UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Final Paper 2020

Title: Programming Education: Using the Raspberry Pi and games to teach programming

Author: Martin Flanagan

Project Abbreviation: PyPiGame

Supervisor(s): Gary Stewart

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	15
Theoretical Analysis	*0*	*25*	0
Experiment Design and Execution	*0*	*20*	10
System Development and Implementation	*0*	*20*	15
Results, Findings and Conclusions	*10*	*20*	10
Aim Formulation and Background Work	*10*	*15*	10
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	0
Total marks		**80**	**80**

Programming Education: Using the Raspberry Pi and games to teach programming

Martin D Flanagan
University of Cape Town
Rondebosch, Cape Town, South Africa
FLNMAR011@myuct.ac.za

ABSTRACT

The concept of programming in Computer Science can be a challenging concept for students to learn. Usually students need to learn both a programming language and general programming constructs at the same time and this can be difficult for new students who have not learned these things before coming to a tertiary institution. The motivation to undertake this project was to ultimately help students who will take an introductory course to programming by creating an assignment which would help the students to understand the programming concepts and to learn them as they do the assignment.

This project aimed to find out whether integrating 2D games and the general concepts of a programming assignment would be better than just using a standard programming assignment and whether by using a a single board computer with a add-on board with a 5-button joystick to act as the input for the game assignment engages the students more in their work.

The system which was created was a 2D game based assignment which runs on the Raspberry Pi while using the SenseHAT add-on board on the Raspberry Pi as the input for the game. This was all coded in Python using the PyGame engine with the SenseHAT module for the 'controller'. It was evaluated by staff from the University of Cape Town which had to first complete the game by following the assignment instructions in the PDF document, then they had to complete a survey mainly intending to find out the difficulty of the questions as well as whether the questions were clear enough for first year students to understand. The completed system includes the completed code (the memorandum for the assignment), a code scaffold and an assignment instruction PDF document. The results showed that the questions are of a standard that is acceptable for a first year student to understand and to complete with some minor adjustments to be made to the code by adding comments and to the PDF document to make things a little more clearer.

CCS CONCEPTS

• **Applied computing → Interactive learning environments**; • **Hardware → Sensor devices and platforms**; **Sensor applications and deployments**.

KEYWORDS

Computer Science, Education, Games, Raspberry Pi, Programming, Python, PyGame, 2D programming

ACM Reference Format:
Martin D Flanagan. 2020. Programming Education: Using the Raspberry Pi and games to teach programming. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Computer Science is a fast growing field. As a Computer Science student one has to take an introductory course to programming in the first year of your studies, but this course is not just limited to Computer Science students. Students from other faculties can also take this kind of course be it out of self-interest or that the requirements of their degree state that they need to take the course.

1.1 Problem

The main goal of an introductory course in Computer Science is to educate the students about the various programming concepts while having them learn to program in a programming language as well. Programming alone can be one of the subjects that is challenging to learn [2, 5, 13] and high failure rates are common in these courses [8] Research has shown that if these assignments are made in a way that is more engaging, the students grasped the concepts quicker and retained the information better while enjoying the course [1, 5, 10, 11, 15, 17]. By using single board computers in these assignments [10, 15, 17] and by creating these assignments in the form of a game [1, 5, 11], the students were more engaged in the assignments and enjoyed completing them.

1.2 Research Questions

Both games and programmable computing devices have become common in the world today and by using these two things, we can create innovative solutions to current problems. This research aims to understand whether using the Raspberry Pi (RPi) and introducing game-based assignments will help students to engage themselves in the assignments (and ultimately the course) and whether the students learn better with assignments of these types.

The questions we aim to answer are:

- Could creating usable 2D game assignments allow students to effectively learn course materials?
- Could game-based assignments requiring an external input device be a motivating factor for programming students?

These questions aim to see whether using both the RPi and game-based assignments together may increase the students engagement in the assignment and motivate the students as well.

1.3 Aim

The aim was to develop a system which will use the RPi's SenseHAT add-on board as the input for the assignment and to make the assignment game-based, where the student will be able to visually see how the code which they will write can have an effect in a game-based environment, instead of an assignment which would just be coded in a text editor and viewed in the terminal as text. This would be done by having the students connect their laptop/workstation to the RPi, complete the code for the assignment on the RPi and test it out by running the assignment (which is effectively playing the game), using the Sense HAT as the input for the game. To do this the assignment needed to be coded in such a way that it would be easy enough for the students to complete, yet testing their knowledge about a certain aspect of programming in each assignment.

1.4 Developed System Overview

The system which was developed is a game-based assignment in the form of a maze game. This was all coded in Python, using the PyGame engine. It runs on the Raspberry Pi as an application which is launched from the terminal by a simple Python command. The input for the game is obtained from the joystick on the RPi's SenseHat add-on board.

There are 2 software versions of this assignment:

- The fully completed assignment (The code memorandum for the assignment)
- The student version of the assignment (The code scaffold which the students will receive to complete)

Along with the two software versions of the assignment is the questions for the assignment in the form of a PDF document which can be found in Appendix A.

2 BACKGROUND

The main aim of an introductory course in programming in Computer Science is to develop one's problem solving skills [8], but this does not seem to be the case as Fee et al. [4] states that only a few students who take such courses want to improve their problem solving skills. The general requirements for these Computer Science courses are critical thinking, a general idea to problem solving and computational thinking [4, 16]. A lot of the students do have these skills, but still struggle learning programming. They tend to learn more about code syntax [13, 16] and they also worry about getting programs to run and resorting to trial and error methods to complete their assignments [2, 13], this should not be the case.

When including games on course projects, Leutenegger et al. [11] found that students took a liking to this approach and they retained the information that they learnt better. Feldgen et al. [5] found that using this game-based approach increased the success rate of the students. They did this by teaching small constructs first and using their assignments as building blocks allowed them to grow these assignments to become a more complete game application.

Ak et al. [1] conducted a study comparing different learning environments (2D, 3D and traditional) to each other. The same game mechanics were kept across all the groups in separate learning environments to keep the study as fair as possible and they found that the learning gains for all 3 learning environments were high and had no significant difference and in the end. It was noted that students are more likely to go for the 2D games because the students value it more [1]. This is why the system which was produced used a 2D game.

While looking at the different game engines two were identified at first: Panda3D and Alice. Panda3D [9, 12] is an open-source, free-to-use engine that can be used for 3D games. It uses both Python and C++ programming languages [12] but Python code is used on the developing end. Alice is a 3D interactive graphics programming environment which is built on the Python programming language with the aim to make development easier for beginners in programming with the use of 3-dimensional environments [3]. Later PyGame was identified as the game engine required for this project as the decision was to create a 2D game and for other reasons stated later in this paper.

Single board computers and micro-controllers were also used in programming assignments to help teach programming concepts. The Arduino board was used by Rubio et al. [15] in a study whereby using breadboards and circuits which connected to the Arduino the students could learn about variables and arrays and could see how it can affect the environment. The Raspberry Pi (RPi) is a "low-cost, credit card sized computer" [7] that is just as powerful as a normal computer and the Sense HAT is an add-on board for the RPi which has an 8x8 LED matrix, a 5-button joystick and a few sensors [6]. The RPi was also used at Indiana University Purdue University Indianapolis (IUPUI) to teach the internet of things [17]. They made use of the sensors on the RPi itself in the assignments and the final assignment was a "Raspberry Pi stock ticker" [17].

While researching four different programming languages stood out which first year students would typically learn. Scratch [10, 13], Python[3, 9, 10], Java[3, 10] and C++ [11, 12, 14] were identified. Below are some 'hello world' programs in each of these programming lanuages:

```
In Java:
    public class ClassName{
        public static void main(String [] args){
            System.out.println("Hello world.");
        }
    }

In C++:
    #include<iostream>
    int main(){
        std::cout << "Hello world." << std::endl;
        return 0;
    }

In Python:
    print("Hello world.")
```

While looking at the famous 'hello world' programs in each on these it was found that:

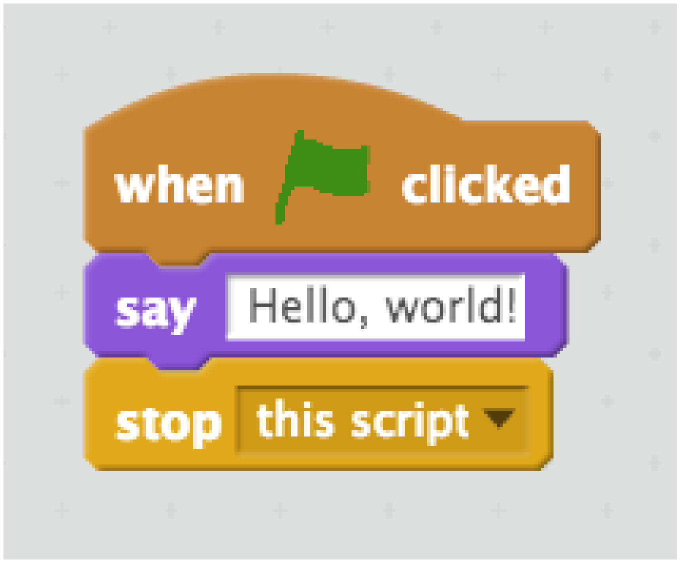


Figure 1: An example of the hello world program in Scratch

- In Java, the students will have to understand the constructs of a class and a main method
- In C++, the concepts of a main method and importing needs to be understood
- In Scratch the user will have to use sprites in order to accomplish this

In Python it is just a simple line of code. No understanding of importing and methods are needed, which makes it easier for a student to learn Python straight off the bat.

3 DESIGN

3.1 System Design

Before designing and creating the system, meetings between us and the supervisor were held to gather the requirements as to what we needed to research about and what hardware and tools were needed to complete this system. (The hardware and software needed as well as the type of game needed to be created.)

3.1.1 Game Engine. After considering a few game engines we decided to use the PyGame engine. PyGame was much more lightweight than the other game engines and was easy to install on a computer and import into a Python code file. It was a good choice as because of it being lightweight can be run on any desktop device and has some features that can be used in a game loop to keep resources from being depleted, such as limiting the number of frames per second so that you do not need to update the game unnecessarily.

3.1.2 Hardware. The reasons the Raspberry Pi was chosen is that as research showed that including single-board computers and microcontrollers the level of engagement from students should rise as they are using a piece of hardware they would not normally use everyday. The additional bonus is that the SenseHAT add-on board has a joystick and by using the joystick on the SenseHAT, one will be able to get the feeling as if they are playing the game whereas using a normal keyboard input might not give one the feeling of playing the game. The Raspberry Pi can also make use the game engine PyGame. Therefore we do not need to limit ourself to using a normal desktop to dedicate resources to image processing as the game engine mentioned above is lightweight enough to do that utilising the resources on the Raspberry Pi alone. The RPi with its SenseHAT was chosen over the Arduino board because the RPi comes pre-installed with Python and the Sense HAT functions library (a code library containing all the various functions for the Sense HAT) is written in Python.

3.1.3 Game Assignment. The type of game chosen was a maze game, where the user can move a character around a maze to the goal state. By implementing such a game, we can test various programming constructs such as: variables, loops, conditional statements and more while creating a simple enough game that can use the game engine properly.

4 IMPLEMENTATION

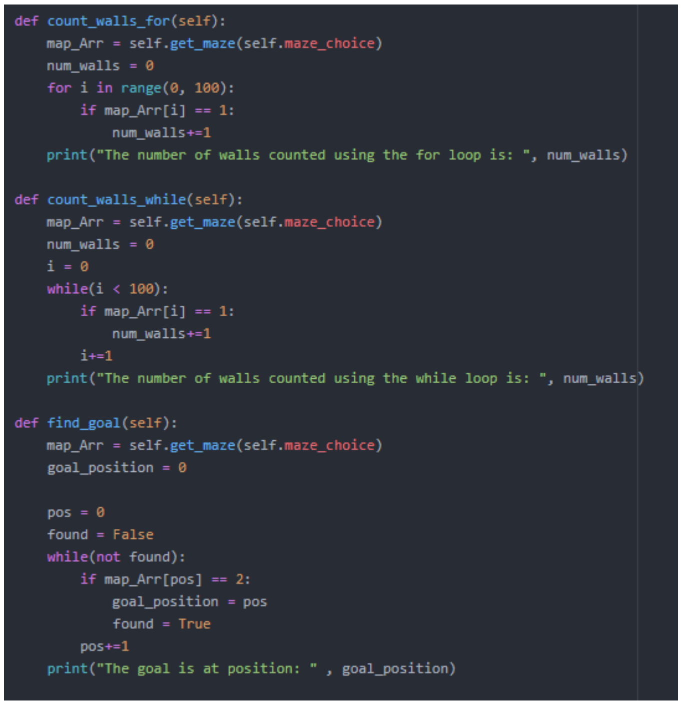
An agile implementation approach was adopted in this process. The first step was to plan in detail which features would be added to this system, implement it, test it, correct it and then retest it and when the feature was completed the next iteration of feature(s) would be implemented. Throughout this process the information obtained from our supervisor had to be looked at to as to keep on track as to what was needed in this game assignment.

The game was then created using the PyGame engine with the SenseHAT controls as inputs. Some collision detection was also needed to ensure that the character was within bounds at all times and that the goal state can be achieved whilst keeping the code simple enough so that it would be readable to first year students. Figure 2 shows the completed code.

The next step was to refactor the code so that there were portions of the code which would be easy enough for first year programming students to be able to do on their own with a bit of help from an assignment document. This would have been difficult had the game been more complex. Due to the game being simple, it was easy enough to get through this process. This process was lengthy as all sections of the code had to be covered to ensure that there would be enough to make a simple assignment. The deliverable from this part was the memorandum code. Figure 3 shows the code scaffold section of the portion of the code in Figure 2, while Figure 4 shows the question relating to the code shown in Figures 3 and 2.

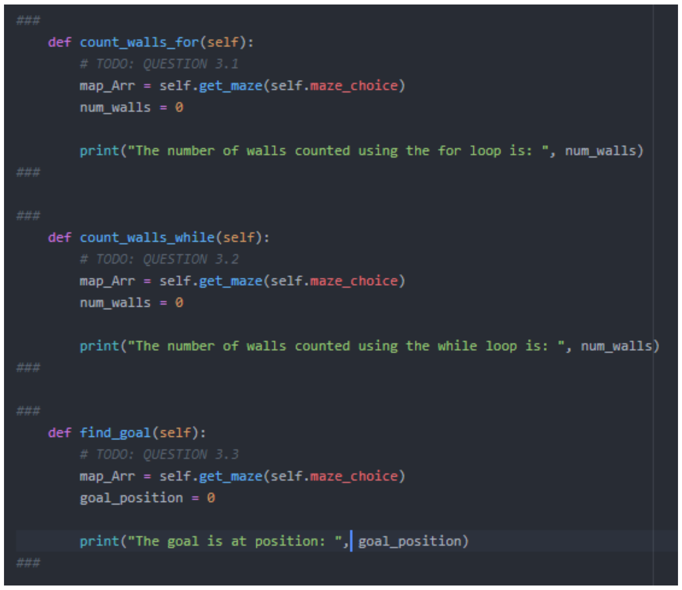
The final step was to create an assignment document with the questions needed to complete this game and to create a code scaffold for this assignment. This was easily achievable due to the refactoring step mentioned above. The deliverables from this part was the code scaffold and the assignment document. Figure 4 shows a question from this assignment document.

We then had to check with our supervisor if this was good enough and to make any changes if there was something unclear or errors were generated.



```
def count_walls_for(self):
    map_Arr = self.get_maze(self.maze_choice)
    num_walls = 0
    for i in range(0, 100):
        if map_Arr[i] == 1:
            num_walls+=1
    print("The number of walls counted using the for loop is: ", num_walls)

def count_walls_while(self):
    map_Arr = self.get_maze(self.maze_choice)
    num_walls = 0
    i = 0
    while(i < 100):
        if map_Arr[i] == 1:
            num_walls+=1
        i+=1
    print("The number of walls counted using the while loop is: ", num_walls)

def find_goal(self):
    map_Arr = self.get_maze(self.maze_choice)
    goal_position = 0

    pos = 0
    found = False
    while(not found):
        if map_Arr[pos] == 2:
            goal_position = pos
            found = True
        pos+=1
    print("The goal is at position: " , goal_position)
```

Figure 2: An example of the completed code



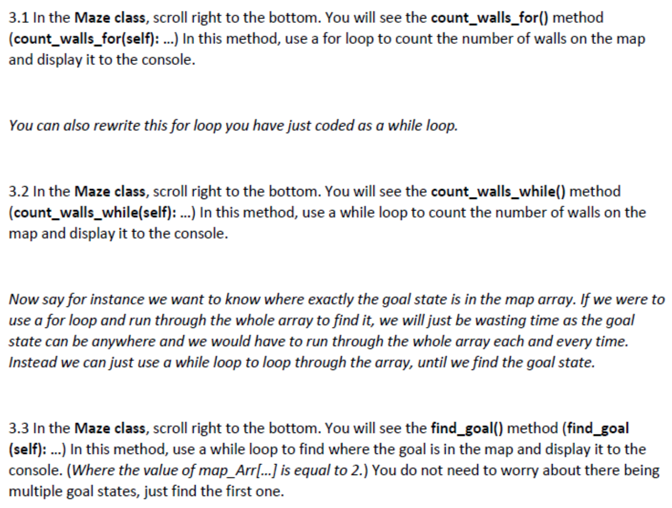
```
###
    def count_walls_for(self):
        # TODO: QUESTION 3.1
        map_Arr = self.get_maze(self.maze_choice)
        num_walls = 0

        print("The number of walls counted using the for loop is: ", num_walls)
###


###
    def count_walls_while(self):
        # TODO: QUESTION 3.2
        map_Arr = self.get_maze(self.maze_choice)
        num_walls = 0

        print("The number of walls counted using the while loop is: ", num_walls)
###


###
    def find_goal(self):
        # TODO: QUESTION 3.3
        map_Arr = self.get_maze(self.maze_choice)
        goal_position = 0

        print("The goal is at position: ", goal_position)
###
```

Figure 3: An example of the code scaffold

5 TESTING

The intention was to have first year students and staff test this system and give us their feedback, but due to the Coronavirus pandemic only staff members were recruited to do the testing.

An issue was encountered before this testing process began. It was realised that these staff members may not have access to a



3.1 In the **Maze class**, scroll right to the bottom. You will see the **count_walls_for()** method (**count_walls_for(self):** ...) In this method, use a for loop to count the number of walls on the map and display it to the console.

You can also rewrite this for loop you have just coded as a while loop.

3.2 In the **Maze class**, scroll right to the bottom. You will see the **count_walls_while()** method (**count_walls_while(self):** ...) In this method, use a while loop to count the number of walls on the map and display it to the console.

Now say for instance we want to know where exactly the goal state is in the map array. If we were to use a for loop and run through the whole array to find it, we will just be wasting time as the goal state can be anywhere and we would have to run through the whole array each and every time. Instead we can just use a while loop to loop through the array, until we find the goal state.

3.3 In the **Maze class**, scroll right to the bottom. You will see the **find_goal()** method (**find_goal (self):** ...) In this method, use a while loop to find where the goal is in the map and display it to the console. (*Where the value of map_Arr[...] is equal to 2.*) You do not need to worry about there being multiple goal states, just find the first one.

Figure 4: An example of a question from the assignment document

Raspberry Pi with a SenseHAT installed, so a desktop version of the game had to be created, which can be run on any computer by using the keyboards direction buttons as input.

The testing process essentially requires the users to complete the assignment and the survey in the allotted time which was 3 hours. They would first have to complete the assignment, then the survey afterwards. There was only one round of testing due to the time constraint which we had. This ran through 21 September 2020 to 27 September 2020.

The testing process required the users to first ensure that their system does have Python and PyGame installed before the testing began. The test sessions were 3 hour long sessions, where the user would be given the assignment document and the code scaffold and were asked to complete the game and to answer the survey after completing the game. 2.5 hours were allocated to the completion of the assignment whilst the other 30 minutes were allocated to completing the survey. Figure 5 shows what the user would see when running the game.

The survey used contained 4 Likert scale questions and a general comments and suggestions section. The Likert scale questions aimed to find out whether the questions in the assignment document are clear, if they could easily find the code section where the testers are supposed to write the code, whether the questions were too difficult or easy and some general questions relating to the assignment, game and running the assignment as a whole were of an acceptable standard for the intended users, the students. The comments and suggestions section was used to determine where improvements can be made to the code and the assignment document such that everything is clear, readable and understandable.

6 RESULTS

The first part of the survey dealt with the clarity of the questions asked in the assignment document. The general response was that the questions are clear on the basis that a first year Computer

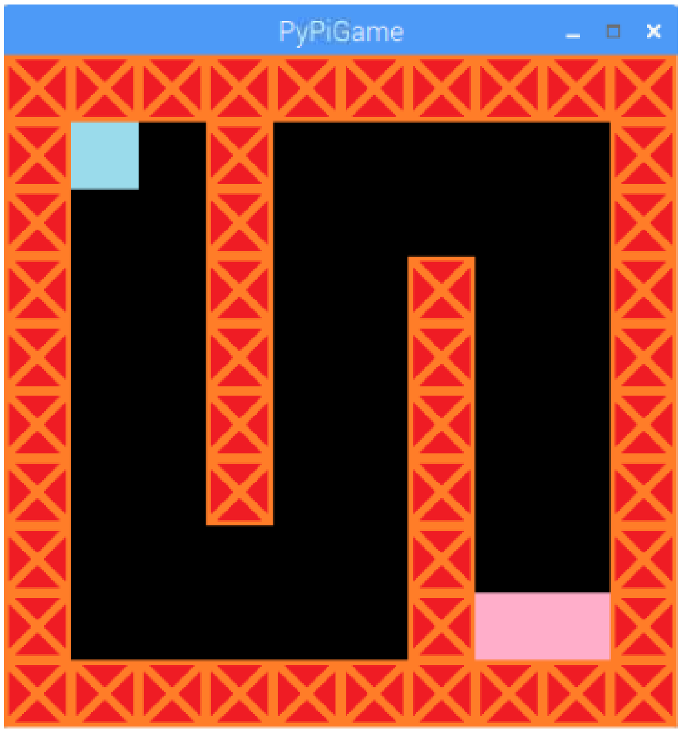


Figure 5: An example of the game running

Science student would be able to understand the question being asked.

Figure 6 shows the responses are recorded for the question: On a Scale of 1-5 (1-Very Unclear, 2- Somewhat Unclear, 3-Neutral, 4- Somewhat Clear, 5 Very Clear) How clear and unambiguous are the questions of the PyPiGame assignment?

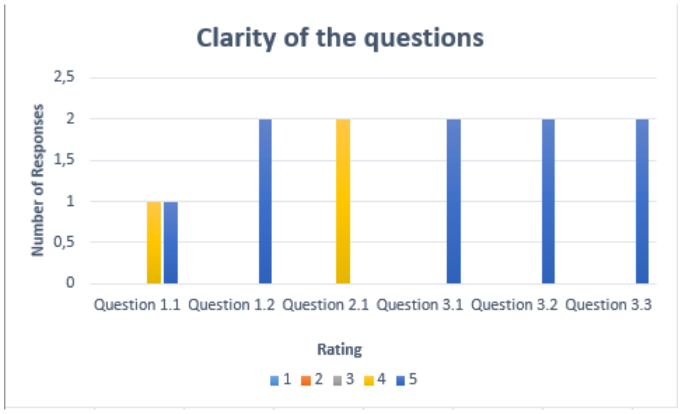


Figure 6: Bar Graph showing responses to the clarity question

The only concerns raised by the testers were:

- In question 1.1 there was a conflict of where to place the `maze_choice` variable as it would make sense to place it in the initialisation constructor of the class.
- In question 2.1 there was confusion as to that the question needs more context.

The second part of the survey dealt with the ability to easily find where the section of code needs to be implemented. The general response was that the questions are easy to find in the code.

Figure 7 shows the responses are recorded for the question: On a Scale of 1-5 (1-Very Challenging, 2- Somewhat Challenging, 3- Neutral, 4- Somewhat Easy, 5 Very Easy) How easily were you able to find each question in the code file?

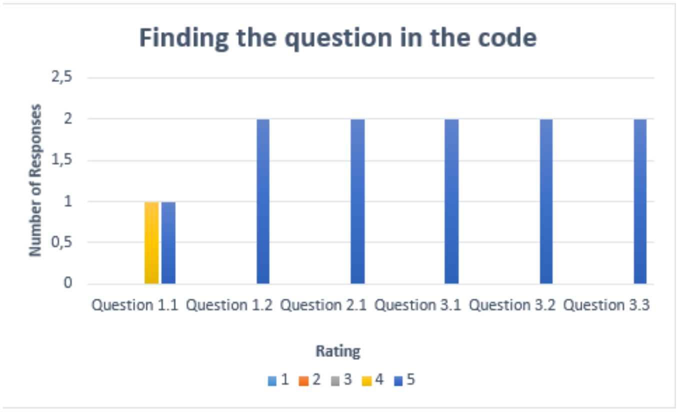


Figure 7: Bar Graph showing responses to the easiness of finding the question in the code scaffold

The testers found that the 'TO DO' comments helped made it easier to find the section of the code where they needed to complete the question as well as that the description of where to find the 'TO DO' comment was very helpful.

The third part of the survey dealt with the difficulty of the questions which the testers had to code.

Figure 8 shows the responses are recorded for the question: On a Scale of 1-5 (1-Very Challenging, 2- Somewhat Challenging, 3-Neutral, 4- Somewhat Easy, 5 Very Easy) How easy were the questions to code?

The testers found that both questions 2.1 and 3.3 would be the most challenging for the students, but it is not too difficult for first year Computer Science to complete.

Among some of the other questions it was noted that there should be more comments in the code even for methods which do not need to be changed in case the student might want to use something in the file. The results from the general questions revealed that:

- the code structure is clear enough for a student to understand it
- the assignment executes at an acceptable standard (does not lag while running)
- the testers did not encounter a bug in the assignment
- there were some problems that they ran into with the assignment but those were sorted out

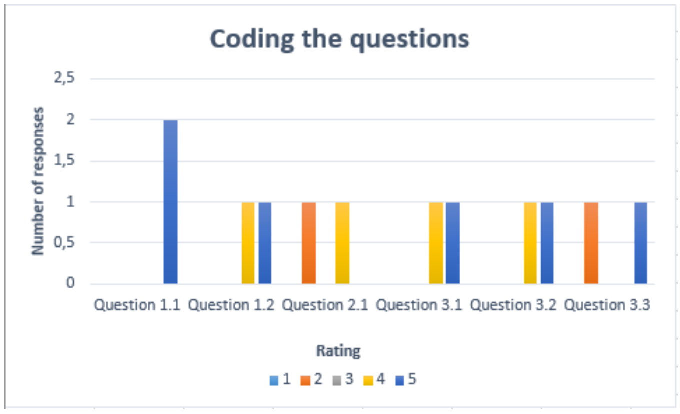


Figure 8: Bar Graph showing responses to the easiness of coding the question

- the testers think that using the standard input (keyboard directional buttons) would work better than the joystick on the RPi's SenseHAT

7 DISCUSSION

From the results, a positive response is drawn in that the game testing went well and that no bugs were encountered in the test phase. That does not mean that there are no bugs in the system itself.

Seeing that the code structure is clear it tells us that a student will be able to read this code easily and this will help them to understand how to structure their code. Due to the this, it can be said that the student could possibly learn more things that they might not know how to do just by looking and running the code. The comments made by the testers to add more comments in the code is a good idea as the students who look at this code might also be looking at different ways to write a section of code in future (i.e How to build classes, methods, etc.). They can learn a lot by looking at the code given to them and could even adopt good programming practices early on in their programming course.

The most positive result to take away from the results would be that it was found that the questions are clear and that the standard of these questions asked vary from easy to challenging. This is good, because we do not want to have an assignment where the students would just easily blast through the assignment and finish it quickly without learning anything or learning something minimal. The students need to spend a good amount of time on this assignment so that they will be able to work through these challenging questions in order to learn something so that they will be able to work on similar challenging questions in future with ease.

Overall the assignment document which was used, did cause a minor confusion with question 2.1 where the testers had to check whether the player was within the boundary lines and asked us to further clarify the question. Other than that the testers found that the assignment document was clear and did help with locating the questions easily which saved them lots of time which they would have used to scroll through the code.

The first research question 'Could creating usable 2D game assignments allow students to effectively learn course materials?' was answered in the sense that yes, we created the PyPiGame assignment and this assignment can help in learning the course materials effectively. However, the second research question: 'Could game-based assignments requiring an external input device be a motivating factor for programming students?' could not be answered as a desktop version of the assignment had to be created as the testers did not have immediate access to a RPi with the SenseHAT add-on board installed.

Lastly a few changes needed to be made to both the memorandum code and the code scaffold to incorporate some of the points raised by the testers in that the `maze_choice` might need to be moved into the `__init__(self)` method as it made more sense to do it this way and to add more comments in the code so that if any student wants to read through the other code which does not need to be modified, they will be able to understand it and might even be able to implement that code in future.

8 CONCLUSIONS

Programming in Computer Science can be difficult if it the students are not motivated or engaged in the learning environment. As such, when reviewing the literature two methods were discovered to help the students to learn to code more effectively and to learn the various programming constructs in a introduction to programming course. This was the inclusion of games in the curriculum (creating game-based assignments) and to use micro-controllers and single-board computers such as the Arduino and the RPi to teach these various programming constructs.

The intention was set out to create a system which combined both these aspects the game-based assignment and the use of the Raspberry Pi with the SenseHAT add-on board. The PyPiGame assignment was created. PyPiGame is an assignment testing the use of loops, conditional statements and variables incorporated in a maze based game where the user needs to move the player around to get to the goal state. This initially ran on the RPi as a game using the PyGame engine.

Testing was conducted with staff from the University of Cape Town (UCT). Before testing began we had realised that the testers would possibly not have access to a RPi with a SenseHAT installed and because of COVID-19 regulations, it was decided that it was best to do the testing virtually. Then a desktop version of the assignment (which would run on any desktop operating system) had to be created and test the system using the desktop version. This in turn meant the second research question: 'Could game-based assignments requiring an external input device be a motivating factor for programming students?' could not be answered.

The results from the testing revealed that the system was good, the questions were clear enough for first year students to understand them and the standard of the coding of the questions ranged from very easy to challenging. After a few tweaks, corrections and addition of more comments, we fully completed the PyPiGame assignment framework which is available as a desktop application or a RPi application. We were able to positively answer our first research question. This means that the assignment is on a level which can be used in a first year programming course to teach these students.

9 FUTURE WORK

The next steps to using this system would be to incorporate the assignment into a first year Computer Science course and to get responses from the students as to how they found the assignment. The system can also be retested with the RPi to find out whether it is better to integrate it with the RPi, to leave it alone as a stand alone desktop assignment or even to use both methods (running on the desktop and running on the RPi) as the only code which will need to be modified is a separate class to the class which runs the game on the desktop or RPi.

By doing this, we can detect any problems with the assignment which was not picked up in this project and can possibly answer the second research question 'Could game-based assignments requiring an external input device be a motivating factor for programming students?'.

10 ACKNOWLEDGEMENTS

The author of this paper would like to thank Gary Stewart, for the help received and his supervision over this project as well as the staff from The University of Cape Town (UCT) who took part in testing the system.

REFERENCES

[1] Oguz Ak and Birgul Kutlu. 2017. Comparing 2D and 3D game-based learning environments in terms of learning gains and student perceptions. *British Journal of Educational Technology* 48, 1 (2017), 129–144. https://doi.org/10.1111/bjet.12346

[2] VH Allan and MV Kolesar. 1997. Teaching computer science: a problem solving approach that works. *ACM SIGCUE Outlook* 25, 1-2 (1997), 2–10. https://doi.org/10.1145/274375.274376

[3] Stephen Cooper, Wanda Dann, Randy Pausch, and Randy Pausch. 2000. Alice: a 3-D tool for introductory programming concepts. In *Journal of computing sciences in colleges*, Vol. 15. Consortium for Computing Sciences in Colleges, 107–116. https://dl.acm.org/doi/pdf/10.5555/364133.364161?download=true

[4] Samuel B Fee and Amanda M Holland-Minkley. 2010. Teaching computer science through problems, not solutions. *Computer Science Education* 20, 2 (2010), 129–144. https://doi.org/10.1080/08993408.2010.486271

[5] Maria Feldgen and Osvaldo Clúa. 2004. Games as a motivation for freshman students learn programming. In *34th Annual Frontiers in Education, 2004. FIE 2004.* IEEE, S1H–11. https://doi.org/10.1109/FIE.2004.1408712

[6] Raspberry Pi Foundation. 2020. *Sense HAT.* Retrieved Jun 4, 2020 from https://www.raspberrypi.org/products/sense-hat/

[7] Raspberry Pi Foundation. 2020. *What is a Raspberry Pi?* Retrieved Apr 24, 2020 from https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/

[8] Anabela Gomes and António José Mendes. 2007. An environment to improve programming education. In *Proceedings of the 2007 international conference on Computer systems and technologies.* 1–6. https://doi.org/10.1145/1330598.1330691

[9] Mike Goslin and Mark R Mine. 2004. The Panda3D graphics engine. *Computer* 37, 10 (2004), 112–114. https://doi.org/10.1109/MC.2004.180

[10] Michael Kölling. 2016. Educational programming on the Raspberry Pi. *Electronics* 5, 3 (2016), 33. https://doi.org/10.3390/electronics5030033

[11] Scott Leutenegger and Jeffrey Edgington. 2007. A games first approach to teaching introductory programming. In *Proceedings of the 38th SIGCSE technical symposium on Computer science education.* 115–118. https://doi.org/10.1145/1227310.1227352

[12] Carnegie Mellon University. 2020. *Panda3D–Free 3D game engine.* Retrieved May 10, 2020 from http://www.panda3d.org/

[13] Dincer Ozoran, N Cagiltay, and Damla Topalli. 2012. Using scratch in introduction to programming course for engineering students. In *2nd International Engineering Education Conference (IEEC2012)*, Vol. 2. 125–132.

[14] Rafaela V Rocha, Rodrigo V Rocha, and Regina B Araújo. 2010. Selecting the best open source 3D games engines. In *Proceedings of the brazilian Symposium on Games and Digital Entertainment, Florianópolis, Santa Catarina, Brazil.* https://pdfs.semanticscholar.org/40ca/2f43ea66fde72040cce18dfed2718b7a45ba.pdf

[15] Miguel A Rubio, Carolina Mañoso Hierro, and APDM Pablo. 2013. Using arduino to enhance computer programming courses in science and engineering. In *Proceedings of EDULEARN13 conference.* IATED Barcelona, Spain, 1–3. https://pdfs.semanticscholar.org/c722/2f0f4b60735ac62bafd9fe17312657983526.pdf

[16] Damla Topalli and Nergiz Ercil Cagiltay. 2018. Improving programming skills in engineering education through problem-based game projects with Scratch. *Computers & Education* 120 (2018), 64–74. https://doi.org/10.1016/j.compedu.2018.01.011

[17] Xiaoyang Zhong and Yao Liang. 2016. Raspberry Pi: an effective vehicle in teaching the internet of things in computer science and engineering. *Electronics* 5, 3 (2016), 56. https://doi.org/10.3390/electronics5030056

A PYPIGAME ASSIGNMENT DOCUMENT

PyPiGame Assignment

In this assignment you will complete the maze game PyPiGame. You will use various programming constructs in the Python programming language.

NOTE: You will **<u>ONLY</u>** be working on the **Maze class** in the **maze.py** file

Please ensure that you have the pygame library installed on your system. Visit https://www.pygame.org/wiki/GettingStarted or https://cs.hofstra.edu/docs/pages/guides/InstallingPygame.html for more instructions.

If you cannot find a particular question look for the following lines in the code:

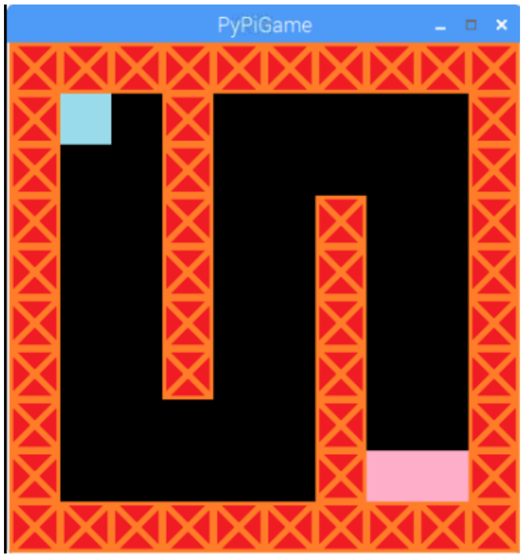
```
###
        # TO DO: QUESTION …
###
```

Try running the assignment. (“python GameApp.py” on the Pi / “python GameAppDesktop.py” on a desktop computer) Do not worry if you get an error that it will not run, this is supposed to happen, we will sort that out in a moment.

<u>Question 1: Variables</u>

1.1 In the **Maze class**, create a variable named **maze_choice** and set it to 1. This variable will be used to set the map which we will play on. Do this in the __init__(self): method. We will use this variable again later.

Try running the assignment now, see it works. You will see a map that sort of looks like this:



Now that we have sorted that issue out, we can move on to displaying a new map which you will create.

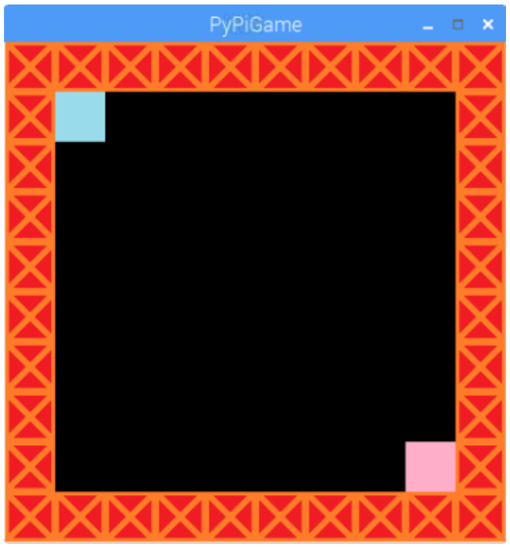
The mazes here is actually a 10x10 grid space stored in a single 1 dimensional array consisting of 100 integers.

1.2 In the same file, scroll down until you can see the **maze_Four(self)** method (**def maze_Four(self):** …) Define a new map using a single dimension array (m = […]). This map should be a clear map with borders and the goal state being in the bottom right corner.
A value of 0 indicates that there will be an open space in that region, a value of 1 indicates a wall will be in that region and a value of 2 indicates that the goal state will be in that region.

NOTE: You can use (copy and paste) any of the mazes defined before this and modify them accordingly to get the desired outcome.

Now set **maze_choice** to 4 and run the program.

The desired map should look like this:



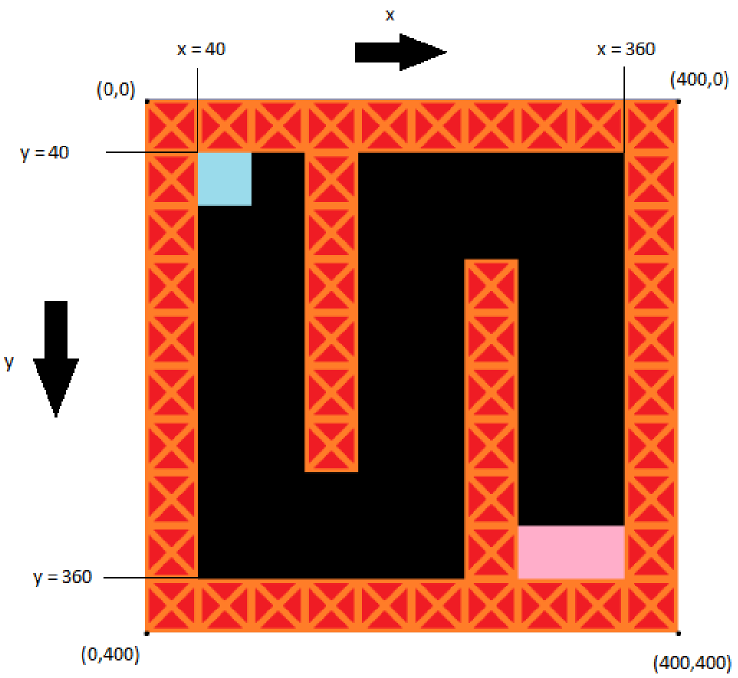
<u>Question 2: Conditional Statements and Boolean expressions</u>

If you move the blue character around using the joystick, you will see that the character will disappear behind the walls instead of 'bouncing' off of them. We will fix that here.

2.1 In the **Maze class**, scroll right to the bottom. You will see the **check_boundaries()** method (**check_boundaries(self, x, y):** …) Check that the x and y values for the player are inside the border using if statements in whichever way you want to (i.e that the player is inside of the border)

Information:

The x-axis runs from left to right and the y-axis runs from top to bottom. Below is an example of the 2D world and where the walls are located. The player will always start at position (40,40). The player is also 40 units in length and width.



Question 3: Loops

This question will demonstrate the concept and importance of loops.

Say for instance we wanted to count the number of walls on the map which we are on (including the borders), we would need something to loop through the entire array while we are counting the walls on the map (the number of 1's in the array). Thankfully we have 2 options for this: the for loop and the while loop.

3.1 In the **Maze class**, scroll right to the bottom. You will see the **count_walls_for()** method (**count_walls_for(self):** …) In this method, use a for loop to count the number of walls on the map and display it to the console.

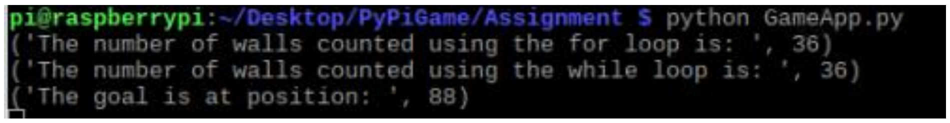
You can also rewrite this for loop you have just coded as a while loop.

3.2 In the **Maze class**, scroll right to the bottom. You will see the **count_walls_while()** method (**count_walls_while(self):** …) In this method, use a while loop to count the number of walls on the map and display it to the console.

Now say for instance we want to know where exactly the goal state is in the map array. If we were to use a for loop and run through the whole array to find it, we will just be wasting time as the goal state can be anywhere and we would have to run through the whole array each and every time. Instead we can just use a while loop to loop through the array, until we find the goal state.

3.3 In the **Maze class**, scroll right to the bottom. You will see the **find_goal()** method (**find_goal (self):** …) In this method, use a while loop to find where the goal is in the map and display it to the console. (*Where the value of map_Arr[…] is equal to 2.*) You do not need to worry about there being multiple goal states, just find the first one.

The output for the question should look similar to this:



```
pi@raspberrypi:~/Desktop/PyPiGame/Assignment $ python GameApp.py
('The number of walls counted using the for loop is: ', 36)
('The number of walls counted using the while loop is: ', 36)
('The goal is at position: ', 88)
```

NOTE: If you reach the goal state and want to play again, you will need to close and re-run the game.