TEST DRIVEN DEVELOPMENT OF A GAME BUILT WITH UNITY AND C#

By

Ross Nolan

Supervisor(s): Dr. Matt Smith

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
M.SC. IN COMPUTING
AT
INSTITUTE OF TECHNOLOGY BLANCHARDSTOWN
DUBLIN, IRELAND
19 APRIL 2018

Declaration

I herby certify that this material, which I now submit for assessment on the programme of study leading to the award of **M.Sc. in Computing** in the Institute of Technology Blanchardstown, is entirely my own work except where otherwise stated, and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fulfillment of the requirements of that stated above.

Dated: 19 April 2018

Author: Ross Nolan

Abstract

Games development is a growing trade. It is constantly evolving as an industry and with new technology being developed and distributed on a regular basis, we are always trying to improve our disciplines and techniques.The game developed in Unity was designed with the architectural pattern, model view controller or usually abbreviated as MVC.

Motivation: Why do we care about the problem and the results?

If the problem isn't obviously "interesting" it might be better to put motivation first; but if your work is incremental progress on a problem that is widely recognized as important, then it is probably better to put the problem statement first to indicate which piece of the larger problem you are breaking off to work on. This section should include the importance of your work, the difficulty of the area, and the impact it might have if successful.

Problem statement: What problem are you trying to solve? What is the scope of your work (a generalized approach, or for a specific situation)? Be careful not to use too much jargon. In some cases it is appropriate to put the problem statement before the motivation, but usually this only works if most readers already understand why the problem is important.

Approach: How did you go about solving or making progress on the problem? Did you use simulation, analytic models, prototype construction, or analysis of field data for an actual product? What was the extent of your work (did you look at one application program or a hundred programs in twenty different programming languages?) What important variables did you control, ignore, or measure?

Results: What's the answer? Specifically, most good computer architecture papers conclude that something is so many percent faster, cheaper, smaller, or otherwise better than something else. Put the result there, in numbers. Avoid vague, hand-waving results such as "very", "small", or "significant." If you must be vague, you are only given license to do so when you can talk about orders-of-magnitude improvement. There is a tension here in that you should not provide numbers that can be easily misinterpreted, but on the other hand you don't have room for all the caveats.

Conclusions: What are the implications of your answer? Is it going to change the world (unlikely), be a significant "win", be a nice hack, or simply serve as a road sign indicating that this path is a waste of time (all of the previous results are useful). Are your results general, potentially generalizable, or specific to a particular case?

Acknowledgements

I would like to thank Dr. Simon Wilson...

Table of Contents

List of Tables

List of Figures

Abbreviations

MVC	Model View Controller
TDD	Test Driven Development

Chapter 1

Introduction and Background

1.1 The Game

The game is a puzzle adventure. It uses programming logic to solve the puzzles presented.

1.1.1 Serious Games

1.2 Architectural Patterns

Describes how the MVC relates to my project

1.2.1 MVC

Using MVC architectural patterns are a common choice for games development. It is favoured for its modular approach. The separation of the model view controller interfaces allow me to build on each module without disturbing the other. When i need to expand on the development of the model class, i will be able to do with ease. There will be minimal, if any, disturbance's to the rest of the program. When the changes are made from the model class, those changes can be displayed to the user interface.

The LevelModel class represents the structure of data that holds the attributes of a level

design. The LevelView class will display the attributes to the user interface . The LevelController class is used to exchange data between the two objects. In the case of this project, the LevelController will also manage the Input controller.

Model

View

Controller

1.2.2 Singleton

Don't Destroy On Load

1.3 Test Driven Development

1.3.1 Subsection header 1

1.4 Git Hub, Version Control

1.5 Games Development

1.5.1 Unity Engine

level editor

reading file

NetHack ...

C#, Syntax, Refactoring:

Chapter 2

Literature Review

2.1 Abstract

Today our student are being labelled 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 basics, it gives them a stronger foundation for understanding programming concepts.

Index: Computer Science, Serious Games, Games Development

2.2 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 todays students, 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?

2.3 Related Work

When teaching programming, one of the concepts that needs 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.

2.4 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.

2.5 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++.

2.6 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 tiles 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.

2.6.1 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.

	Recognise	Understand	Apply	Assimilate
Processes and Transformations				
Models and Abstractions				
Patterns and Algorithms				
Tools and Resources				
Inference and Logic				
Evaluations and Improvements				

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

2.6.2 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.

2.6.3 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.

2.6.4 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.

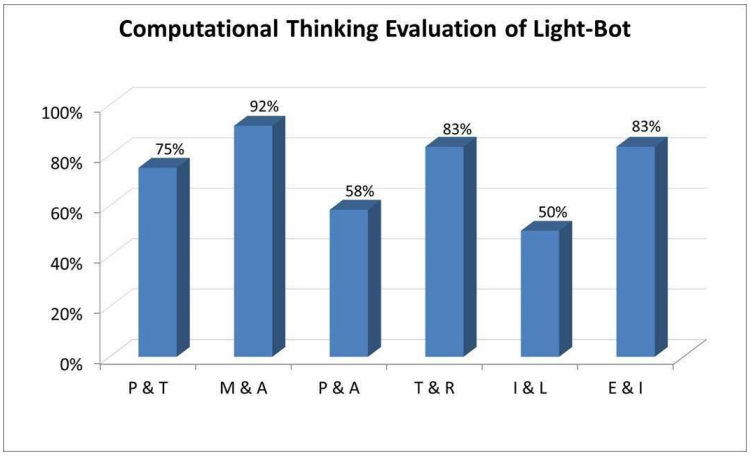

Figure 2: Results of Test

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.

2.6.5 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.

2.7 section header 1

2.7.1 Subsection header 1

2.7.2 Subsection header 2

2.7.3 Subsection header 3

2.7.4 Subsection header 4

2.8 Notes

Faster performance, on game logic, not really a big deal, but i other situations it is.

Why faster performance Faster performances and possibly bad code vs normal performance and solid code(tested??).

CPU cycles

Design patterns , for and against, and why MVC

General thoughts on techniques picked up along the way //commenting on exact places as i try to debug(tricks of the trade)

what is the point of a thesis From my understanding, a thesis is another word for dissertation. Both are names for a written report about the research that you have done as part of your academic requirements.

Naming Convention From my own study and experience, when naming variables they need to make sense, and to follow up with where the logic is directed. Else, its hacky.

Dynamic Naming??Providers... For better automatic comments...

URL to Git Hub, nice layout page

return (lightable == 0); returning true this kul instead of just true

unreachable code in a true false situation, or any for that matter

Order of invoke

Show study of Serious games

MVC

refactoring

theres a few extra lines of code, but the way it works, is it will go straight to the value instead of a loop, removing the need to go through all other values

TDD

Data Providers

check nethack

public static readonly(SquareTypep[] LightableTiles = new SquareType[3];)

Game Logic (Performance) Performance Scripts (Job lists?) Entity Systems. Which lead into no if else loops(Nvidia Self driving cars, actualy ,but they were onto somthing) which brings it back to performances scripts, but removes game logic, because game logic isnt the heavy,i think)

Variables at the start of a program, the public private, check runtime,when they run and what happens to them.

Trying to eliminate all loops,find out what happens in the background from direct search in array, to a loop array, Do a statistics on Loop and direct grab (Unity Profiler)

Programming baggage? Debt? Design debt, when i hard code because of order(for example), and then i cant use it later.

annonymous methods

List Action...func

set the model(Level Model) //Alll on the same page

To make assertions about collections, you should use CollectionAssert:

CollectionAssert.AreEqual(expected, actual); List¡T¿ doesn't override Equals, so if Assert.AreEqual just calls Equals, it will end up using reference equality.

CollectionAssert.AreEquivalent

Delegates.... Event handlers...

Level State -¿ mapSize 10 -¿ playerX playerY start position [1,1] -¿ starting conditions(starting state), (State after move etc)

Display window(eidtor behaviour??)

noun and verbs to methods

Method signitures

Further work

Website....

hot controll on mouse click, no other buttons to be pushed when mouse pushed down

mcv sittng on top of service layer

Try to prove the way i did the MVC instatiation is the beter or a good way to do it Also, try to prove that the way i did the loop is more efficient.

Object pooling!

UX- 2 arguments for a situation, where , direct mode would display the input, north south example, or where just the instruction list would display. The direct mode, not needing the text because the user just pushed the button, but the list because the user need to see the list. Being a minimal approach.

Further work.. Fix program commands

Why am i using the specific methods, why am i using MVC, why test driven development, unity etc. Are these methods really capable of addressing the objectives? The project has a dominant MVC architectural pattern.

My objectives are:

- To build a puzzle game that will teach basic programming logic.
- To develop a test driven game, for the purpose of fully tested code free from bugs.
- To maintain and research MVC and the befits of using the pattern in games development.
- To test a mid to high level style of programming.
- To increase written code efficiency with elegant solutions.
- To refactor where possible.

2.8.1 Research In action

How am i going to execute the research specifically when it comes to the order of things, Am i going to write the code and then test.

I might want to gain an initial understanding through writing the code, then construct the tests.

Try to address how im going to research each objective.

Objectives .... Methods

Make sure the objective are all handled by at least one methods

2.9 Research Plan

2.9.1 Exploratory

If MVC or test driven methods are used often in the industry

2.9.2 Descriptive

How many company are using these methods

2.9.3 Explanatory

Why are these methods used

Chapter 3

Design and Implementation

The design of the software is an MVC pattern for the main game logic components. These components are broken into two sepereate systems. The two systems are Direct Mode and Programing Mode. I will explain each one seperatly, and then explain how i implemented them both as one system.

3.1 Direct Mode Design

3.2 Direct Mode Implementation

3.3 Progamming Mode Design

3.4 Progamming Mode Implementation

References

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.