Development of Serious Games on Programming Logic

Ross Nolan

05 November 2017

# Abstract

Today our student are being labeled as digital natives. As this is the case, it leaves an area for more investigation. Computer programming is being brought into junior schools across the world. It's no longer a hobby for enthusiasts. The curriculum of computer science is evolving and the data flow of subjects is being changed regularly. Serious Games have been used for a number years to introduce different subjects. There is a case in Italy where they introduce workers to a restaurant through a Serious Game. It brings them through the restaurant and how to deal with customers through a text speech game. There is a multitude of Serious Games. This literature review explains in detail Serious Games in the genre of computer programming. These games are designed to teach novice programmer's the basics of computer programming and computer programming logic. With student understanding these basic's it gives them a stronger foundation for understanding programming concepts.

Index: **Computer Science, Serious Games, Games Development**

# Introduction

As we progress more into the digital era we need to educate the natives in the latest trades. Computer science in some schools across the globe is being introduced as a subject with state exams. TAMPA Preparatory School in Florida, America, has introduced Robot C, C programming, and Swift iOS app development to go along with engineering and robotics (Thompson 2017). Coder Dojo is another example of pro-programming. Its an out of school club, young and old people can join. They hold coding events on weekends where they teach the student how to code. There are also other courses that teach computing through a single context, digital storytelling workshops, and robotics camps. Computer Science is becoming a more widespread field. A majority of everything that is delivered to us has Computer Science embedded into it, from the machines that make your clothes to your toothbrush. Our daily life revolves around the Internet of the things, whether it's in front of us or related to it in some way. As this is the case, the education of programming languages and programming logic is important for these students. Serious games or educational games is a good way to introduce novice programmer to the understanding of programming comprehension, see: Related Work section.

The reason for this literature review is to answers question's on how to develop the research project. The following questions need to be answered.

* With today’s student’s, are digital games a good platform for education?
* Considering Serious Games about programming and programming logic. Who uses them as tools for learning? What was the general approach for their use? What age group is using them? Are people using them for past times or for learning?
* One of the issue's educational digital games have in a learning environment is that they may not address the subject in full. How can this project be delivered so that it addresses the depth of that subject?

# Related Work

When teaching programming, one of the concepts that need’s to be grasped is programming logic. If a programming language is taught with the logic in an introductory computing class, it tends to ”distract from the core issue of algorithmic problem solving” (Shackelford and LeBlanc 1997). Programming Comprehension is when the student understands what the program is doing (Gugerty and Olson 1986). The author of (Lu and Fletcher 2009) suggest that computational thinking is required to prepare students for programming courses §3. New students to a programming module may have problems because they lack the programming comprehension, even if they can remember the syntax. Through the use of games, as a complement to the standard teaching curriculum, teachers found that the lessons are more absorbed than with a stand-alone lesson (Carrington, Baker, and van der Hoek 2005).

Educational games or Serious Games are designed so as not to intimidate the user who is unfamiliar with a specific subject. Serious Games are designed for a specific purpose (Deterding, Dixon, Khaled, and Nacke 2011). In the case of games based on programming or programming logic, they are useful to a student who does not understand the concept of ”loops or iteration”. There are a number of games that exist that aim to deliver the comprehension before the syntax.

**RobotOn!**

RobotOn!(Miljanovic and Bradbury 2016) was designed for first year Computer Science students learning C++. The user's average age was between 18-20 years of age. It was built with the cross-platform game engine Unity using C# as its scripting language. It is a game that helps the student understand programming comprehension(Miljanovic and Bradbury 2016). The game has a number of different tools and comprehension tasks that are based on simple programming concepts. The tasks are given to teach control flow, code behavior, variable purpose and data flow. As they progress through the stages, their progress is saved to log files. Once they have completed the game, the students will have a basic understanding of programming comprehension.(Miljanovic and Bradbury 2016).

The author's methods were to evaluate the student's experiences with a number of questions. Is the game playable for undergraduate students? Do the students enjoy the game? Does RobotOn! teach C++ comprehension sufficiently? These are all relevant questions toward the future development of educational or serious games. Answers to questions like this will help designers in future to create a more directed approach to building serious games. Having a keener sense to what a student needs, is the underlying answer on how to develop these games. Unfortunately, there were no results for this particular study.

# Alice2

Alice2 (Kelleher, Cosgrove, Culyba, Forlines, Pratt, and Pausch 2002) is designed to help students understand programming through building a 3D world with a drag and drop approach. With this design, it prevents syntax error's giving the user more confidence to experiment.The author believes there are 3 main tasks a new user might find difficult. 1) Finding a structured solution to a problem. 2) Express the solution in a syntax. 3) Understand the behavior of the program. In a user test of Alice1, they found that typing the syntax was a dominant problem(Pierce, Audia, Burnette, Christiansen, Cosgrove, Conway, Hinckley, Monkaitis, Patten, Shochet, et al. 1997).

In Alice2, the interface consists of a scene window, an object tree, the object details area, the animation editing area and the behaviors area. The user is able to drag and drop commands into animations giving it a result of the animation moving. One of the observations made during the study was that the students would often look through an object tree for ideas on what to build. This leaves the impression that worrying of correct syntax was not there. Alice2 capable of building programs to the size of 3000 lines(Kelleher, Cosgrove, Culyba, Forlines, Pratt, and Pausch 2002). Alice2 gives the user the opportunity to explore conditionals, count loops, while loops, variables, parameters, and procedures. It also implements a simple form of parallel programming and programming constructs found in Java and C++.

# LightBot

The author of Lightbot states, "It(programming) is more about the process with which we come to a solution and think algorithmically about how to solve a problem.". Lightbot is a programming puzzle game. The solution to the puzzles is in direct correlation with programming concepts. LightBot puzzle game is a drag and drop game. You drop the tile commands into an instruction box, telling a character on the screen what do. The objective is to light up all the ground blue tile’s in each level. Each of the command tiles represents a different move, for example, move forward one space or light up the tile the character is standing on. The author states the game is broken up into two sections, Programming Practices, and Control-Flow. Programming practices are - Planning, Programming, Testing then Debugging. ControlFlow is - Sequence Instructions, Procedures, and Loops.

# Computational thinking(LightBot)

In Computational thinking in Education(Gouws, Bradshaw, and Wentworth 2013), the authors address the problem of identifying, evaluating and incorporating computational thinking into education. They use a framework to assess computational thinking in the game LightBot. Their author named the framework, Computational Thinking Framework(CTF). It designed to work for a range of applications.The CTF is pictured in Figured 1.

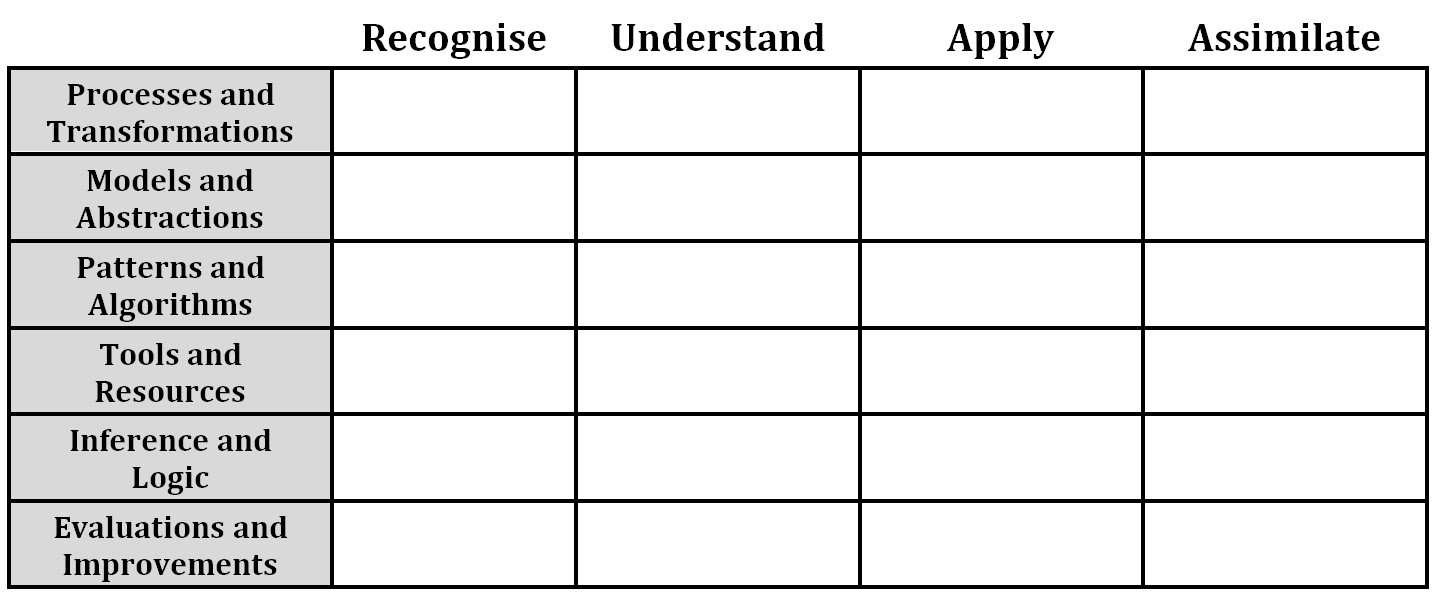


Figure 1: CTF model

The vertical axis is broken down into 6 distinct areas.

* Process and Transformations
* Models and Abstractions
* Patterns and Algorithms
* Tools and Resources
* Inference and Logic
* Evaluations and Improvements

The horizontal axis is broken down into what the authors call 4 levels.

* Recognize
* Understand
* Apply
* Assimilate

**The vertical axis**

## Process and Transformations

Computational exercises circle around Processes and Transformation(Hu 2011). The student can also be introduced to activities of input, output and parallel processes(Hu 2011). This section is breaking down a problem into multiple steps and then solving in a logical manner.

## Models and Abstractions

The student needs to think abstractly to create elegant solutions to such problems as data storage and manipulation(Hu 2011). Models can reinforce the comprehension of abstraction(Hu 2011).This section is how to represent a problem and solution(Gouws, Bradshaw, and Wentworth 2013).

## Patterns and Algorithms

The author says that the likes of loops and iteration would all be representative of patterns in computer science(Hu 2011). If a student is able to recognize a pattern this will help them solve a problem(Hu 2011). This section is the ability to recognize patterns in a problem(Hu 2011).

## Tools and Resources

The author views programming as tools for students. The student's way of thinking with programming is using tools to help solve a problem.A student selecting an appropriate tool to solve a problem is an accomplished programmer(Hu 2011). This section represents the availability of tools used to solve a problem(Hu 2011).

## Inference and Logic

The student use of logic is important to solve computational problems. This section represents the logical skills used to solve a problem.

## Evaluations and Improvements

The level a student can locate and break down a problem for an error. The level of understanding of why the error occurred.This section represents debugging and performance.

**The horizontal axis**

## Recognize

At the recognized level this represents the student's ability to recognize and remember computational problems.

## Understand

This level correlates to the student's comprehension of the computational problem.

## Apply

This level relates to the student's ability to apply a solution to the given problem.

## Assimilate

This level will measure the student ability to use the tools given in full. How to use smaller solutions with larger ones to solve a problem.

## LightBot case study

The authors used LightBot to collect a Computational Thinking Score. After studying all the different levels they, they were able to gather what type of skills were required to pass the level. They then filled out the CTF and score were assigned to each block. Scores ranging from 0-3 indicate whether a concept is complete or not, 0 being least and 3 being best. The scores are then aggregated and converted to a percentage. The results were as follows in Figure 2.

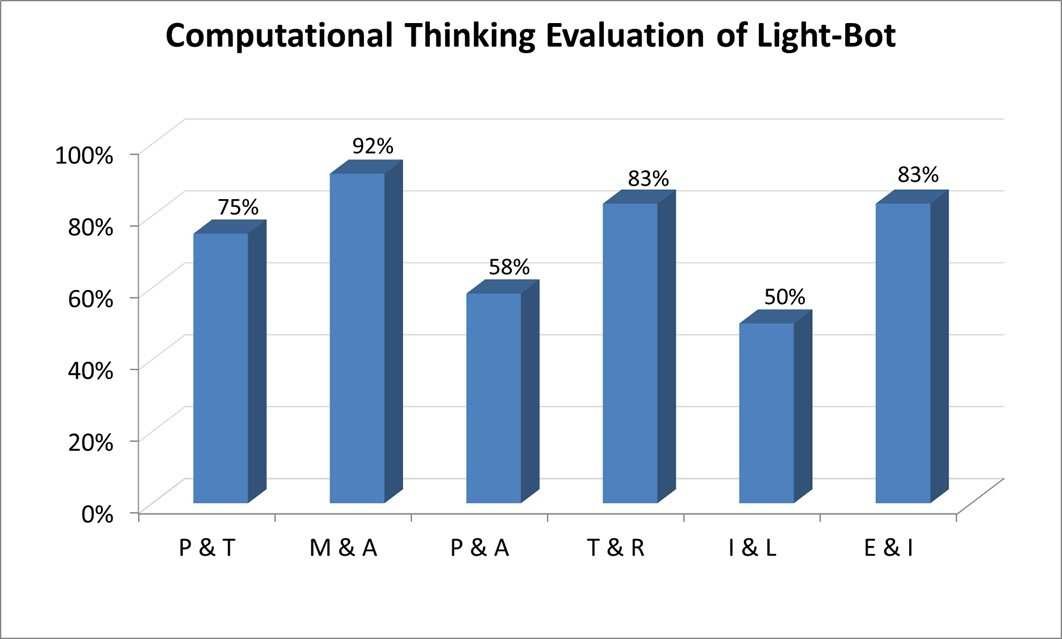


Figure 2: Results of Test

When developing a serious game a framework like this should be taken into account. The results show where an application could need to be improved, giving the designer a more focused idea on how to deliver the intended message.

# Conclusion

In conclusion to this review, Serious Games, and my research project, Serious Games is an area continuing to grow as with the rest of the Computer Science sector. As there is more investigation into the subject, a more thorough model will develop to answer translations of lessons to games.

CTF has been used to evaluate project's to increase the level of usefulness towards understanding the computational thinking. After developing the Serious Game, it will use the CTF to gather a score. The game will be adjusted to gain scores where necessary.

The drag and drop system for input of code used in both Alice2 and Lightbot have a strong appeal to the user who is unfamiliar with syntax. As it is a strong approach to the novice to make them feel more confident in the complications of programming, the game will be implementing this into the research project.

# References

Carrington, D., A. Baker, and A. van der Hoek (2005). Its all in the game: teaching software process concepts. In *Frontiers in Education, 2005. FIE’05. Proceedings 35th Annual Conference*, pp. F4G–F4G. IEEE.

Deterding, S., D. Dixon, R. Khaled, and L. Nacke (2011). From game design elements to gamefulness: defining gamification. In *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments*, pp. 9–15. ACM.

Gouws, L. A., K. Bradshaw, and P. Wentworth (2013). Computational thinking in educational activities: an evaluation of the educational game light-bot. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, pp. 10–15. ACM.

Gugerty, L. and G. Olson (1986). Debugging by skilled and novice programmers. *ACM SIGCHI Bulletin 17*(4), 171–174.

Hu, C. (2011). Computational thinking: what it might mean and what we might do about it. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, pp. 223–227. ACM.

Kelleher, C., D. Cosgrove, D. Culyba, C. Forlines, J. Pratt, and R. Pausch (2002). Alice2: programming without syntax errors. In *User Interface Software and Technology*.

Lu, J. J. and G. H. Fletcher (2009). Thinking about computational thinking. *ACM SIGCSE Bulletin 41*(1), 260–264.

Miljanovic, M. A. and J. S. Bradbury (2016). Robot on!: a serious game for improving programming comprehension. In *Proceedings of the 5th International Workshop on Games and Software Engineering*, pp. 33– 36. ACM.

Pierce, J. S., S. Audia, T. Burnette, K. Christiansen, D. Cosgrove, M. Conway, K. Hinckley, K. Monkaitis, J. Patten, J. Shochet, et al. (1997). Alice: easy to use interactive 3d graphics. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*, pp. 77–78. ACM.

Shackelford, R. L. and R. LeBlanc (1997). Introducing computer science fundamentals before programming. In *Frontiers in Education Conference, 1997. 27th Annual Conference. Teaching and Learning in an Era of Change. Proceedings.*, Volume 1, pp. 285–289. IEEE.

Thompson, G. (2017). Coding comes of age: Coding is gradually making its way from club to curriculum, thanks largely to the nationwide science, technology, engineering, and mathematics phenomenon embraced by so many American schools. *THE Journal (Technological Horizons In Education) 44*(1), 28.