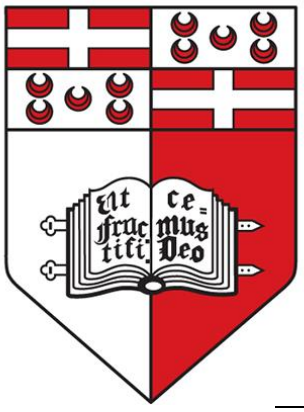


Group Assigned Practical Task

ICS 2000

Julian Agius

Nikolai Debono

Michelle Falzon

Introduction

- Overall brief of project – aims and objectives
- Summary of functionality developed
- Uses
- Approach used in carrying out the project;
- Any assumptions on which the work is based,

The aim of this project was to create a fun and interactive game to help 10 – 12 year olds understand the basic concepts of programming with Prolog. Our game, Furball, was designed using Unity and is intended to teach recursion, backtracking and pattern matching.

Furball consists of 5 levels. In the first level a treasure chest is shown to the player. This chest is locked and can only be opened by using a key found in a later level. The aim of the game is to solve all levels and return to open this chest.

In every level the player must collect *furflies* to break down walls and solve the level. Each *furlfly* is coloured and on collection, the same coloured wall is destroyed. Thus, the user must solve this pattern matching puzzle to advance to the next level. This mimics recursion as each level corresponds to a simpler version of the first level.

Upon solving the third level, the user obtains the key and must return to the previous levels to reach the chest (main goal) once more. The revisited levels have the same basic structure but the *furflies* and their walls have moved to form a new puzzle. Going through these levels once more brings about the idea of backtracking, reaching the base case and unwinding to solve the main goal.

Literature Review

Prolog (Programming Logic) is a high- level programming language based on formal logic. Unlike most programming languages which carryout sequences of commands, Prolog is a declarative language. This means that a program written in Prolog will contain a list of facts and rules on which a query is run. One main advantage of using logic programming is that it can be used to express knowledge in a way that does not depend on the implementation making it versatile. As a result, it is often preferred when working with artificial intelligence natural language processing and expert systems.

Our game aims at explaining backtracking, recursion and pattern matching to young children. Recursion deals with solving a problem in terms of smaller versions of the same problem. A good example of where recursion can be used is to solve the factorial of a number. In order to solve 3!, the program will first solve 2!. In order to solve 2!, the program will compute 1!. Each recursive method will consist of a stopping condition. In the factorial case the stopping condition may be the answer of 1!. One this is obtained, the program unwinds and solves 2! And finally 3!. This unwinding is called backtracking. Once the program returns to the first call of the recursive method, either an answer is obtained or the program backtracks further to find another overloaded method.

Moreover, the game focusses on pattern matching. In Prolog, this involves comparing the parameters of a function to see which overloaded method should be tried. For example, using the factorial example above, we have a recursive step and a base case which will be the factorial of 1: Factorial(1) = 1

Factorial(n) = n* Factorial (n-1)

Thus when the function is called using 3 as a paremeter, it tries to mach the first case and fails since 3 /= 1. It then moves on to the next case and uses that since n represents any number. This pattern matching allows methods to be overloaded.

Further to this, research was made on the elements that make games educational yet fun and engaging. Each research paper listed a number of features which it deemed important. The final product tried to incorporate a number of the common elements.

Due to the young age of the target audience, simple graphics and lighthearted music were chosen to make the game look and sound interesting. The game allows children to learn the concepts of logic programming in a challenging way. Each level consists of a small pattern matching puzzle which they are required to solve in order to proceed to the next level. These challenges are aimed at keeping the children fully immersed in the game and allowing them to use their creativity to find different ways of solving the same level. According to the author of 'How To Create Effective eLearning Games' giving the user the power to choose how to solve the level is important as it gives the user the impression that he/she is in control of his/her learning experience. The children may also need to formulate a plan before attempting to solve certain levels as the furflies needed to be collected in a certain order to complete the level. As a result, the game was made challenging but not too difficult.

Many papers outlined the importance of interaction and a coherent theme. This was achieved by allowing the audience to control the main character in the game in a restricted environment. The player may collect items and explore the environment easily.

As clearly stated by Karl M. Kapp in his article 'Eight Game Elements to Make Learning More Intriguing' and Mr. Dave Gaymon in '5 Elements of Good Games that Can Make Us Better Teachers', the risk of failure makes the game more immersive by allowing the users to take risks and learn accordingly. Thus, thorns were added to keep the player cautious and make the levels a bit tougher.

Lastly, the game works with a single goal in mind, to collect the treasure in the chest. This is done to keep the player focused on one main goal while trying to solve other subgoals. Once completing all levels, the player is guaranteed to obtain the treasure. This sure probability of obtaining the gold after successfully finishing the levels ensures that the player will always be justly rewarded to provide ego gratification.

Technologies Employed

The game was designed using Unity, a game development platform that can build 2D and 3D games. Unity also allows for games to be ported to Windows, Linux and Mac operating systems, as well as mobile devices. The programming languages used for the development of Furball was carried out using C# and JavaScript. C# was used to program:

- the player gameobject
- the relation between a furfly and its respective wall(s) of the same color
- wrap around and gravity-change features

- the thorns to “kill” the player and restart the level

JavaScript was used to create the start menu.

Design

The objective of the game is to open the chest found in level 1. However, at the start of the game, the player does not hold the key. After completing a few of the levels the player will find the key required to open the chest and travel back through the levels to find and open it.

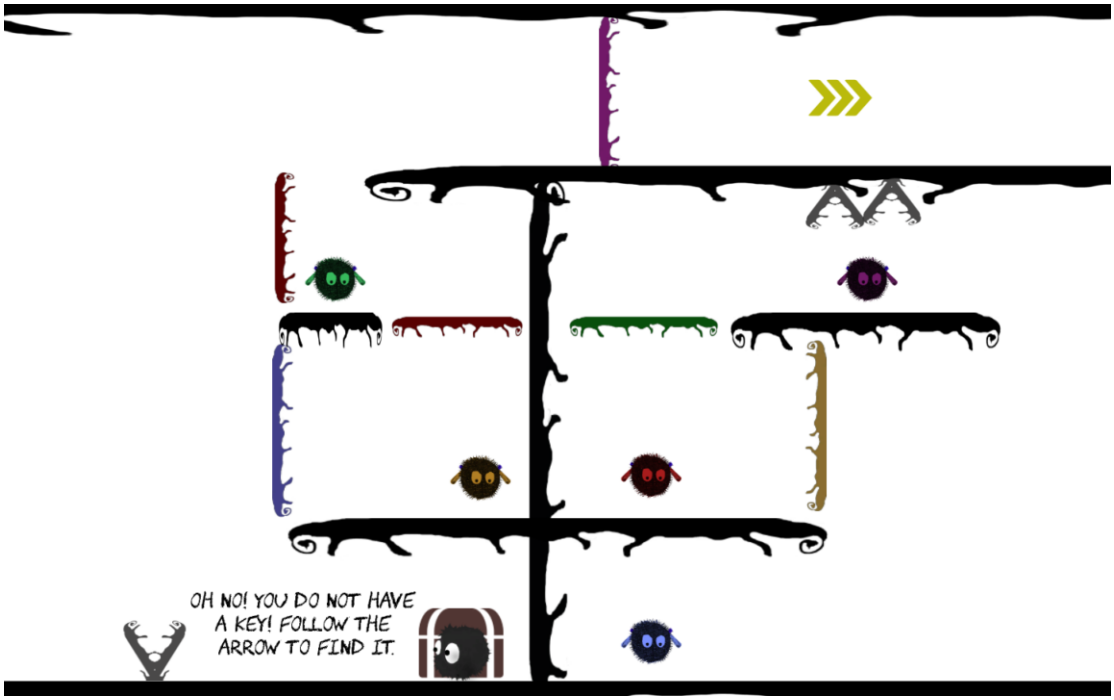


Figure 1: Level 1

The player can move left and right and can also jump, similar to other platform games. Furball can also wrap around the screen. This means that although the screen space is finite, the player can leave one side of the screen and end up on the other side, while maintaining their speed and trajectory.

Figure 2: Level 2

The player can destroy a wall by catching a furfly of a similar colour. By doing so, the player can unlock a new area within the level itself. In certain levels, the order in which the furflies are caught is very important as can be seen in Figure 3.

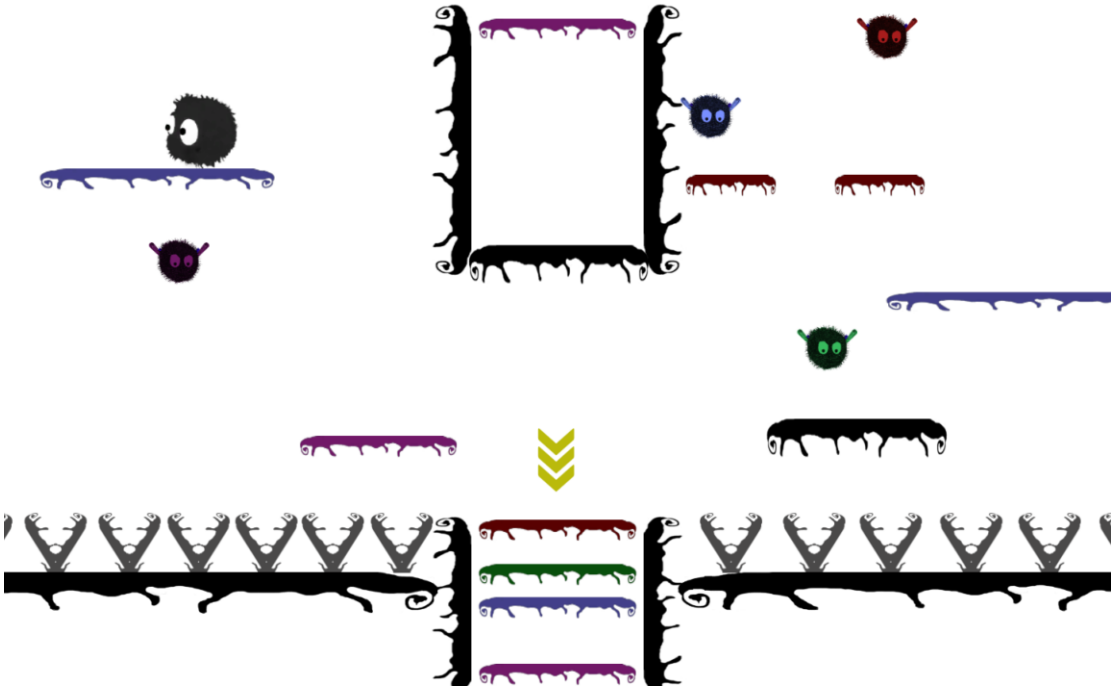


Figure 3: In level 2, order is important

Since the player collected the orange furlfly, the orange wall was destroyed, it is impossible to collect the purple furlfy.

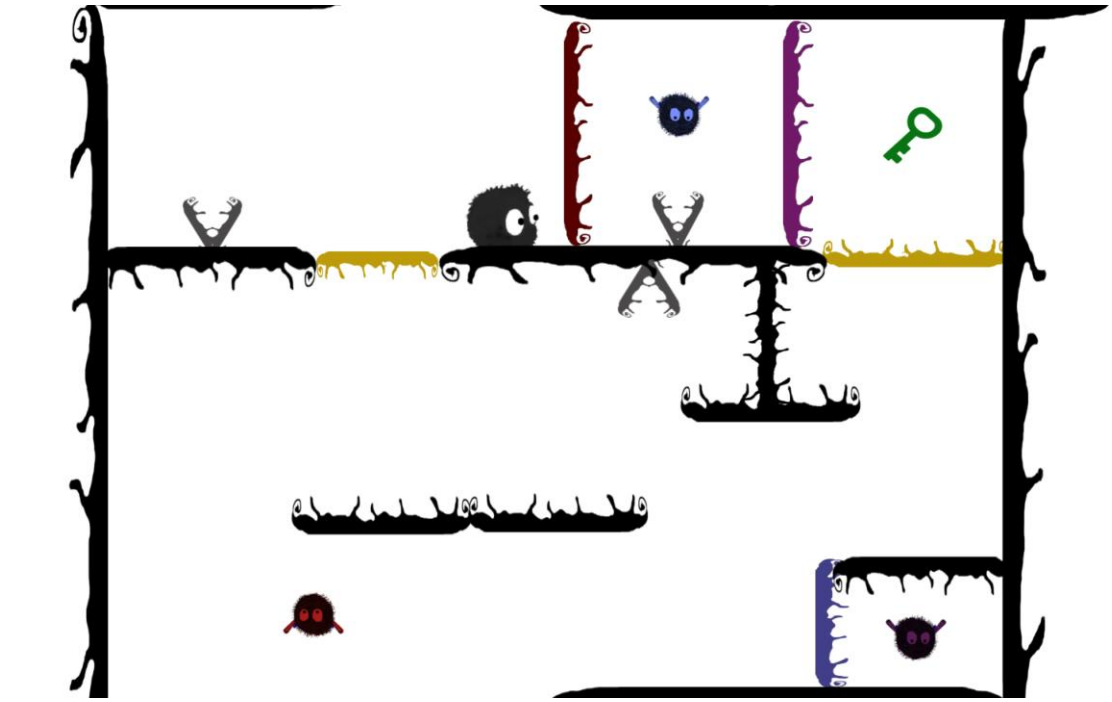


Figure 4: Level 3

When stepping on the yellow ground, gravity changes and flips the player upside down. The player can still move left and right and can also jump. To change gravity back to normal the player will have to find a new yellow tile.

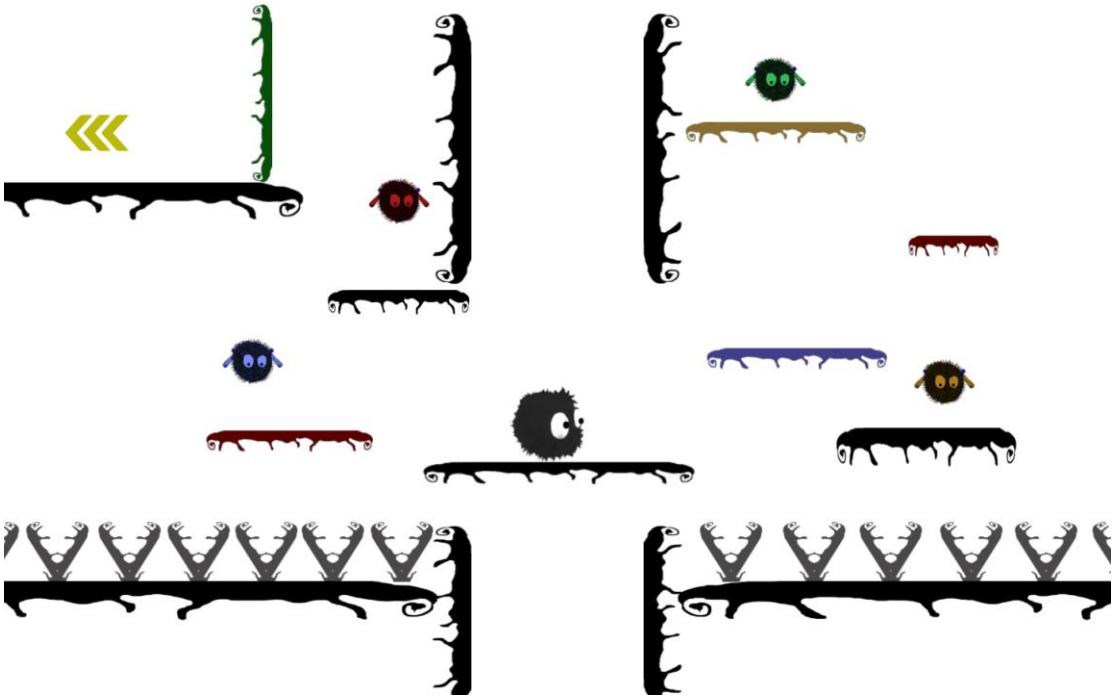


Figure 5: Level 4

Each level is littered with thorns which will "kill" the player and reset the level. These thorns restrict the player from certain parts of the level and will force some "out-of-the-box" strategies involving the wrap-around game feature.

Figure 6: Level 5

Implementation

Link to assets used. ->

The player game object has a physics controller attached to it so that it obeys the rules of gravity. When the game object touches a yellow surface, the gravity of the level is switched.

Each furfly has a collider attached to it so that when the player collects the furfly it triggers the destruction of similar coloured walls and of itself.

The wrap around game feature was programmed using C#. The Wrap() method found in the WrapAround.cs script checks whether the player game object is on screen. If the game object was found leaving from the left side of the screen, its position is changed to the right side of the screen. The height, speed and possibly, the jump trajectory are unaffected when the wrap around occurs. Similarly, the player can wrap around from top to bottom and vice-versa.

Figure 7: System architecture

Technologies employed; Unity C-Sharp, JS

- Similar systems investigated; platformer, http://gamedev.stackexchange.com/questions/83893/how-can-i-make-backtracking-interesting
- Pros and Cons -> Nikolai D

- Design and specification
- Implementation details and Testing;
  - System architecture

  - How to play (Design)
  - Basic concepts of the game -> thorns, wrap around movement, jump + gravity, key-chest
  - Scripts (Implm) physics controller, trigger-furfly, + explain basic concepts in relation to scripts
  - Testing
  - problems that may have arisen during implementation
  - code that are especially critical to the operation of the system;
    - you feel might be of particular interest to the reader for some reason;
    - are exemplary, i.e. they illustrate an algorithm, data structure, etc. that is used widely throughout the system.

Testing

The aim of our testing phase was twofold. Firstly we wanted to make sure that mechanics critical to our game were functioning as intended. This led us to playing the levels over and over, testing that all game mechanics were in order. The features we tested for include:

- Movement such that the player is able to move and jump smoothly. This was tested by repeated playtime.
- Vertical and horizontal wraparound where the player can possibly access one side of the map by moving through the opposite side of it.
- Wall – furfly relation where upon collection of a particularly colored furfly, a wall of that same color gets destroyed. This was testing by going through all the levels and clearing all the furflys, making sure the right colored wall gets removed.
- Area accessibility where we made sure all the platforms and objects that had to be reached actually had a traversable path.
- Scene continuity to make sure that between one scene and another the game keeps track of any required information (ex whether final key was collected) and also to make sure that during the process of backtracking the levels look similar to the starting ones.

The second aim of the testing phase was to measure how well the game could teach the concepts of recursion, backtracking, and pattern matching. To this end we visited the PwC Academy during their Kids at PwC course. Kids@PwC is an IT training programme for kids where they learn various computing concepts through many tools like computer games and other educational software. Every student was given a laptop on which they could play Furball. Once they were done the students were given a small questionnaire (Figure 1) for them to fill out in order for us to get some feedback.

Playing with Prolog Questionnaire

Age:

_____

1. Do you know what programming is?

   ☐ Yes ☐ Partially ☐ No

2. Have you heard about Prolog (Logic Programming)?

   ☐ Yes ☐ Partially ☐ No

3. Were you able to finish the game?

   ☐ Yes ☐ No

4. Were the demo levels provided helpful to understand the basic game concepts?

   ☐ Yes ☐ Partially ☐ No

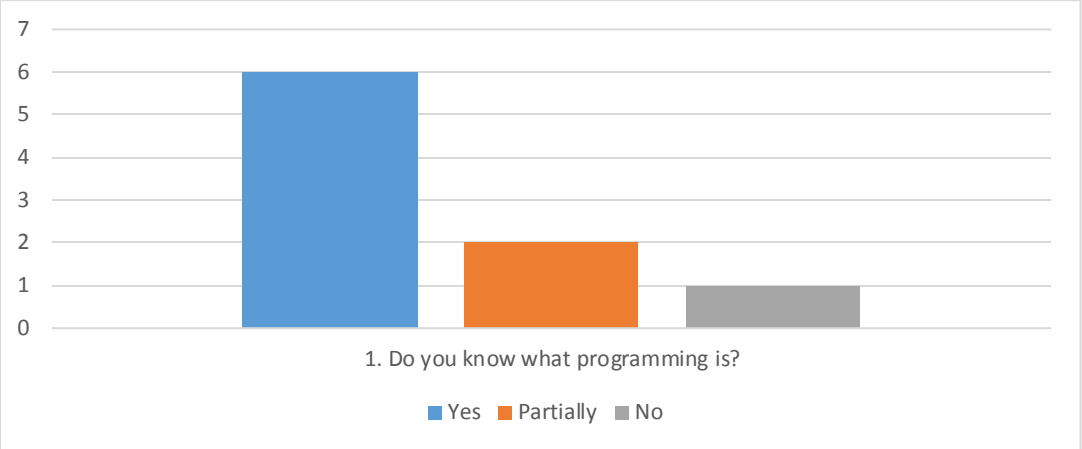
5. Did you understand the concepts of pattern matching and backtracking?

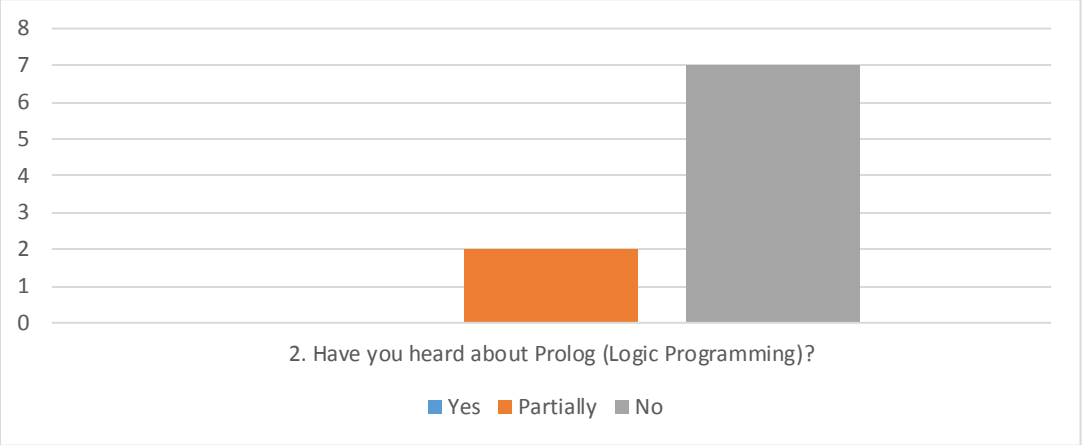
   ☐ Yes ☐ Partially ☐ No

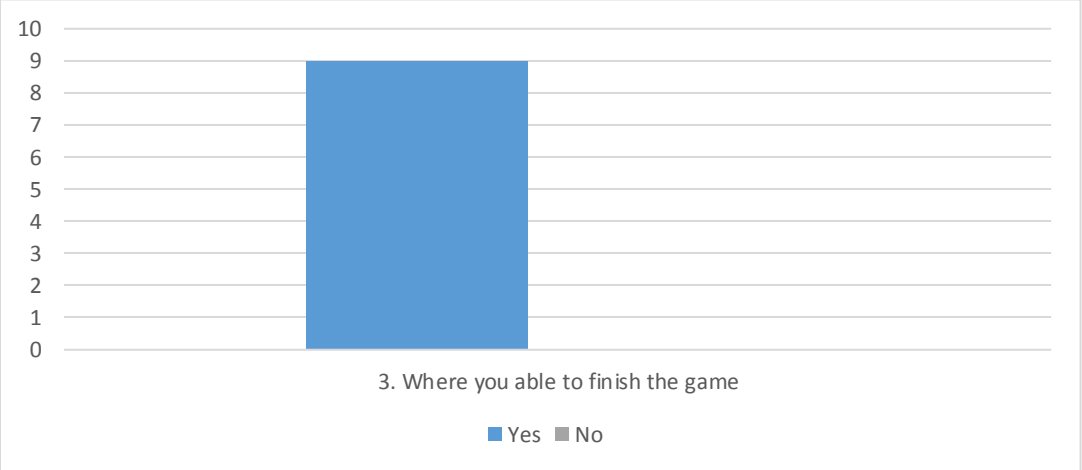
6. Did you enjoy playing the game?

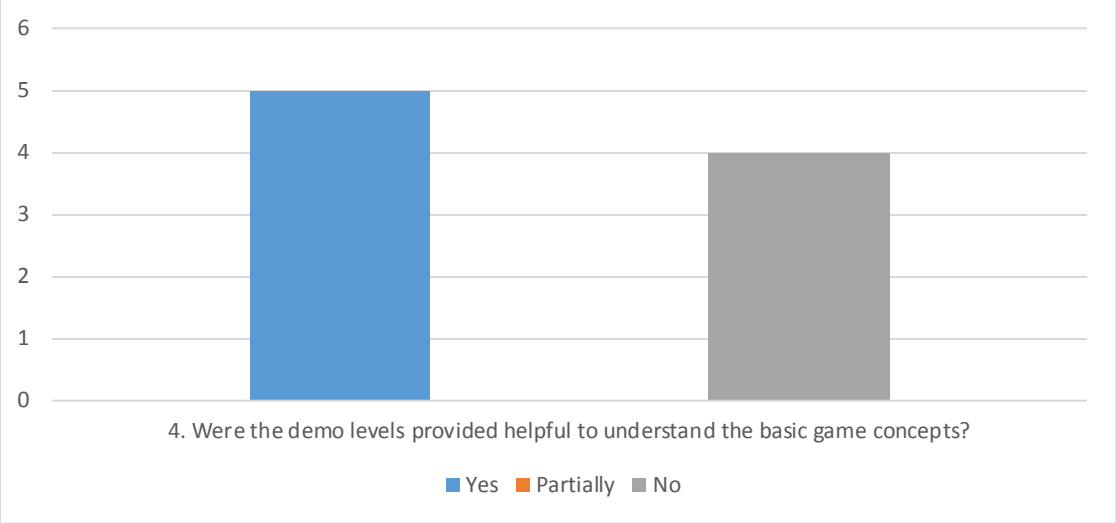
   ☐ Yes ☐ No

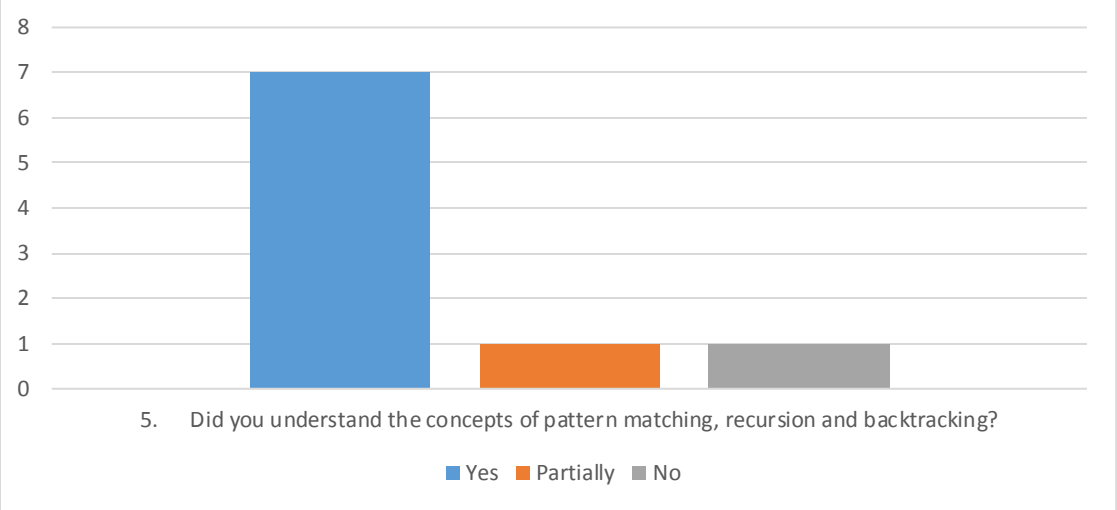
Results of questionnaire

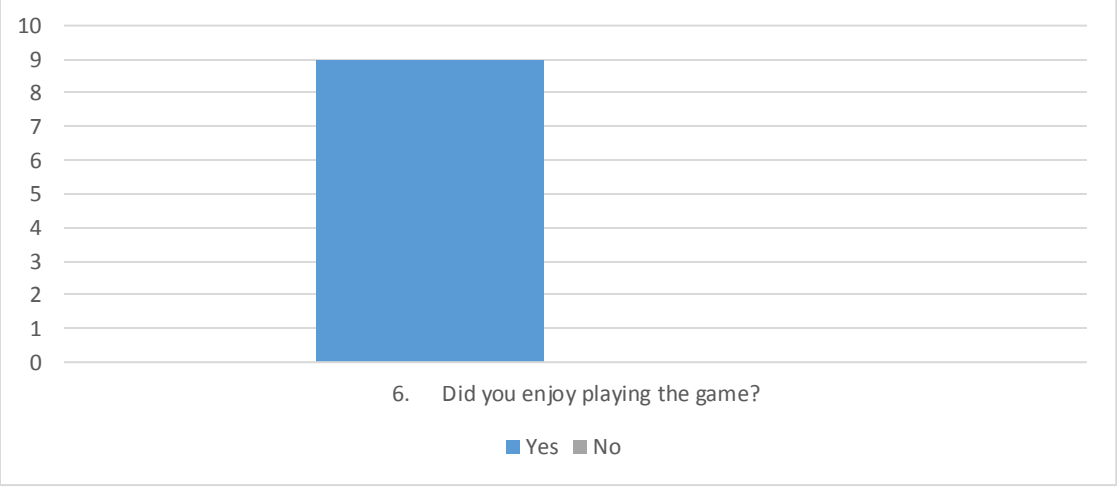







6
5
4
3
2
1
0
4. Were the demo levels provided helpful to understand the basic game concepts?
Yes
Partially
No


8
7
6
5
4
3
2
1
0
5. Did you understand the concepts of pattern matching, recursion and backtracking?
Yes
Partially
No


10
9
8
7
6
5
4
3
2
1
0
6. Did you enjoy playing the game?
Yes
No

Evaluation and Critical Analysis

With regards to the game mechanics testing that was performed, the results showed that

- The player is able to move and jump adequately during all game scenes.
- Vertical and horizontal wraparound were working correctly throughout the game, even when gravity is flipped.
- The Wall – Furfly relation was functioning as intended in all its instances, including the places where one furfly has many walls of the same colors related to it.
- One area was really hard to access due to a thorn game object unintentionally blocking the way. This was fixed before the students at PwC tested the game.
- The game kept track of all required information between levels.
- The game scenes showcasing backtracking (the ones where the player is going 'back' through the levels) look similar to the ones through which the player originally passed through.

The results from the testing performed during Kids@PwC gave us a lot more information about the educational aspect of our game. The results showed that

- The students had prior knowledge of basic programming
- Through Furball the students understood recursion, backtracking, and pattern matching.
- The game demos could be improved so that they explain the game better.
- The game was easy enough for it to be completed by everyone, yet still fun and educational.

During our session at the PwC Academy we also got feedback by interacting with the students and also observing them playing the game. In general the game was well received from all children. They had fun playing all five levels despite the challenges they encountered in every level. Game mechanics like movement, jumps, and wrap around are intuitive however it took time for the students to learn how they can be applied to solve the puzzles in every level. One of the children found a bug where jumping from the top left side of scene 2 would cause the play to get lost, thus requiring a restart of the whole game.

The game encourages students not to give up. Despite failing multiple times, every student was eager to get to the key in the third level and using it to complete the game. The concept of pattern matching was understood early during their time playing the game. Once they knew the relation between a furfly and a wall, the students automatically knew they had to match the colored furflys in the right order for them to advance to the next level. With regards to recursion and backtracking, the students understood the concept at the later parts of the game since it was then that they noticed the levels they're now completing are very similar to ones they'd already gone through.

With regards to further work, the bug on level two ought to be fixed as soon as possible. Next thing would be for the game to have demos that help children understand the game concepts better as some students didn't find them helpful enough. We will also add more

levels as students really enjoyed playing the game and wanted to have more challenges to solve.

Conclusion

The aim of this project was to create a fun and interactive game to help 10 – 12 year olds understand the basic concepts of programming with Prolog. Our game, Furball, was designed using Unity and is intended to teach recursion, backtracking and pattern matching.

Furball consists of 5 levels. In the first level the user encounters a chest that he must unlock. Upon trying to open it, the player is told he must go through the levels to find the required key. After solving the third level, the user obtains the key and must return to the previous levels to reach the chest (main goal) once more. The revisited levels have the same basic structure but the *furflies* and their walls have moved to form a new puzzle. Going through these levels once more brings about the idea of backtracking, reaching the base case and unwinding to solve the main goal.

Every level has furfly-wall pairs of different colors. The player has to match the furflies with their respective walls and collect the furflies in an order that allows him to solve the level. This showcases the pattern matching part of our game where the player learns that in prolog a solution to a problem can be found by simply using pattern matching of variables in a particular order.

To check whether the game is able to teach these concepts we visited the PwC Academy during one of their Kids@PwC sessions. Furball was well received by the children. They found all five levels to be both fun and challenging. By the end of it, the children were able to describe in simple terms the prolog concepts of pattern matching, recursion, and backtracking.

- References. – unity forums and doc - APA

Prolog

http://www.webopedia.com/TERM/P/Prolog.html

logic programming

http://www.doc.ic.ac.uk/~cclw05/topics1/summary.html

backtracking

https://www.cis.upenn.edu/~matuszek/cit594-2012/Pages/backtracking.html

recursion

http://www.wolframalpha.com/input/?i=recursion

pattern matching

http://c2.com/cgi/wiki?PatternMatching

Educational Game Features

http://www.marcprensky.com/writing/Prensky%20-%20Digital%20Game-Based%20Learning-Ch5.pdf

https://www.td.org/Publications/Blogs/Learning-Technologies-Blog/2014/03/Eight-Game-Elements-to-Make-Learning-More-Intriguing

http://elearningindustry.com/7-tips-create-effective-elearning-games

http://ocw.usu.edu/instructional-technology-learning-sciences/instructional-games/AERA06_IDgames-bes_dw.pdf

http://gettingsmart.com/2014/02/5-elements-good-games-can-make-us-better-teachers/

- Appendices
  - Slide presentation
  - code