),* a usability questionnaire was used (please see below).

	QUESTIONS
Q1	The application was simple and easy to use.
Q2	The software’s interface is appealing.
Q3	The software has all the functions and capabilities I expect it to have
Q4	The information provided by the software is clear.
Q5	The software increased my knowledge in loop and functions
Q6	Overall, I am satisfied with this application.

Questions 2 and 3 had a median of 3 which means the students find it fair that the app's interface was appealing and the expectations that the software has all the functions and capabilities the student expect it to have.

5.3.3 Evaluation Results from Deployment

The final testing involving novice programmers of sample size 23 revealed that:

For measuring the project's objectives on learning, the following are the results:

1. **Understanding of the "Top-Down" Concept of Control Flow.**

The respondents median rating is 5 with an IQR of 1. Five means they strongly agree that the game made them understand the Top-Down concept of structured programming. The middle 50% responses only differed by 1 rating interval, which would denote that the median rating of 5 can be considered as a majority answer.

2. **Understanding of the Conditions and branching concepts**

The respondents median rating is 4 and with an IQR of 1. Since they rated 4 it means that they agree that the game made them understand the of conditionals and branching.

3. **Understanding the function definition**

With a median rating of 5 and IQR or 1, they strongly agree that the game made them understand about function definitions

4. **Understanding the concept of reusing set of commands multiple times**

The respondents agree that they understood how to reuse set commands multiple times with a median rating of 4 and IQR of 1.

5. Understanding the concept of repetition

The respondents agree that the game made them understand the concept of repetition through the use of the game's loop feature.

6. **Understanding the construct of loops, specifically on the concepts of iteration, actions and counter**

The respondents were able to understand these concepts by giving them a counter on loops and to know what the next iteration is.

The following are the general comments and suggestions of the respondents referring to their evaluation of learning:

1. The game makes you really think hard because the puzzles are difficult and takes time.

2. The game is unique in teaching basic programming concepts.

3. Interesting concept.

4. The game is hard if you do not have a background in programming

The following are the general comments and suggestions of the respondents referring to their evaluation of the game's usability:

1. Instructions were well explained.

2. The design and graphics were simple enough to keep their interest

3. The game was fun

4. It takes time to finish a level.

Overall, for learning the median rating for all questionnaire items is 4 or Agree. For usability, the median rating for all questionnaire items is also 4 or Agree.

Learning Metrics

Usability Test 2

6. Conclusion and Recommendation

6.1 Conclusion

We were able to develop an educational game for teaching programming concepts using the framework that LightBot used in teaching programming concepts, i.e. Computational Thinking and Illustrative methods. The educational game, CodeNinja, is able to “teach” and result to the understanding of the concepts in programming like, control flow, branching in procedures and repetition in loops. In the validation and deployment experiment conducted with One CS111 professor and his 25 first year students, the game was evaluated in terms of applicability to C programming and usability. In this experiment, the game received acceptable results for usability, however, the post test learning gains can not lead to the findings that CodeNinja will result to

learning gains in terms of C programming. But the participants expressed their understanding of underlying concepts as stated in their evaluation responses.

For the evaluation with novice programmers with the objective of capturing whether the game will allow the students to learn and understand the programming concepts, the overall median rating as reported in the previous section is 4, which is a positive rating. This leads to the finding that the respondents/participants agree that the game has helped them understand the underlying concepts. In terms of usability, the game also was given positive ratings and results, i.e. the game, aside from enabling them to learn the desired concepts, was usable, easy to use, and fun to use.

6.2 Recommendation

We recommend that additional introductory concepts be included in the game's content for advanced levels. We also recommend a possible mobile version for the game as the results showed that the students liked to play it.

The participants also gave recommendations like : the inclusion of easier puzzles for those with no programming background at all, or for those who really find programming very difficult, as beginners; improve the graphical design and interface to make the game at par with other games that are in the same genre.

REFERENCES:

Tamarisk Lurlyn Scholtz and Ian Sanders. Mental models of recursion: Investigating students' understanding of recursion. In Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE '10, pages 103–107, New York, NY, USA, 2010. ACM.

Elynn Lee, Victoria Shan, and Bradley Beth. A structured approach to teaching recursion using cargo-bot. ACM New York, NY, USA, 2014, pages 59-66

Lindsey Ann Gouws, Karen Bradshaw, Peter Wentworth. Computational thinking in educational activities: an evaluation of the educational game light-bot, ACM New York, NY, USA, 2013, pages 10-15

Marc Prensky, Digital Game-Based Learning, Paragon House Publishers, 2007

Keesee, G. (2015). Educational Games. Retrieved January 29, 2015, from http://teachinglearningresources.pbworks.com/w/page/35130965/Educational Games

Park, A. (2014). Little By Little, Violent Video Games Make Us More Aggressive. Retrieved March 16, 2015, from http://time.com/34075/how-violent-video-games-change-kids-attitudes-about-aggression/

Malykhina, E. (2014). Fact or Fiction?: Video Games Are the Future of Education. Retrieved March 16, 2015, from http://www.scientificamerican.com/article/fact-or-fiction-video-games-are-the-future-of-education/

Squire, K. (2011). Video Games and Learning. Teachers College Press

Jenkins, T. (2002, August). On the difficulty of learning to program. InProceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences (Vol. 4, pp. 53-58).

Miliszewska, I., & Tan, G. (2007). Befriending computer programming: A proposed approach to teaching introductory programming. Informing Science: International Journal of an Emerging Transdiscipline, 4(1), 277-289.

Blackman, S. (2011). Beginning 3D Game Development with Unity: All-in-one, multi-platform game

Perlman, 2009. Usability and User Experience Surveys. Retrieved March 19, 2016, from http://edutechwiki.unige.ch/en/Usability_and_user_experience_surveys