COS10009 – Introduction to Programming

Learning Summary Report

SM RAGIB REZWAN (103172423)

Self-Assessment Details

The following checklists provide an overview of my self-assessment for this unit.

	Pass (D)	Credit (C)	Distinction (B)	High Distinction (A)
Self-Assessment (please tick)				✓

Self-assessment Statement

	Included (please tick)
Learning Summary Report	✓
Test 1 and Test 2 are Compete in Ed	✓
All Pass level tasks completed (including tutorial tasks)	✓

Minimum Pass Checklist

	Included (please tick)
All Credit Tasks are Complete in Ed	✓

Minimum Credit Checklist, in addition to Pass Checklist

	Included (please tick)
Distinction tasks (other than Custom Program) are Complete	✓
Custom program meets Distinction criteria & Interview booked	✓
Design report has structure chart and screenshots of program	✓

Minimum Distinction Checklist, in addition to Credit Checklist

	Included (please tick)
HD Project included	✓
Custom project meets HD requirements	✓

Minimum High Distinction Checklist, in addition to Distinction Checklist

Declaration

I declare that this portfolio is my individual work. I have not copied from any other student's work or from any other source except where due acknowledgment is made explicitly in the text, nor has any part of this submission been written for me by another person.

Signature:_____

Portfolio Overview

This portfolio includes work that demonstrates that I have achieve all Unit Learning Outcomes for COS10009 Unit Title to a **High Distinction** level.

This course helped me understand how to design and code in a structural procedural way in depth, using not only the coding language “ruby”, but also other such similar languages like “C” and “python”. It has accomplished this by teaching me crucial aspects in programming itself, regarding both code and design, via the help of lecture material, reference books, Doubtfire and Ed tasks, help desk sessions and tutorials.

Although it’s a bit difficult to pick out pieces of work that focuses on any single specific Unit Learning Outcomes (as they are all sort of mixed and merged in between the allocated tasks), I will try my best to give you a list of ones that I deem as my best piece of work in each of those category. (giving in order followed in Doubtfire portfolio task)

1) Functional Decomposition: This is basically the ability to break a large problem down in to smaller problems or functions.

   This is beneficial in many cases, like debugging (as we can just test each component or procedure separately instead of entire code at once), reusability (as we can just remove the specific functions we need, instead of taking entire code), understanding of entire code (as we can easily see which small part of code is doing what task, what is being passed and how it is all linked together), etc.

   This has been demonstrated in almost all of the tasks. But the one where I focused on this most was on task 7.1 (where I had to merge information from task 5.1, 5.2, 6.2 and 3.2) and 7.2 (where I had to merge information from task 7.2 with task 3.3 and 4.3). Hence I am keeping those as max level here and graduating down the levels for the rest. (I am not mentioning Foodhunter and maze task as max level here as even though they also have high level of modularisation, these were already set up as such in sample code and the amount I added extra, with regard to modularisation, was very little).

2) Structured Coding principles: There are mainly 4 structured principles:
   a. Sequence: The code is written and executed following a certain order
   b. Repetition: Using for, while, etc. loops to iterate through a block of code until starting condition becomes false.
   c. Selection: Using if, elsif (ruby used this instead of else if), else, case (or switch), etc. to run only a specific case or condition out of the many options given.
   d. Modularisation: Breaking down a code into smaller parts and moving them out of the main code (and putting inside functions and procedures) with the main intention of reusing them in a similar code.

   This is a crucial aspect of structured programming and it is impossible to write a decent code without using any of these (unless the code is a “hello world” type or a poorly written, small code like printing a single line or value). Moreover, in case of big and complicated codes (like in making games, music player, network, etc), where many parts are working in unison and passing data to and from the functions, there is usually combination of all these principles being used there.

   Hence, after contemplating on this for some time, I have decided to give all codes a max rating (except the “hello world” type codes where I am giving a rating of 1) in this regard. That’s because the structured coding principles are like the building block for each and every code and hence I believe that all codes are highly relevant to it.

   I am also giving the task given in concept map as max rating as well as there I had to compare OOP with structural programming (although out of the 4 core principles,

I only focused on modularisation there and the other point was on information hiding which is not part of the core principles)

3) Programming: This basically means being able to code efficiently in the languages covered by utilizing all the techniques and functions provided in it (i.e. if, selection, array, looping, functions and procedures, data type, local and global scoping, classes, parameter, pointers, etc.)

   For this case, I will refer to a famous quote spoken by Henrik Ibsen “A thousand words will not leave so deep an impression as one’s deed”

   So, instead of speaking in detail about each and every one of the codes I have written throughout this semester and submitted on Doubtfire, I believe it will be better to have a glimpse though them and then decide whether I can code or not. (Here I am rating them with respect to type of task they are (T or P or C or D or HD) instead of their relevance as they are all programs and in a sense, they are all equally highly relevant!)

   Furthermore, if you look through it all chronologically (i.e. following the week order) instead of rating order, you will also be able to see how I have grown over time with regards to programming, from the ignorant fool (who couldn’t even input and output variables) to a decent coder (who can analyze and modify codes on the fly).

   [if you are short on time, you can quickly check my growth and understanding by comparing the code I had written at fist in 4.4 and one I wrote for 10.3 and seeing the change I made in the neighbor linking part in order to make sure the code executes as intended.]

.

4) Code reading or debugging: This is an absolutely crucial skill without which a person can’t really call himself to be programmer.

   Its sort like the fact that you need to both read and write to be able to call yourself literate. Till now, all the points we have looked at were mostly focused on writing the code itself and how it looked like. But without being able to understand what you are seeing (i.e. being able to read the code properly), a person will never be able to code programs in depth or help fellow programmers debug their code.

   Moreover, this is a must with respect to career as after joining a company for the first time, you will be following up with a senior programmer where your task will be to go through his code and help him debug it for errors. There if you can’t read code, you can’t properly debug them and hence might even lose the job.

   Technically, most of the tasks submitted on Doubtfire are part of code reading (as there I just had to add or alter a program given in order to make sure it does the desired work and hence provide the desired output). But debugging mainly on took place in tasks 2.1, 9.1, 12.1. So I am giving those specific task as max level rating and others as low level rating.

Since I have been aiming for HD level Band 2 (i.e. 90 to 100), I have completed all the T level, P level, C level, D level and HD level tasks. In case of music player, I have completed up 7.2 (ie the simple GUI one) as Matt had said those who are going for Custom Program and custom project for HD route, do not need to do 7.3 music player task.

Furthermore, as already mentioned, I have done a custom code where I not only used and linked everything I have learned so far from the course, but also used some interesting things

I have found while coding (like being able to restrict button selection to a specific z order level, adding in Enemy ai, creating a custom timer without using Gosu's inbuilt one, etc.). Furthermore, after implementing an Enemy Ai in the code, I had become really curious about the topic, leading me to research on the concept in depth and ended up writing a paper regarding what I have found on it. Thus, I believe I should achieve the HD level 2 band's mark.

Reflection

The most important things I learnt:

Well, truth to be told, I had never actually coded before coming to this course. That's because even though my school had ICT subject, it lacked programming topics in its syllabus and instead prioritized on general knowledge stuff. So everything I learnt here, from basic topics (like reading in values, code syntax, etc.) to the complicated ones (like debugging, 2D arrays, etc.) has been extremely important in my eyes.

But surprisingly, the most important thing I learnt here wasn't actually the key concepts. Instead it was the value of the following sequence of steps: keeping calm in frustrating situations (preferably by taking deep breaths), analysing the questions carefully, noting down possible solutions in copy, testing each and every one of the solutions until desired output has been obtained and cleaning up code and writing comments to make sure I don't fall into the same problem next time.

Although it might seem like a fairly simple sequence of steps, it has been critical in helping me solve the problems I had faced while coding. Thus I am ranking this at the very top.

The things that helped me most were:

1)Lectures (as they guided me though important key concepts and provided hint on tasks)

2) Doubtfire and Ed tasks (as they helped not only reinforce my knowledge but also let me test out what the functions can and can't do)

3) Ruby and Gosu documentations (as they helped me find out important syntax and in built functions and classes)

4)Tutors (as they helped fully explain what task really requires, break simplify it and provide necessary detailed feedback)

5)Helpdesks (as they gave a further opportunity to ask questions and thus solve confusions)

6)Doubtfire's inbuilt deadlines for each tasks (as it helped make sure I finished all my weekly tasks within the week, instead of letting them all pile up until the end)

I found the following topics particularly challenging:

Well, since it was my first time coding, I actually found all of them more or less challenging. But the most challenging ones were:

1)Multi-array (as I didn't understand how they were being set up at first. But then again, it can be because back then I hadn't really understood what a normal array did and hence seeing the code for 4.4 had crashed my brain at that time)

2)Pathfinding task in maze (as I had just come across details on that type of algorithm for the very first time in a different course and thus wasn't able to understand what was going on back then. But thankfully, by the week 10's end, I had been able to understand what the new code was actually supposed to do, modify my previous 4.4 code appropriately and could finally see the task for the simple thing it is, instead of over complicating it)

3) Solving "Rendering Error" in Gosu (as the solution was to separate the image of objects and motion of the object into two different parts which went against my plan of keeping all the information about the object in a single module for later reusability and code-snippet testing)

I found the following topics particularly interesting:

1) Classes (as it provided me with a brand new way of scoping (other than using local and global) which in turn, helped me code with further ease)
2) Recursion (as I had learnt about the concept in my high school mathematics class in induction topic and thus implementing in a code was both nostalgic and fun)

I feel I learnt these topics, concepts, and/or tools really well:

1) Looping with while, begin, for, etc.
2) Passing data into different classes and creating new arrays and objects to store them
3) Selection using if, case, etc.
4) Selecting specific elements in an array, add or removing data at certain parts or to all part of array, keeping count of no of elements in array, passing data by piping
5) Debugging using puts, testing code snippets, using test harness, defensive programming and validation, etc.

There are many more that can be added to the list. But then it would become too long and end up echoing out almost the entire syllabus. So instead I just listed my top 5 here.

I believe the evidence for these and more can be clearly seen in the codes I have submitted as my portfolio via Doubtfire and thus am not elaborating it again here.

I still need to work on the following areas:

Currently I am trying to work out when to use which level of coupling and cohesion and how far I should go in case of modularization in designing my codes.

Furthermore, I am also wondering whether it will be ok or not to use multiple levels of coupling and cohesion throughout in my code, and if so, I am curious of which level of coupling and cohesion I should call my entire code as.

This unit will help me in the future:

I believe this unit will help me make a decision on where I wish to go in life, how far I want to delve in the world of coding and thus, fix my career path. But of most of all, I believe it will work as an encouragement for me, showing me that as long as a I work hard enough, I can not only understand any new concepts but can also master them.

If I did this unit again I would do the following things differently:

1) From day 1, I would look up a table of all possible syntax and logical errors the program can output, what they each mean and how to resolve them. This would have helped me resolve lots of silly mistakes in short time, instead of spending hours and hours trying the modify a part of the code, only to realise in the end that all I am missing is just a bracket or an “end”.
2) Attend all helpdesk from the very first no matter what, not hesitate with my problems and not let embarrassment hold me back from resolving my problems
3) Attend every tutor’s classes (as long as they don’t mind me participating in it) from the very first week and listen to what they say there. That’s because different people have different views and thoughts about the same problem and listening to them can help find new insights and ideas about a problem or topic.
4) Not get too worried if unable to do a task at the first try. Instead, take breaks, think about what’s going wrong and try again and again.

Other:

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

S M Ragib Rezwan

Portfolio Submission

Submitted By:
S M Ragib Rezwan
103172423

Tutor:
NAJAM NAZAR

June 2, 2021

Contents

2 Overall Task Status

Task	Status	Times Assessed
Hello World	Complete	1
Bill Total	Complete	1
Hello User	Complete	1
Debug	Complete	1
Input Functions (Hello User Part 1)	Complete	1
Hospital Charges	Complete	1
Name Tester (Silly Name) - Part 1	Complete	1
Simple Menu	Complete	1
Shape Drawing	Complete	1
Name Tester (Silly Name) - Part 2	Complete	1
File Handling	Complete	1
Gosu Cycle	Complete	1
Shape Moving	Complete	1
Maze Creation	Complete	1
Track File Handling	Complete	1
Music Records	Complete	1
Hover Button	Complete	1
Array Search	Complete	1
Album File Handling	Complete	1
Custom Program Design	Complete	1
Maze Search	Complete	1
Text Music Player with Menu	Complete	1
Simple GUI Music Player	Complete	1
Full GUI Music Player	Not Started	
Concept Map	Complete	1
Food Hunter	Complete	1
Fix-it	Complete	1
Distinction Custom Code	Complete	1
High Distinction Custom Code	Complete	1
Test 1	Complete	1
Recursive Factorial	Complete	1
C Hello World	Complete	1
Python Hello World	Complete	1
Python Shape Moving	Complete	1
Test 2	Complete	1
10.4HD Custom Project	Complete	1
Learning Summary	Complete	1
Python Silly Name Program	Complete	1
C program	Complete	1
Custom Code Video	Complete	1
Custom Project Video	Complete	1
Test Harness	Complete	1

3 Learning Outcomes

3.1 Apply Reading Techniques

Apply code reading and debugging techniques to analyse, interpret, and describe the purpose of program code, and locate within this code errors in syntax, logic, style and/or good practice.

Task	Rating	Status	Times Assessed
Hello World	♦◊◊◊	Complete	1
Bill Total	♦◊◊◊	Complete	1
Debug	♦♦♦♦	Complete	1
Input Functions (Hello User Part 1)	♦◊◊◊	Complete	1
Name Tester (Silly Name) - Part 1	♦◊◊◊	Complete	1
Name Tester (Silly Name) - Part 2	♦◊◊◊	Complete	1
File Handling	♦◊◊◊	Complete	1
Gosu Cycle	♦◊◊◊	Complete	1
Track File Handling	♦◊◊◊	Complete	1
Array Search	♦◊◊◊	Complete	1
Concept Map	♦◊◊◊	Complete	1
Fix-it	♦♦♦♦	Complete	1
C Hello World	♦◊◊◊	Complete	1
Learning Summary	♦♦♦◊	Complete	1
Hello User	♦◊◊◊	Complete	1
Hospital Charges	♦◊◊◊	Complete	1
Simple Menu	♦◊◊◊	Complete	1
Shape Drawing	♦◊◊◊	Complete	1
Shape Moving	♦◊◊◊	Complete	1
Maze Creation	♦◊◊◊	Complete	1
Music Records	♦◊◊◊	Complete	1
Hover Button	♦◊◊◊	Complete	1
Album File Handling	♦◊◊◊	Complete	1
Text Music Player with Menu	♦◊◊◊	Complete	1
Simple GUI Music Player	♦◊◊◊	Complete	1
Food Hunter	♦◊◊◊	Complete	1
Distinction Custom Code	♦♦♦♦	Complete	1
High Distinction Custom Code	♦♦♦♦	Complete	1
Recursive Factorial	♦◊◊◊	Complete	1
Maze Search	♦◊◊◊	Complete	1
Python Silly Name Program	♦◊◊◊	Complete	1
Python Shape Moving	♦◊◊◊	Complete	1
Custom Code Video	♦♦♦♦	Complete	1
Custom Project Video	♦♦♦♦	Complete	1
Test Harness	♦♦♦♦	Complete	1
C program	♦◊◊◊	Complete	1
Custom Program Design	♦♦♦♦	Complete	1
Test 1	♦◊◊◊	Complete	1
Test 2	♦◊◊◊	Complete	1
Python Hello World	♦◊◊◊	Complete	1
10.4HD Custom Project	♦♦♦♦	Complete	1

3.2 Structured Coding Principles

Describe the principles of structured programming, and relate these to the syntactical elements of the programming language used and the way programs are developed.

Task	Rating	Status	Times Assessed
Hello World	♦◊◊◊◊	Complete	1
Bill Total	♦♦♦♦♦	Complete	1
Debug	♦♦♦♦♦	Complete	1
Input Functions (Hello User Part 1)	♦♦♦♦♦	Complete	1
Name Tester (Silly Name) - Part 1	♦♦♦♦♦	Complete	1
Name Tester (Silly Name) - Part 2	♦♦♦♦♦	Complete	1
File Handling	♦♦♦♦♦	Complete	1
Gosu Cycle	♦♦♦♦♦	Complete	1
Track File Handling	♦♦♦♦♦	Complete	1
Array Search	♦♦♦♦♦	Complete	1
Concept Map	♦♦♦♦♦	Complete	1
Fix-it	♦♦♦♦♦	Complete	1
C Hello World	♦◊◊◊◊	Complete	1
Learning Summary	♦♦♦♦♦	Complete	1
Hello User	♦◊◊◊◊	Complete	1
Hospital Charges	♦♦♦♦♦	Complete	1
Simple Menu	♦♦♦♦♦	Complete	1
Shape Drawing	♦♦♦♦♦	Complete	1
Shape Moving	♦♦♦♦♦	Complete	1
Maze Creation	♦♦♦♦♦	Complete	1
Music Records	♦♦♦♦♦	Complete	1
Hover Button	♦♦♦♦♦	Complete	1
Album File Handling	♦♦♦♦♦	Complete	1
Text Music Player with Menu	♦♦♦♦♦	Complete	1
Simple GUI Music Player	♦♦♦♦♦	Complete	1
Food Hunter	♦♦♦♦♦	Complete	1
Distinction Custom Code	♦♦♦♦♦	Complete	1
High Distinction Custom Code	♦♦♦♦♦	Complete	1
Recursive Factorial	♦♦♦♦♦	Complete	1
Maze Search	♦♦♦♦♦	Complete	1
Python Silly Name Program	♦♦♦♦♦	Complete	1
Python Shape Moving	♦♦♦♦♦	Complete	1
Custom Code Video	♦♦♦♦♦	Complete	1
Custom Project Video	♦♦◊◊◊	Complete	1
Test Harness	♦♦♦♦♦	Complete	1
C program	♦♦♦♦♦	Complete	1
Custom Program Design	♦♦♦◊◊	Complete	1
Test 1	♦♦♦♦♦	Complete	1
Test 2	♦♦♦♦♦	Complete	1
Python Hello World	♦♦♦♦♦	Complete	1
10.4HD Custom Project	♦♦◊◊◊	Complete	1

3.3 Programming

Construct small programs, using the programming languages covered, that include the use of arrays, functions and procedures, parameter passing with call by value and call by reference, custom data types and pointers.

Task	Rating	Status	Times Assessed
Hello World	♦◊◊◊	Complete	1
Bill Total	♦◊◊◊	Complete	1
Debug	♦◊◊◊	Complete	1
Input Functions (Hello User Part 1)	♦◊◊◊	Complete	1
Name Tester (Silly Name) - Part 1	♦◊◊◊	Complete	1
Name Tester (Silly Name) - Part 2	♦◊◊◊	Complete	1
File Handling	♦◊◊◊	Complete	1
Gosu Cycle	♦◊◊◊	Complete	1
Track File Handling	♦◊◊◊	Complete	1
Array Search	♦◊◊◊	Complete	1
Concept Map	♦♦◊◊	Complete	1
Fix-it	♦◊◊◊	Complete	1
C Hello World	♦◊◊◊	Complete	1
Learning Summary	♦◊◊◊	Complete	1
Hello User	♦♦◊◊	Complete	1
Hospital Charges	♦♦◊◊	Complete	1
Simple Menu	♦♦◊◊	Complete	1
Shape Drawing	♦♦♦◊	Complete	1
Shape Moving	♦♦♦◊	Complete	1
Maze Creation	♦♦♦♦◊	Complete	1
Music Records	♦♦◊◊	Complete	1
Hover Button	♦♦♦◊	Complete	1
Album File Handling	♦♦◊◊	Complete	1
Text Music Player with Menu	♦♦◊◊	Complete	1
Simple GUI Music Player	♦♦♦◊	Complete	1
Food Hunter	♦♦♦♦◊	Complete	1
Distinction Custom Code	♦♦♦♦♦	Complete	1
High Distinction Custom Code	♦♦♦♦♦	Complete	1
Recursive Factorial	♦♦♦◊	Complete	1
Maze Search	♦♦♦♦♦	Complete	1
Python Silly Name Program	♦♦◊◊	Complete	1
Python Shape Moving	♦♦♦◊	Complete	1
Custom Code Video	♦♦♦♦♦	Complete	1
Custom Project Video	♦♦♦♦◊	Complete	1
Test Harness	♦♦♦◊	Complete	1
C program	♦♦♦♦◊	Complete	1
Custom Program Design	♦♦♦♦◊	Complete	1
Test 1	♦♦♦◊	Complete	1
Test 2	♦♦◊◊	Complete	1
Python Hello World	♦♦◊◊	Complete	1
10.4HD Custom Project	♦♦♦♦◊	Complete	1

3.4 Functional Decomposition

Use modular and functional decomposition to break problems down functionally, represent the resulting structures diagrammatically, and implement these structures in code as functions and procedures.

Task	Rating	Status	Times Assessed
Hello World	♦◊◊◊◊	Complete	1
Bill Total	♦◊◊◊◊	Complete	1
Debug	♦♦♦◊◊	Complete	1
Input Functions (Hello User Part 1)	♦♦◊◊◊	Complete	1
Name Tester (Silly Name) - Part 1	♦♦◊◊◊	Complete	1
Name Tester (Silly Name) - Part 2	♦♦◊◊◊	Complete	1
File Handling	♦♦♦◊◊	Complete	1
Gosu Cycle	♦♦◊◊◊	Complete	1
Track File Handling	♦♦♦◊◊	Complete	1
Array Search	♦♦♦◊◊	Complete	1
Concept Map	♦◊◊◊◊	Complete	1
Fix-it	♦♦♦◊◊	Complete	1
C Hello World	♦♦◊◊◊	Complete	1
Learning Summary	♦♦♦◊◊	Complete	1
Hello User	♦◊◊◊◊	Complete	1
Hospital Charges	♦♦♦◊◊	Complete	1
Simple Menu	♦♦♦◊◊	Complete	1
Shape Drawing	♦♦◊◊◊	Complete	1
Shape Moving	♦♦◊◊◊	Complete	1
Maze Creation	♦♦♦♦◊	Complete	1
Music Records	♦♦♦♦◊	Complete	1
Hover Button	♦♦◊◊◊	Complete	1
Album File Handling	♦♦♦♦◊	Complete	1
Text Music Player with Menu	♦♦♦♦♦	Complete	1
Simple GUI Music Player	♦♦♦♦♦	Complete	1
Food Hunter	♦♦♦♦◊	Complete	1
Distinction Custom Code	♦♦♦♦♦	Complete	1
High Distinction Custom Code	♦♦♦♦♦	Complete	1
Recursive Factorial	♦♦♦◊◊	Complete	1
Maze Search	♦♦♦♦◊	Complete	1
Python Silly Name Program	♦♦◊◊◊	Complete	1
Python Shape Moving	♦◊◊◊◊	Complete	1
Custom Code Video	♦♦♦♦♦	Complete	1
Custom Project Video	♦♦♦◊◊	Complete	1
Test Harness	♦♦◊◊◊	Complete	1
C program	♦♦◊◊◊	Complete	1
Custom Program Design	♦♦◊◊◊	Complete	1
Test 1	♦♦♦♦◊	Complete	1
Test 2	♦♦◊◊◊	Complete	1
Python Hello World	♦◊◊◊◊	Complete	1
10.4HD Custom Project	♦♦♦◊◊	Complete	1

4 Hello World

Run a simple Ruby program

Outcome	Weight
Functional Decomposition	♦◊◊◊◊

Outcome	Weight
Structured Coding Principles	♦◊◊◊◊

Outcome	Weight
Programming	♦◊◊◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

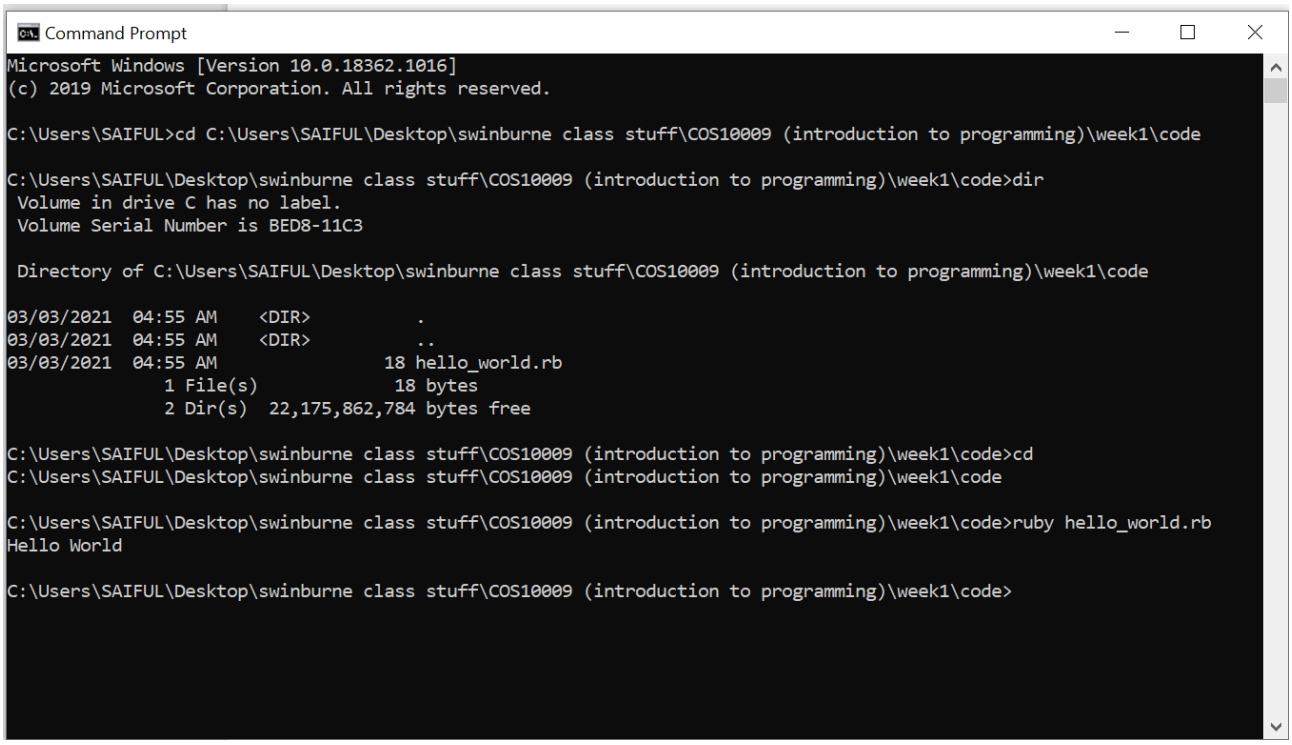
Hello World

Submitted By:
S M Ragib Rezwan
103172423
2021/03/03 10:19

Tutor:
NAJAM NAZAR

March 3, 2021

```ruby
puts 'Hello World'
```


Command Prompt
Microsoft Windows [Version 10.0.18362.1016]
(c) 2019 Microsoft Corporation. All rights reserved.
C:\Users\SAIFUL>cd C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week1\code
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week1\code>dir
Volume in drive C has no label.
Volume Serial Number is BED8-11C3
Directory of C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week1\code
03/03/2021 04:55 AM <DIR> .
03/03/2021 04:55 AM <DIR> ..
03/03/2021 04:55 AM 18 hello_world.rb
1 File(s) 18 bytes
2 Dir(s) 22,175,862,784 bytes free
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week1\code>cd
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week1\code
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week1\code>ruby hello_world.rb
Hello World
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week1\code>

5 Bill Total

A simple Ruby program

Outcome	Weight
Functional Decomposition	♦◇◇◇◇

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦◇◇◇◇

Outcome	Weight
Apply Reading Techniques	♦◇◇◇◇

Date	Author	Comment
2021/03/05 08:41	NAJAM NAZAR	I accept your submission.
2021/03/05 08:43	NAJAM NAZAR	For short answers three you can say assignment and addition.
2021/03/05 09:06	S Rezwan	Oh ok. Thank you!
2021/03/05 09:12	S Rezwan	Ok. Corrected the no 3 question now.

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

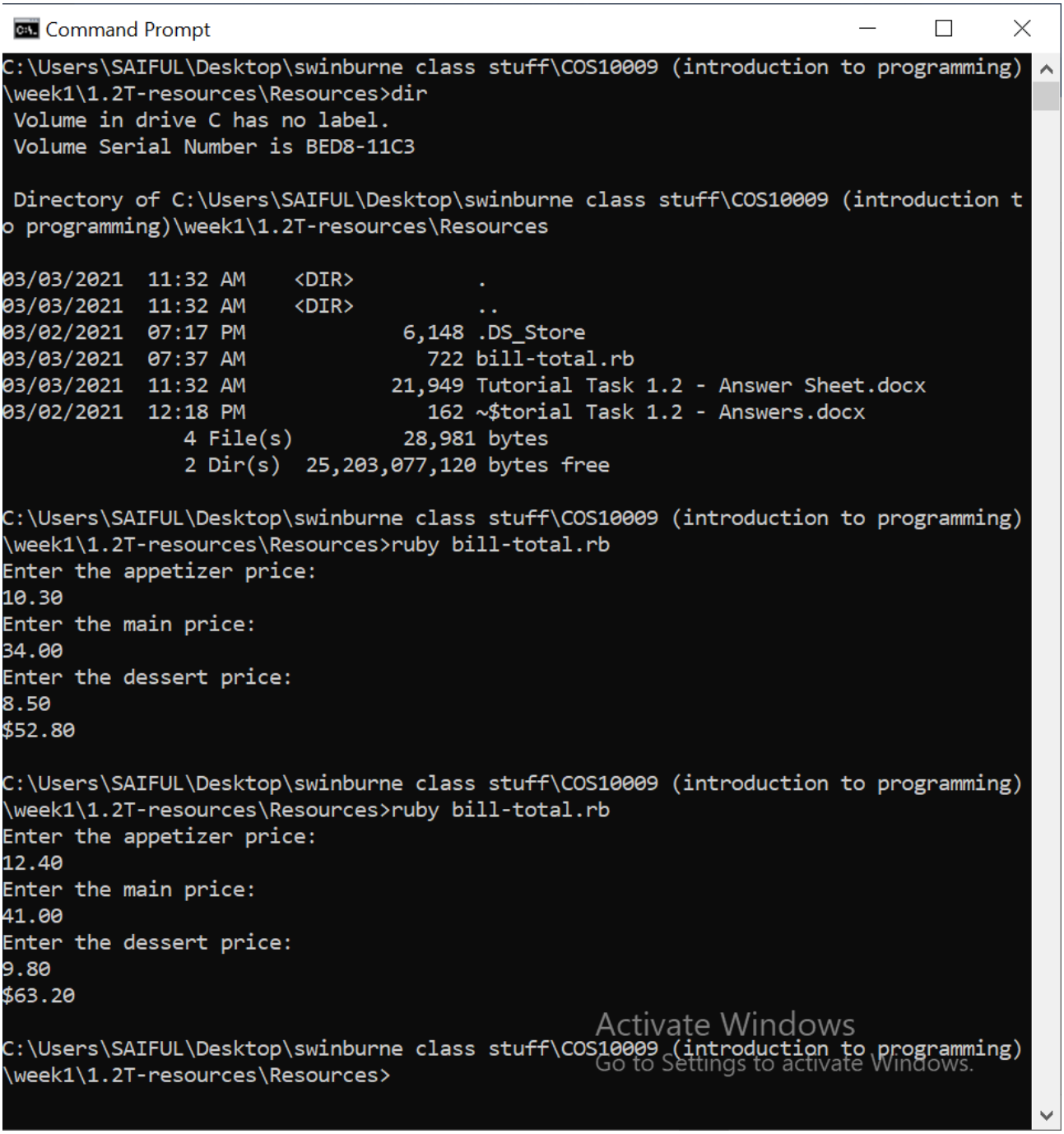
Bill Total

Submitted By:
S M Ragib Rezwan
103172423
2021/03/05 09:11

Tutor:
NAJAM NAZAR

March 5, 2021

```ruby
# Complete the three missing lines of code below

def main()
    puts('Enter the appetizer price:')
    appetizer_price = gets.chomp.to_f()

    # complete the code below using appetizer price
    # above as an example then uncomment the line:
    puts('Enter the main price:')
    main_price = gets.chomp.to_f()

    # complete the code below using appetizer price
    # above and uncomment the line:
    puts('Enter the dessert price:')
    dessert_price = gets.chomp.to_f()

    # Add up the price for the appetizer, main and dessert
    total_price = (appetizer_price + main_price + dessert_price)

    print("$")
    printf("%.2f", total_price) # format the float to 2 decimal places
    print("\n") # print a newline character to move down one line
end

main()
```


Command Prompt
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)
\week1\1.2T-resources\Resources>dir
Volume in drive C has no label.
Volume Serial Number is BED8-11C3
Directory of C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction t
o programming)\week1\1.2T-resources\Resources
03/03/2021 11:32 AM <DIR> .
03/03/2021 11:32 AM <DIR> ..
03/02/2021 07:17 PM 6,148 .DS_Store
03/03/2021 07:37 AM 722 bill-total.rb
03/03/2021 11:32 AM 21,949 Tutorial Task 1.2 - Answer Sheet.docx
03/02/2021 12:18 PM 162 ~$torial Task 1.2 - Answers.docx
4 File(s) 28,981 bytes
2 Dir(s) 25,203,077,120 bytes free
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)
\week1\1.2T-resources\Resources>ruby bill-total.rb
Enter the appetizer price:
10.30
Enter the main price:
34.00
Enter the dessert price:
8.50
$52.80
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)
\week1\1.2T-resources\Resources>ruby bill-total.rb
Enter the appetizer price:
12.40
Enter the main price:
41.00
Enter the dessert price:
9.80
$63.20
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)
\week1\1.2T-resources\Resources>
Activate Windows
Go to Settings to activate Windows.

Answers to Questions from TT1.2

Name:
Student ID:

1. Desk Check Task: Calculate Bill Total

Required Variables:

Integer: appetizer_price, main_price, dessert_price

Real (floating point): total_price

Pseudocode:

Read the value of appetizer_price

Read the value of main_price

Read the value of dessert_price

total_price = appetizer_price + main_price + dessert_price

Print '$' then the value of total_price to the terminal showing two decimal places.

Test Data:

	First data set	Second data set
appetizer_price	10.30	12.40
main_price	34.00	41.00
dessert_price	8.50	9.80

Expected Result:

	First data set	Second data set
Output:	$52.80	$63.20

Desk check - fill this in by completing the missing code in **bill_total.rb** (in the tasks Resources folder) then running it with the test data above:

	Statement	*appetizer_price*	*main_price*	*dessert_price*	*total_price*	*output*
First Pass	*Read the value of appetizer_price*	*10.30*				
	Read the value of main_price		*34.00*			
	Read the value of dessert_price			*8.50*		
	Calculate the total_price				*52.80*	
	Output the unit (dollars)					*$*
	Output the total_price					*52.80*
Second Pass	*Read the value of appetizer_price*	*12.40*				
	Read the value of main_price		*41.00*			
	Read the value of dessert_price			*9.80*		
	Calculate the total_price				*63.20*	
	Output the unit (dollars)					*$*
	Output the total_price					*63.20*

2. Short Answer Questions:

3.

Focus in the following on using the correct computing terminology.

Here are some terms that may help you: Assignment, evaluate, increment,

1. Using a few sentences explain why it may be important to execute statements in the correct sequence. (eg: what might happen if the last statement in Program 2 was executed earlier)

Its needs to go in order or sequence. Otherwise the code will either break or not show the correct answer. (in program 2's case, the last statement was executed earlier, the program would have abruptly ended without evaluating and outputting the total. That's because program does something in each step before moving to the next)

2: The code `main_price = 10` is an example of which kind of programming statement?

This is a variable assignment statement.

3: What **actions** does the computer perform when it executes `a = a + b`?

The computer first assigns the value a
Then it increments the value with b to get a new 'a' value

4: How would the value of variable i change in the statement `i = i + 1`?

The value of i will be incremented by 1.

5: *What sort of types will Ruby use to store the following variables* (given the associated variable values)?

1: *What sort of types will Ruby use to store the following variables* (given the associated variable values)?

Data	Type
Number of students in a class e.g: 23	Integer
Average age of a group of people e.g: 23.5	float
A temperature in Celsius e.g: 45.7	float
True or false e.g: 1 == 2	Boolean

2. *Variables have a scope – what are two different scopes variables can have in Ruby?*

1) local scope

2) global scope

See the lesson materials for help with these questions. You could also see:

https://www.tutorialspoint.com/ruby/ruby_variables.htm

6 Hello User

Hello User

Outcome	Weight
Functional Decomposition	♦◊◊◊◊

Outcome	Weight
Structured Coding Principles	♦◊◊◊◊

Outcome	Weight
Programming	♦♦◊◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

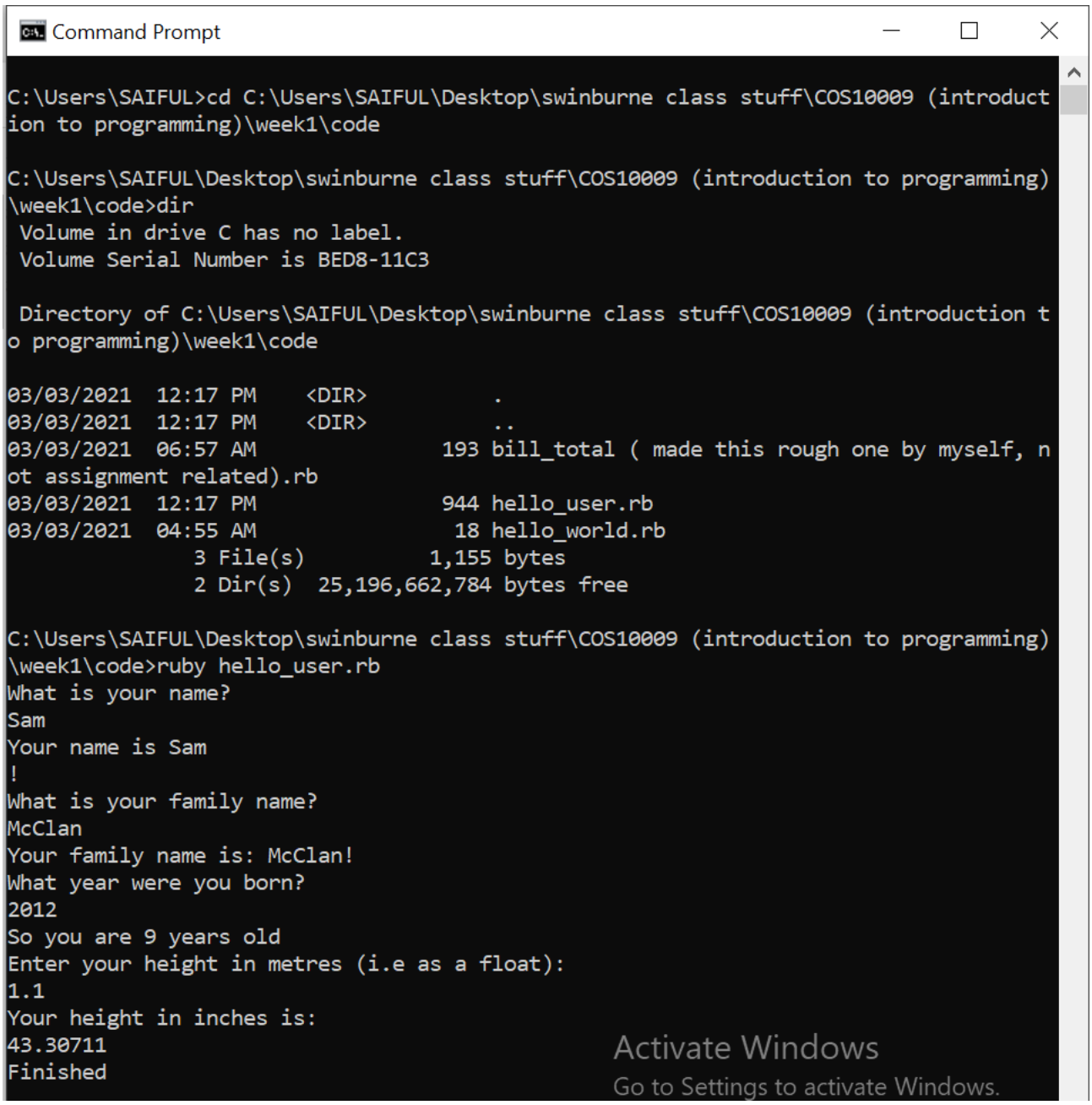
Hello User

Submitted By:
S M Ragib Rezwan
103172423
2021/03/03 17:23

Tutor:
NAJAM NAZAR

March 3, 2021

```ruby
require 'date'

INCHES = 39.3701  # This is a global constant

# Insert the missing code here into the statements below:
# gets
# gets.chomp
# Date.today.year
# year_born.to_i
# gets.to_f

def main
    puts 'What is your name?'
    name = gets
    puts 'Your name is ' + name + '!'
    puts 'What is your family name?'
    family_name = gets.chomp
    puts 'Your family name is: ' + family_name + '!'
    puts 'What year were you born?'
    year_born = gets
    # Calculate the users age
    age = Date.today.year - year_born.to_i
    puts 'So you are ' + age.to_s + ' years old'
    puts 'Enter your height in metres (i.e as a float): '
    value =  gets.to_f

    # Should this be a float or an Integer? Why?
    # it should be a float as height will have decimal values.
    # Also it will be multiplied with inches, a global constant, in float form

    value = value * INCHES
    puts 'Your height in inches is: '
    puts value.to_s
    puts 'Finished'
end

main  # call the main procedure
```


Command Prompt
C:\Users\SAIFUL>cd C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week1\code
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week1\code>dir
Volume in drive C has no label.
Volume Serial Number is BED8-11C3
Directory of C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week1\code
03/03/2021 12:17 PM <DIR> .
03/03/2021 12:17 PM <DIR> ..
03/03/2021 06:57 AM 193 bill_total (made this rough one by myself, not assignment related).rb
03/03/2021 12:17 PM 944 hello_user.rb
03/03/2021 04:55 AM 18 hello_world.rb
3 File(s) 1,155 bytes
2 Dir(s) 25,196,662,784 bytes free
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week1\code>ruby hello_user.rb
What is your name?
Sam
Your name is Sam
!
What is your family name?
McClan
Your family name is: McClan!
What year were you born?
2012
So you are 9 years old
Enter your height in metres (i.e as a float):
1.1
Your height in inches is:
43.30711
Finished
Activate Windows
Go to Settings to activate Windows.

7 Debug

Debug a simple Ruby program

Outcome	Weight
Functional Decomposition	♦♦♦◊◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦◊◊◊◊

Outcome	Weight
Apply Reading Techniques	♦♦♦♦♦

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

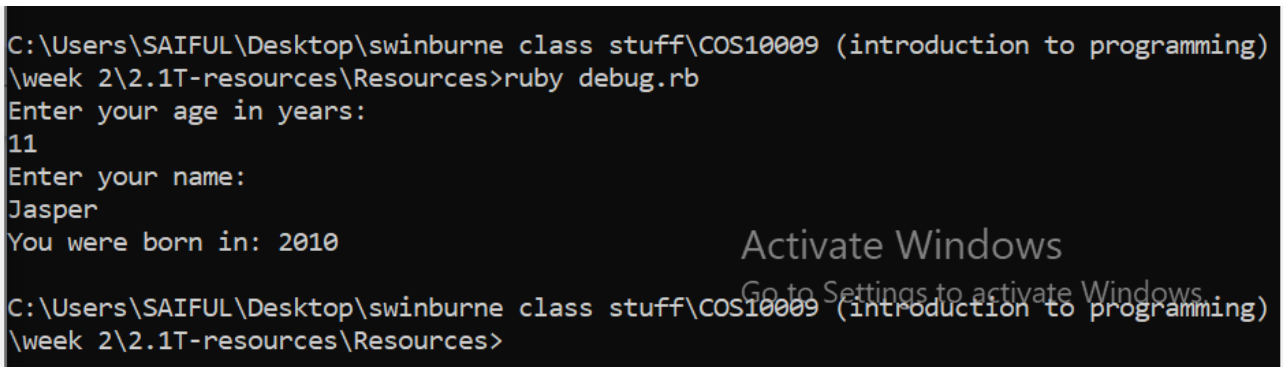
Debug

Submitted By:
S M Ragib Rezwan
103172423
2021/03/06 08:29

Tutor:
NAJAM NAZAR

March 6, 2021

```ruby
require 'date'
$age_in_years

# Fix up the following code that it works and produces the expected output
# in the task specification.

# Asks the user to enter their age and returns an integer age
def get_age()
  puts "Enter your age in years: "
  $age_in_years = gets
  return $age_in_years
end

# takes a prompt and displays it to the user then returns the
# entered string
def get_string()
  puts "Enter your name: "
  s = gets.chomp
  return s
end

# Calculate the year born based on the parameter age and print that out
# along with the name of the user
def print_year_born(age)
  year_born = Date.today.year -  $age_in_years.to_i
  puts "You were born in: " + year_born.to_s
end

def main
  age = get_age()
  name = get_string()
  print_year_born(age)
end

main
```


C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)
\week 2\2.1T-resources\Resources>ruby debug.rb
Enter your age in years:
11
Enter your name:
Jasper
You were born in: 2010
Activate Windows
Go to Settings to activate Windows.
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)
\week 2\2.1T-resources\Resources>

8 Input Functions (Hello User Part 1)

Input and output functions in Ruby

Outcome	Weight
Functional Decomposition	♦♦◊◊◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦◊◊◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Input Functions (Hello User Part 1)

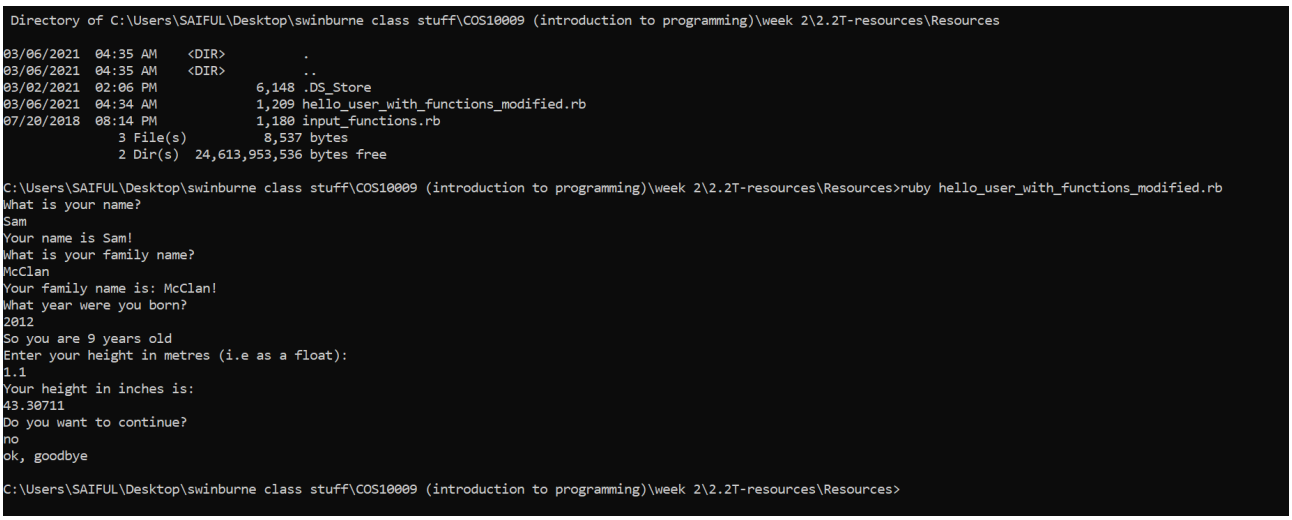
Submitted By:
S M Ragib Rezwan
103172423
2021/03/06 09:38

Tutor:
NAJAM NAZAR

March 6, 2021

```ruby
require 'date'
require './input_functions'

# Multiply metres by the following to get inches:
INCHES = 39.3701

def read_string prompt
    puts prompt
    value = gets.chomp
end

def read_integer prompt
    value = read_string(prompt)
    value.to_i
end

def read_float prompt
    value = read_string(prompt)
    value.to_f
end

def read_boolean prompt
    value = read_string(prompt)
    case value
    when 'y', 'yes', 'Yes', 'YES'
        true
    else
        false
    end
end

# Insert into the following your hello_user code
# from task 1.3P and modify it to use the functions
# in input_functions

def main()

    name = read_string('What is your name?')
    puts 'Your name is ' + name + '!'


    family_name = read_string('What is your family name?')
    puts 'Your family name is: ' + family_name + '!'


    year_born = read_integer('What year were you born?')

    age = Date.today.year.to_i - year_born.to_i
    puts 'So you are ' + age.to_s + ' years old'


    value =  read_float('Enter your height in metres (i.e as a float): ')

```

```
54      value = value * INCHES
55      puts 'Your height in inches is: '
56      puts value.to_s
57
58      continue= read_boolean('Do you want to continue?')
59      if (continue)
60          puts 'ok, lets continue'
61      else
62          puts 'ok, goodbye'
63      end
64
65
66  end
67
68  main()
```


Directory of C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 2\2.2T-resources\Resources
03/06/2021 04:35 AM <DIR> .
03/06/2021 04:35 AM <DIR> ..
03/02/2021 02:06 PM 6,148 .DS_Store
03/06/2021 04:34 AM 1,209 hello_user_with_functions_modified.rb
07/20/2018 08:14 PM 1,180 input_functions.rb
3 File(s) 8,537 bytes
2 Dir(s) 24,613,953,536 bytes free
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 2\2.2T-resources\Resources>ruby hello_user_with_functions_modified.rb
What is your name?
Sam
Your name is Sam!
What is your family name?
McClan
Your family name is: McClan!
What year were you born?
2012
So you are 9 years old
Enter your height in metres (i.e as a float):
1.1
Your height in inches is:
43.30711
Do you want to continue?
no
ok, goodbye
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 2\2.2T-resources\Resources>

9 Hospital Charges

Complete a program with procedures and functions.

Outcome	Weight
Functional Decomposition	♦♦♦◊◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦♦◊◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Hospital Charges

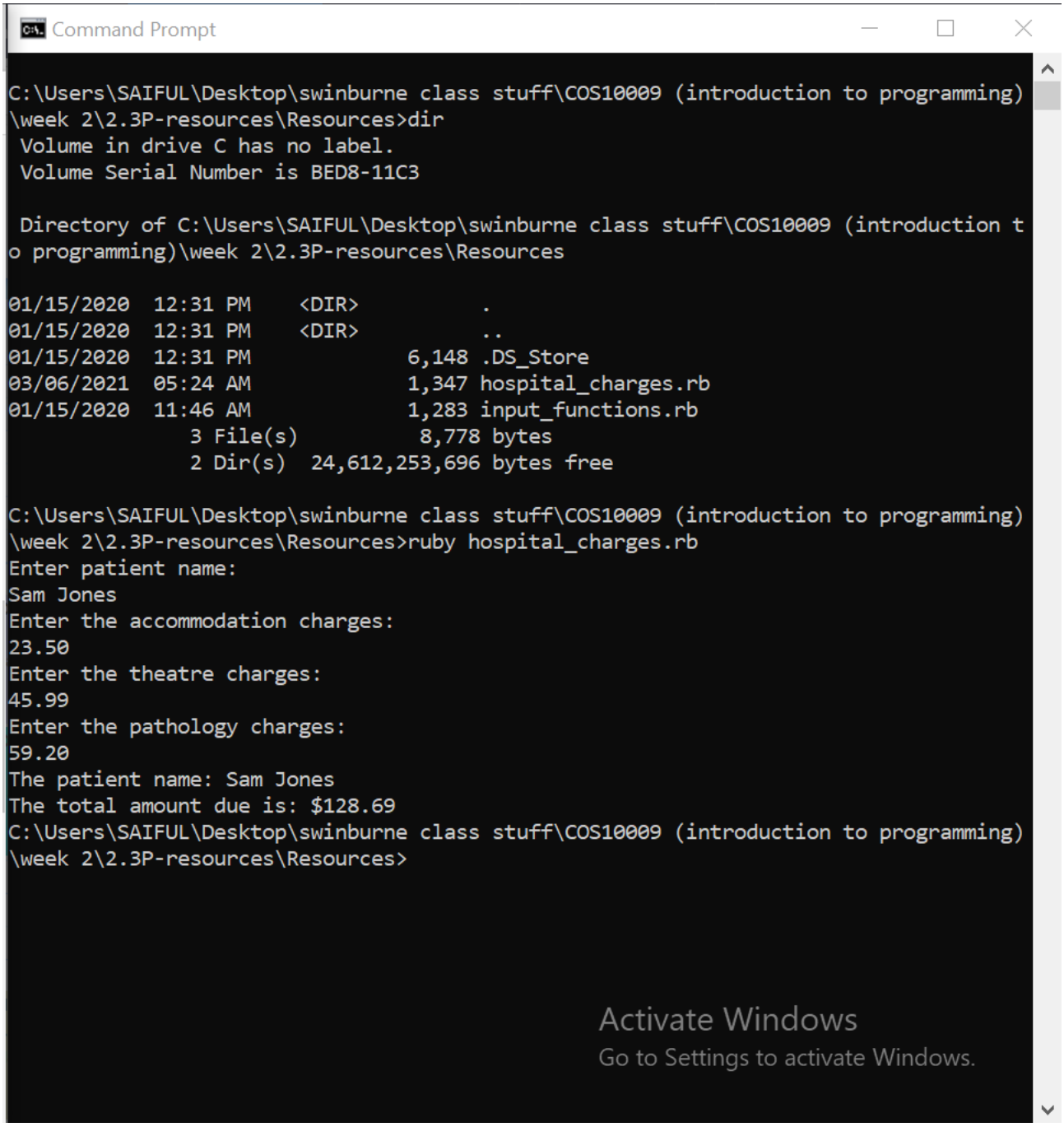
Submitted By:
S M Ragib Rezwan
103172423
2021/03/06 10:27

Tutor:
NAJAM NAZAR

March 6, 2021

```ruby
require './input_functions'
$name


def read_patient_name()
    $name = read_string("Enter patient name: ")
    return $name

    # write this function - use the function read_string(s)
    # from input_functions.rb to read in the name
    # make sure you 'return' the name you read to the calling module
end

def calculate_accommodation_charges()
    charge = read_float("Enter the accommodation charges: ")
    return charge
end

def calculate_theatre_charges()
    charge = read_float("Enter the theatre charges: ")
    return charge
end

def calculate_pathology_charges()
    charge = read_float("Enter the pathology charges: ")
    return charge
    # complete this function based on the above examples
end

def print_patient_bill(name, total)
    puts 'The patient name: ' + $name
    print 'The total amount due is: $'
    bill = print_float(total, 2)
    return


    # write this procedure to print out the patient name
    # and the bill total - use the procedure (from input_functions)
    # print_float(value, decimal_places) to print the total
end

def create_patient_bill()
    total = 0 # it is important to initial variables before use!
    patient_name = read_patient_name()
    total += calculate_accommodation_charges()
    total += calculate_theatre_charges()
    total += calculate_pathology_charges()
    print_patient_bill(patient_name, total)
end

def main()
    create_patient_bill()
end
```

```
54
55 main()
```


Command Prompt
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)
\week 2\2.3P-resources\Resources>dir
Volume in drive C has no label.
Volume Serial Number is BED8-11C3
Directory of C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction t
o programming)\week 2\2.3P-resources\Resources
01/15/2020 12:31 PM <DIR> .
01/15/2020 12:31 PM <DIR> ..
01/15/2020 12:31 PM 6,148 .DS_Store
03/06/2021 05:24 AM 1,347 hospital_charges.rb
01/15/2020 11:46 AM 1,283 input_functions.rb
3 File(s) 8,778 bytes
2 Dir(s) 24,612,253,696 bytes free
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)
\week 2\2.3P-resources\Resources>ruby hospital_charges.rb
Enter patient name:
Sam Jones
Enter the accommodation charges:
23.50
Enter the theatre charges:
45.99
Enter the pathology charges:
59.20
The patient name: Sam Jones
The total amount due is: $128.69
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)
\week 2\2.3P-resources\Resources>
Activate Windows
Go to Settings to activate Windows.

10 Name Tester (Silly Name) - Part 2

A program that uses selection and a loop.

Outcome	Weight
Functional Decomposition	♦♦◊◊◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦◊◊◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

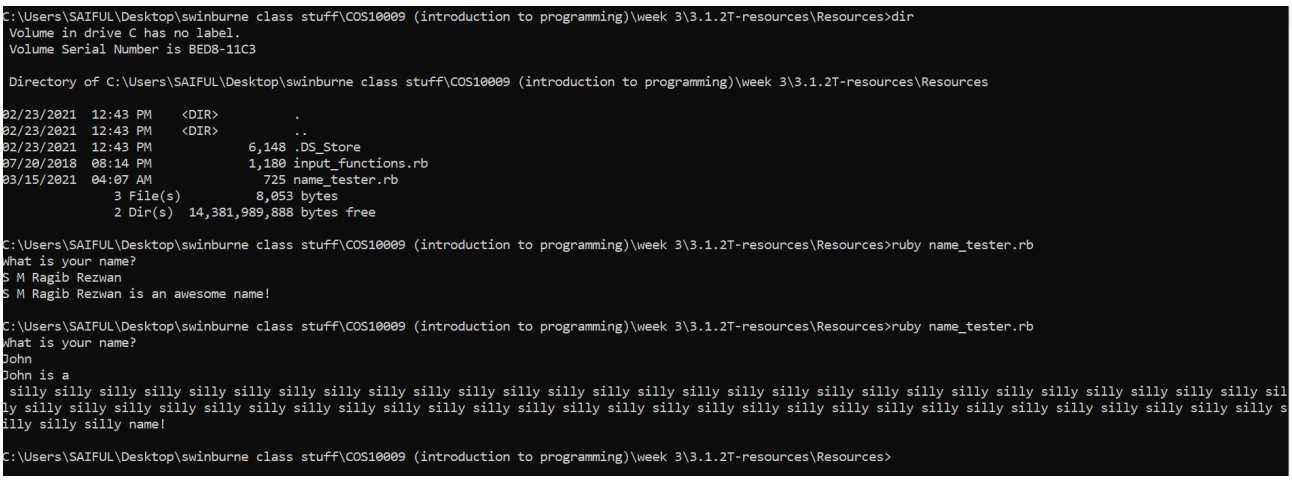
Name Tester (Silly Name) - Part 2

Submitted By:
S M Ragib Rezwan
103172423
2021/03/15 09:09

Tutor:
NAJAM NAZAR

March 15, 2021

```ruby
require './input_functions'

# you need to complete the following procedure that prints out
# "<Name> is a " then print 'silly' (60 times) on one long line
# then print ' name.' \newline

def print_silly_name(name)
    i=0
    puts(name + " is a")

    while(i<60)
        print' silly'
        i=i+1
    end
    puts " name!"
    # complete the code needed here - you will need a loop.
end

def main()
  name = read_string("What is your name?")
  if (name == "S M Ragib Rezwan") or (name == "Najam Nazar")
    puts(name + " is an awesome name!")
  else
    print_silly_name(name)
  end
end
# copy your code from the previous stage below and
# change it to call the procedure above, passing in the name:


# put your main() from stage one here

main()
```


C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 3\3.1.2T-resources\Resources>dir
Volume in drive C has no label.
Volume Serial Number is BED8-11C3
Directory of C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 3\3.1.2T-resources\Resources
02/23/2021 12:43 PM <DIR> .
02/23/2021 12:43 PM <DIR> ..
02/23/2021 12:43 PM 6,148 .DS_Store
07/20/2018 08:14 PM 1,180 input_functions.rb
03/15/2021 04:07 AM 725 name_tester.rb
3 File(s) 8,053 bytes
2 Dir(s) 14,381,989,888 bytes free
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 3\3.1.2T-resources\Resources>ruby name_tester.rb
What is your name?
S M Ragib Rezwan
S M Ragib Rezwan is an awesome name!
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 3\3.1.2T-resources\Resources>ruby name_tester.rb
What is your name?
John
John is a
silly sil
ly silly s
illy silly silly name!
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 3\3.1.2T-resources\Resources>

11 Simple Menu

Create a menu for the Music Player

Outcome	Weight
Functional Decomposition	♦♦♦◊◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦♦◊◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Simple Menu

Submitted By:
S M Ragib Rezwan
103172423
2021/03/19 09:21

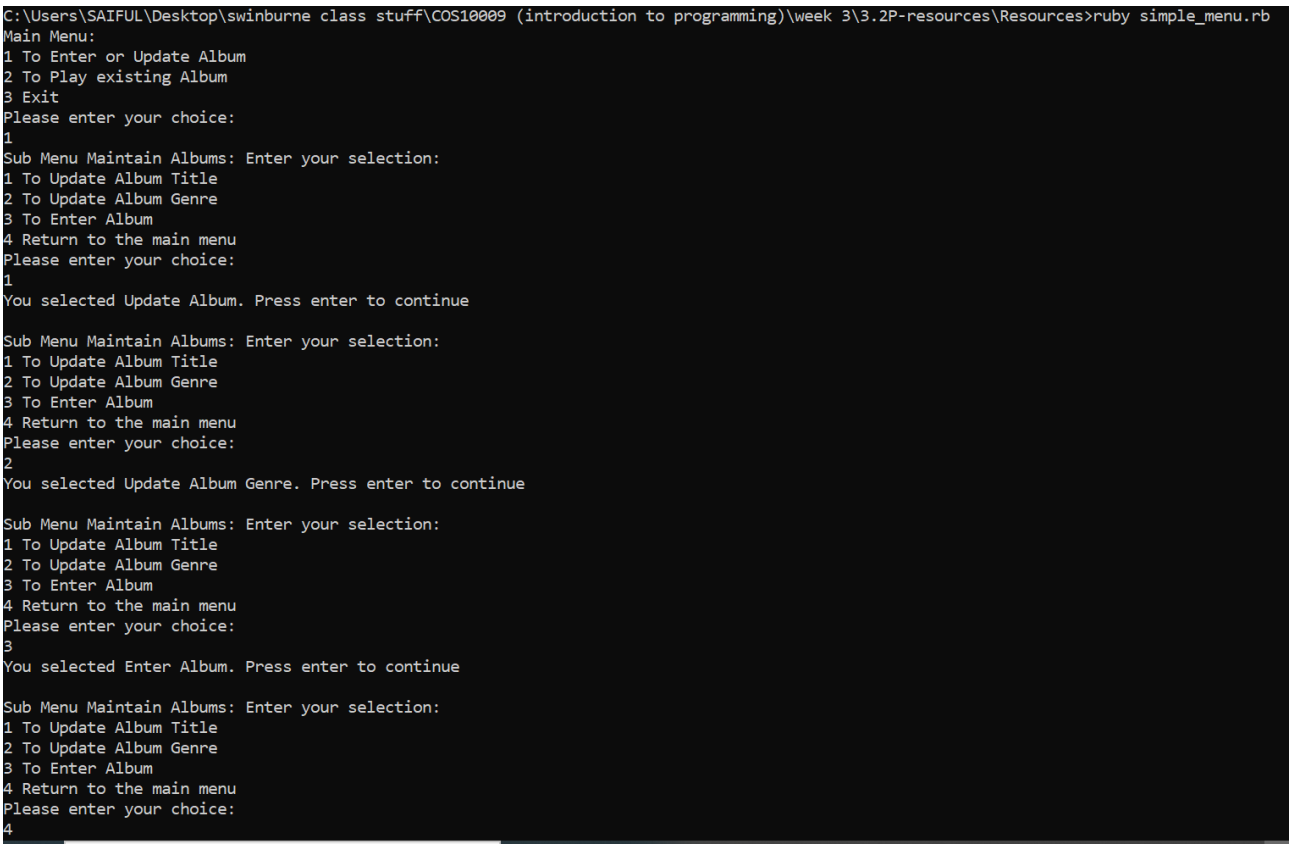
Tutor:
NAJAM NAZAR

March 19, 2021

```ruby
require './input_functions'

def maintain_albums()
    begin
    puts 'Sub Menu Maintain Albums: Enter your selection:'
    puts '1 To Update Album Title'
    puts '2 To Update Album Genre'
    puts '3 To Enter Album'
    puts '4 Return to the main menu '
    choice = read_integer_in_range("Please enter your choice:", 1, 4)
    case choice
    when 1
      Update_Album_Title()
    when 2
      Update_Album_Genre()
    when 3
      Enter_Album()
    when 4
      Return_to_the_main_menu()
    else
      puts 'Please select again'
    end
  end until finished
end

def Update_Album_Title()
  puts "You selected Update Album. Press enter to continue"
  gets

  maintain_albums()
end

def Update_Album_Genre()
  puts "You selected Update Album Genre. Press enter to continue"
  gets

  maintain_albums()
end

def Enter_Album()
  puts "You selected Enter Album. Press enter to continue"
  gets

  maintain_albums()
end

def play_existing_album()
  puts "You selected Play Existing Album. Press enter to continue"
  gets

  Return_to_the_main_menu()
end

```

```ruby
def Return_to_the_main_menu()
  finished = false
  begin
    puts 'Main Menu:'
    puts '1 To Enter or Update Album'
    puts '2 To Play existing Album'
    puts '3 Exit'
    choice = read_integer_in_range("Please enter your choice:", 1, 3)
    case choice
    when 1
      maintain_albums()
    when 2
      play_existing_album()
   when 3
        finished()
    else
      puts 'Please select again'
    end
  end until 3

  #looped until we give 3, then it will stop loop and run 3
  #which says goodbye ---Ragib

end

def finished()
    puts 'goodbye'
end

# complete the case statement below and
# add a stub like the one above for option 2
# of this main menu
def main()
  finished = false
  begin
    puts 'Main Menu:'
    puts '1 To Enter or Update Album'
    puts '2 To Play existing Album'
    puts '3 Exit'
    choice = read_integer_in_range("Please enter your choice:", 1, 3)
    case choice
    when 1
      maintain_albums()
    when 2
      play_existing_album()
    when 3
        finished()
    else
      puts 'Please select again'
    end
  end until 3

  #looped until we give 3, then it will stop loop and run 3
```

```
107   #which says goodbye ---Ragib
108
109 end
110
111 main()
```


C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 3\3.2P-resources\Resources>ruby simple_menu.rb
Main Menu:
1 To Enter or Update Album
2 To Play existing Album
3 Exit
Please enter your choice:
1
Sub Menu Maintain Albums: Enter your selection:
1 To Update Album Title
2 To Update Album Genre
3 To Enter Album
4 Return to the main menu
Please enter your choice:
1
You selected Update Album. Press enter to continue
Sub Menu Maintain Albums: Enter your selection:
1 To Update Album Title
2 To Update Album Genre
3 To Enter Album
4 Return to the main menu
Please enter your choice:
2
You selected Update Album Genre. Press enter to continue
Sub Menu Maintain Albums: Enter your selection:
1 To Update Album Title
2 To Update Album Genre
3 To Enter Album
4 Return to the main menu
Please enter your choice:
3
You selected Enter Album. Press enter to continue
Sub Menu Maintain Albums: Enter your selection:
1 To Update Album Title
2 To Update Album Genre
3 To Enter Album
4 Return to the main menu
Please enter your choice:
4

12 Shape Drawing

Draw a shape using GOSU

Outcome	Weight
Functional Decomposition	♦♦◊◊◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦♦♦◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Shape Drawing

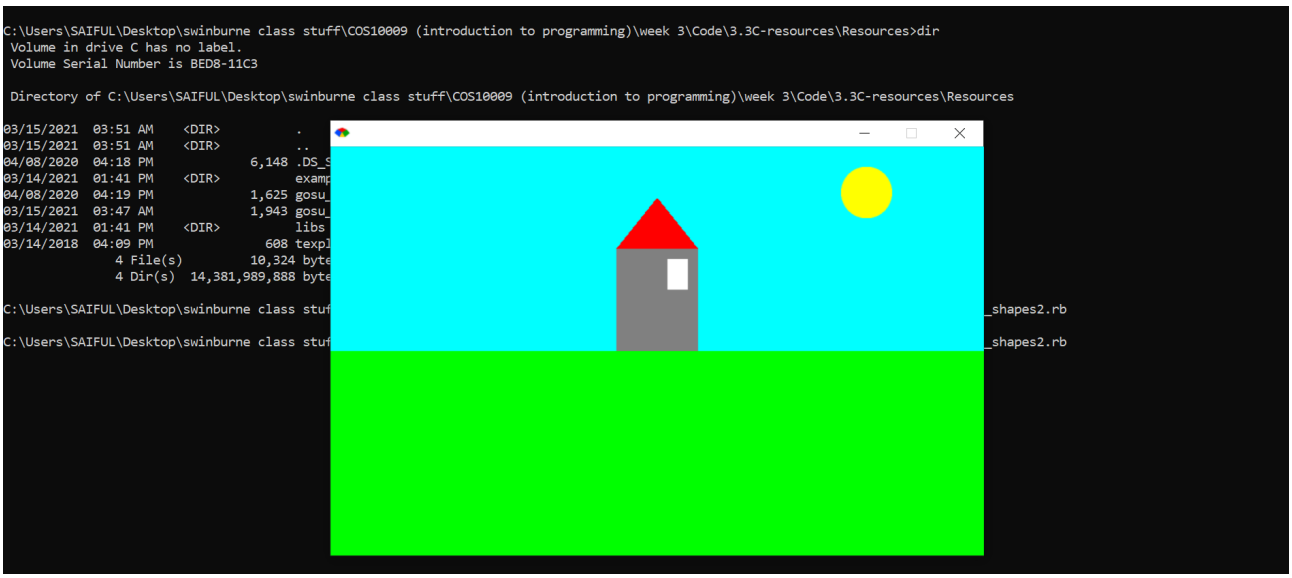
Submitted By:
S M Ragib Rezwan
103172423
2021/03/15 08:55

Tutor:
NAJAM NAZAR

March 15, 2021

```ruby
require 'rubygems'
require 'gosu'
require './libs/circle'

# The screen has layers: Background, middle, top
module ZOrder
  BACKGROUND, MIDDLE, TOP = *0..2
end

class DemoWindow < Gosu::Window
  def initialize
    super(640, 400, false)
  end

  def draw
    # see www.rubydoc.info/github/gosu/gosu/Gosu/Color for colours
    draw_quad(0, 0, 0xff_00ffff, 640, 0, 0xff_00ffff, 0, 400, 0xff_00ffff, 640, 400, 0xff_00ffff, ZOrder::BACKGROUND)
    draw_quad(0, 200, Gosu::Color::GREEN, 0, 400, Gosu::Color::GREEN, 640, 200, Gosu::Color::GREEN, 640, 400, Gosu::Color::GREEN, ZOrder::MIDDLE)
    draw_quad(280, 200, Gosu::Color::GRAY, 360, 200, Gosu::Color::GRAY, 280, 100, Gosu::Color::GRAY, 360, 100, Gosu::Color::GRAY, ZOrder::MIDDLE)
    draw_triangle(280, 100, Gosu::Color::RED, 360, 100, Gosu::Color::RED, 320, 50, Gosu::Color::RED, ZOrder::MIDDLE, mode=:default)
    draw_quad(330, 140, Gosu::Color::WHITE, 350, 140, Gosu::Color::WHITE, 330,110, Gosu::Color::WHITE, 350,110, Gosu::Color::WHITE, ZOrder::MIDDLE)


    #draw_line(340, 125, Gosu::Color::BLACK, 340, 120, Gosu::Color::BLACK, ZOrder::TOP, mode=:default)
    # draw_rect works a bit differently:
    #Gosu.draw_rect(340, 200, 100, 100, Gosu::Color::GREEN, ZOrder::TOP, mode=:default)

    # Circle parameter - Radius
    img2 = Gosu::Image.new(Circle.new(50))
    img2.draw(500, 20, ZOrder::TOP, 0.5, 0.5, Gosu::Color::YELLOW)
    # Image draw parameters - x, y, z, horizontal scale (use for ovals), vertical scale (use for ovals), colour
    # Colour - use Gosu::Image::{Colour name} or .rgb({red},{green},{blue}) or .rgba({alpha}{red},{green},{blue},)
    # Note - alpha is used for transparency.
    # drawn as an elipse (0.5 width:)
    #img2.draw(200, 200, ZOrder::TOP, 0.5, 1.0, Gosu::Color::BLUE)
    # drawn as a red circle:

    # drawn as a red circle with transparency:
    #img2.draw(300, 250, ZOrder::TOP, 1.0, 1.0, 0x64_ff0000)




  end
```

```
45 end
46
47 DemoWindow.new.show
```


C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 3\Code\3.3C-resources\Resources>dir
Volume in drive C has no label.
Volume Serial Number is BED8-11C3
Directory of C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 3\Code\3.3C-resources\Resources
03/15/2021 03:51 AM <DIR> .
03/15/2021 03:51 AM <DIR> ..
04/08/2020 04:18 PM 6,148 .DS_S
03/14/2021 01:41 PM <DIR> examp
04/08/2020 04:19 PM 1,625 gosu_
03/15/2021 03:47 AM 1,943 gosu_
03/14/2021 01:41 PM <DIR> libs
03/14/2018 04:09 PM 608 texpl
4 File(s) 10,324 byte
4 Dir(s) 14,381,989,888 byte
C:\Users\SAIFUL\Desktop\swinburne class stuf
_shapes2.rb
C:\Users\SAIFUL\Desktop\swinburne class stuf
_shapes2.rb

13 Name Tester (Silly Name) - Part 1

A program to read input and respond

Outcome	Weight
Functional Decomposition	♦♦◊◊◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦◊◊◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

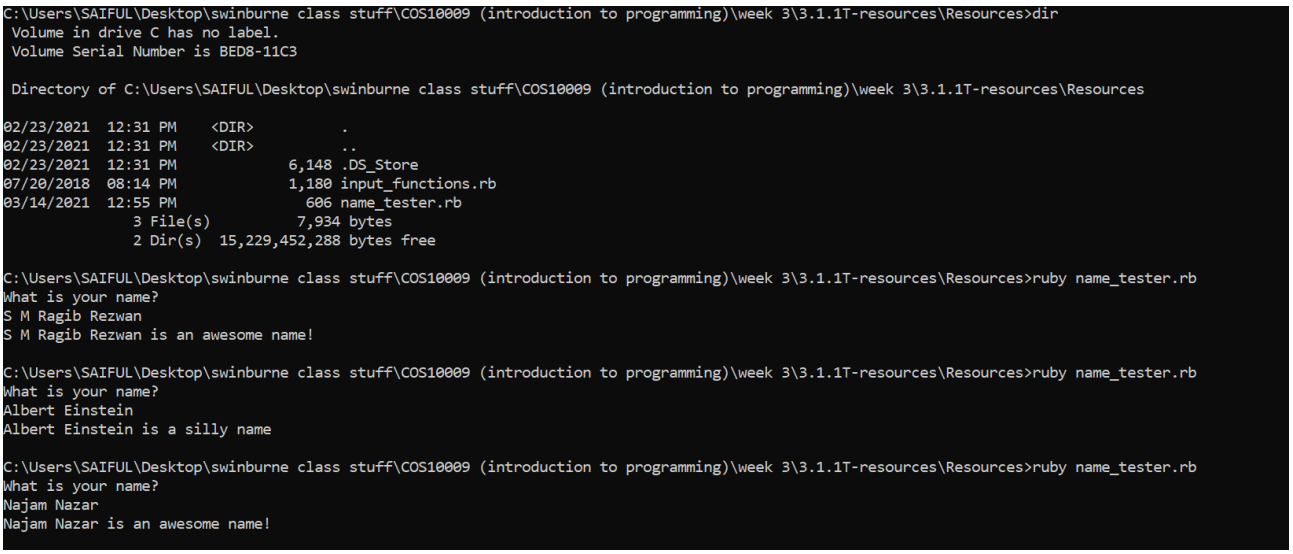
Name Tester (Silly Name) - Part 1

Submitted By:
S M Ragib Rezwan
103172423
2021/03/14 17:59

Tutor:
NAJAM NAZAR

March 14, 2021

```ruby
require './input_functions'

# write code that reads in a user's name from the terminal.
# (use the functions from input_functions.rb) If the name matches
# your name or your tutor's name, then print out "<Name> is an awesome name!"
# Otherwise print out a message that says "<Name> is a silly name"

def print_silly_name(name)
  puts name + " is a " + "silly " + "name!"
end

def main()
  name = read_string("What is your name?")
  if (name == "S M Ragib Rezwan") or (name == "Najam Nazar")
    puts(name + " is an awesome name!")
  else
    puts(name + " is a " + "silly " + "name")
  end
end

main()
```


C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 3\3.1.1T-resources\Resources>dir
Volume in drive C has no label.
Volume Serial Number is BED8-11C3
Directory of C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 3\3.1.1T-resources\Resources
02/23/2021 12:31 PM <DIR> .
02/23/2021 12:31 PM <DIR> ..
02/23/2021 12:31 PM 6,148 .DS_Store
07/20/2018 08:14 PM 1,180 input_functions.rb
03/14/2021 12:55 PM 606 name_tester.rb
3 File(s) 7,934 bytes
2 Dir(s) 15,229,452,288 bytes free
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 3\3.1.1T-resources\Resources>ruby name_tester.rb
What is your name?
S M Ragib Rezwan
S M Ragib Rezwan is an awesome name!
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 3\3.1.1T-resources\Resources>ruby name_tester.rb
What is your name?
Albert Einstein
Albert Einstein is a silly name
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 3\3.1.1T-resources\Resources>ruby name_tester.rb
What is your name?
Najam Nazar
Najam Nazar is an awesome name!

14 File Handling

Reading from a file.

Outcome	Weight
Functional Decomposition	♦♦♦◊◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦◊◊◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

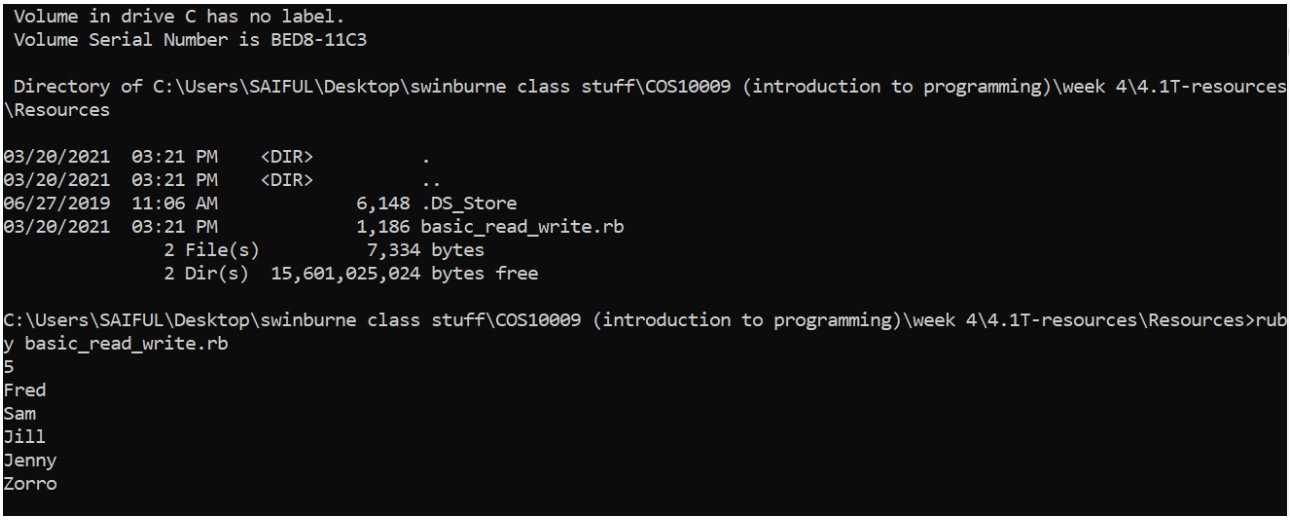
File Handling

Submitted By:
S M Ragib Rezwan
103172423
2021/03/20 20:26

Tutor:
NAJAM NAZAR

March 20, 2021

```ruby
# writes the number of lines then each line as a string.

def write_data_to_file(a_file)
   a_file.puts('5')
   a_file.puts('Fred')
   a_file.puts('Sam')
   a_file.puts('Jill')
   a_file.puts('Jenny')
   a_file.puts('Zorro')
end

# reads in each line.
# you need to change the following code
# so that it uses a loop which repeats
# acccording to the number of lines in the File
# which is given in the first line of the File
def read_data_from_file(a_file)
  count = a_file.gets.to_i
  puts count.to_s
  i = 0
    while(i < count.to_i)
        puts a_file.gets
        i=i+1
    end


end

# writes data to a file then reads it in and prints
# each line as it reads.
# you should improve the modular decomposition of the
# following by moving as many lines of code
# out of main as possible.
def make_file()
 a_file = File.new("mydata.txt", "w") # open for writing
  write_data_to_file(a_file)
  a_file.close
end

def write_from_file()
    a_file = File.new("mydata.txt", "r") # open for reading
  read_data_from_file(a_file)
  a_file.close
end

def main
    make_file()
    write_from_file()


end

main
```



```
 Volume in drive C has no label.
 Volume Serial Number is BED8-11C3

 Directory of C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 4\4.1T-resources
\Resources

03/20/2021  03:21 PM    <DIR>          .
03/20/2021  03:21 PM    <DIR>          ..
06/27/2019  11:06 AM             6,148 .DS_Store
03/20/2021  03:21 PM             1,186 basic_read_write.rb
               2 File(s)          7,334 bytes
               2 Dir(s)  15,601,025,024 bytes free

C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 4\4.1T-resources\Resources>rub
y basic_read_write.rb
5
Fred
Sam
Jill
Jenny
Zorro
```

15 Gosu Cycle

Learn to use GOSU for graphical programming.

Outcome	Weight
Functional Decomposition	♦♦◊◊◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦◊◊◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Gosu Cycle

Submitted By:
S M Ragib Rezwan
103172423
2021/03/20 20:38

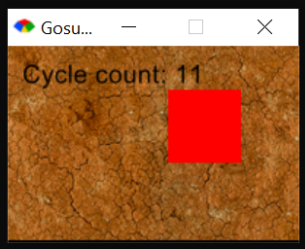
Tutor:
NAJAM NAZAR

March 20, 2021

```ruby
require 'gosu'

module ZOrder
  BACKGROUND, MIDDLE, TOP = *0..2
end


# Instructions:
# Add a shape_x variable in the following (similar to the cycle one)
# that is initialised to zero then incremented by 10 in update.
# Change the draw method to draw a shape (circle or square)
# (50 pixels in width and height) with a y coordinate of 30
# and an x coordinate of shape_x.
class GameWindow < Gosu::Window

  # initialize creates a window with a width an a height
  # and a caption. It also sets up any variables to be used.
  # This is procedure i.e the return value is 'undefined'
  def initialize
  @shape_x = 0
  @shape_y = 30
    super 200, 135, false
    self.caption = "Gosu Cycle Example"

    # Create and load an image to display
    @background_image = Gosu::Image.new("media/earth.png")

    # Create and load a font for drawing text on the screen
    @font = Gosu::Font.new(20)
    @cycle = 0
    puts "0. In initialize\n"
  end

  # Put any work you want done in update
  # This is a procedure i.e the return value is 'undefined'
  def update

  @shape_x +=10
      puts "1. In update. Sleeping for one second\n"
    @cycle += 1 # add one to the current value of cycle
    sleep(1)
  end

  # the following method is called when you press a mouse
  # button or key
  def button_down(id)
    puts "In Button Down " + id.to_s
  end

  # Draw (or Redraw) the window
  # This is procedure i.e the return value is 'undefined'
  def draw
    Gosu.draw_rect(@shape_x, @shape_y, 50, 50, Gosu::Color::RED, ZOrder::TOP,
    ↪  mode=:default)
```

```
54      # Draws an image with an x, y and z
55      #(z determines if it sits on or under other things that are drawn)
56
57      @background_image.draw(0, 0, z = ZOrder::BACKGROUND)
58      @font.draw("Cycle count: #{@cycle}", 10, 10, z = ZOrder::TOP, 1.0, 1.0,
        ↪  Gosu::Color::BLACK)
59      puts "2. In draw\n"
60    end
61  end
62
63  window = GameWindow.new
64  window.show
```

```
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 4\4.2T-resources\Resources>rub
y gosu_cycle_example.rb
0. In initialize
1. In update. Sleeping for one second
DEPRECATION WARNING: draw is deprecated; use Font#draw_text or Font#draw_markup instead.
draw called from gosu_cycle_example.rb:58:in `draw'.
2. In draw
1. In update. Sleeping for one second
2. In draw
1. In update. Sleeping for one second
2. In draw
1. In update. Sleeping for one second
2. In draw
1. In update. Sleeping for one second
2. In draw
1. In update. Sleeping for one second
2. In draw
1. In update. Sleeping for one second
2. In draw
1. In update. Sleeping for one second
2. In draw
1. In update. Sleeping for one second
2. In draw
1. In update. Sleeping for one second
2. In draw
1. In update. Sleeping for one second
```



16 Shape Moving

A program to move a shape in GOSU

Outcome	Weight
Functional Decomposition	♦♦◊◊◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦♦♦◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Shape Moving

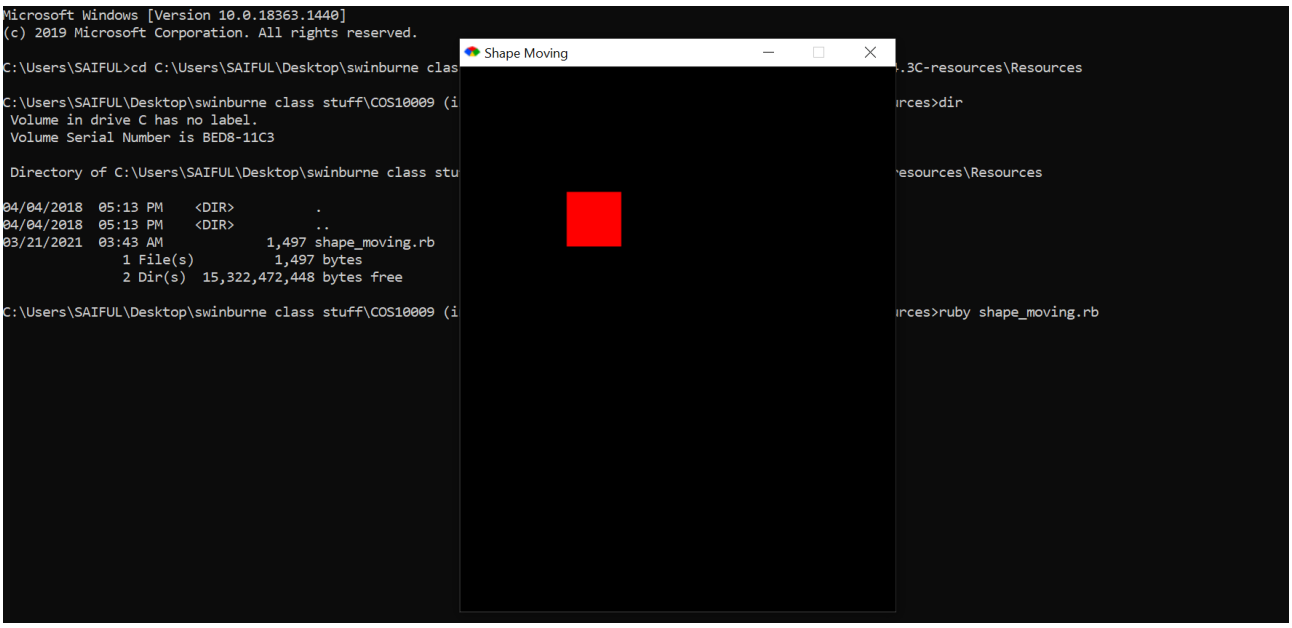
Submitted By:
S M Ragib Rezwan
103172423
2021/03/21 08:46

Tutor:
NAJAM NAZAR

March 21, 2021

```ruby
require 'gosu'

module ZOrder
  BACKGROUND, MIDDLE, TOP = *0..2
end

WIDTH = 400
HEIGHT = 500
SHAPE_DIM = 50

# Instructions:
# Fix the following code so that:
# 1. The shape also can be moved up and down
# 2. the shape does not move out of the window area

class GameWindow < Gosu::Window

  # initialize creates a window with a width an a height
  # and a caption. It also sets up any variables to be used.
  # This is procedure i.e the return value is 'undefined'
  def initialize
    super WIDTH, HEIGHT, false
    self.caption = "Shape Moving"

    @shape_y = HEIGHT / 2
    @shape_x = WIDTH / 2
  end

  # Put any work you want done in update
  # This is a procedure i.e the return value is 'undefined'

#note to self, for some reason != part doesnt restrict properly
#but for some reason > and < signs do. Need to ask Tutor about this
  def update

    if button_down?(Gosu::KbRight) && @shape_x < (WIDTH - SHAPE_DIM)
        @shape_x += 3

    end
    if button_down?(Gosu::KbLeft) && (@shape_x > (0) )
        @shape_x -= 3
    end
    if button_down?(Gosu::KbUp) && (@shape_y > 0)
        @shape_y -= 3

    end
    if button_down?(Gosu::KbDown) && (@shape_y < HEIGHT - SHAPE_DIM)
        @shape_y += 3
    end

  end

  # Draw (or Redraw) the window
```

```
54    # This is procedure i.e the return value is 'undefined'
55    def draw
56      Gosu.draw_rect(@shape_x, @shape_y, SHAPE_DIM, SHAPE_DIM, Gosu::Color::RED,
        ↪  ZOrder::TOP, mode=:default)
57    end
58  end
59
60  window = GameWindow.new
61  window.show
```


Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.
C:\Users\SAIFUL>cd C:\Users\SAIFUL\Desktop\swinburne clas
.3C-resources\Resources
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (i
rces>dir
Volume in drive C has no label.
Volume Serial Number is BED8-11C3
Directory of C:\Users\SAIFUL\Desktop\swinburne class stu
esources\Resources
04/04/2018 05:13 PM <DIR> .
04/04/2018 05:13 PM <DIR> ..
03/21/2021 03:43 AM 1,497 shape_moving.rb
1 File(s) 1,497 bytes
2 Dir(s) 15,322,472,448 bytes free
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (i
rces>ruby shape_moving.rb
Shape Moving

17 Maze Creation

Create a grid for the search task in 10.3

Outcome	Weight
Functional Decomposition	♦♦♦♦◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦♦♦♦◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Maze Creation

Submitted By:
S M Ragib Rezwan
103172423
2021/03/26 12:57

Tutor:
NAJAM NAZAR

March 26, 2021

```ruby
require 'gosu'

module ZOrder
  BACKGROUND, MIDDLE, TOP = *0..2
end

MAP_WIDTH = 200
MAP_HEIGHT = 200
CELL_DIM = 20

class Cell
  # have a pointer to the neighbouring cells
  attr_accessor :north, :south, :east, :west, :vacant, :visited, :on_path

  def initialize()
    # Set the pointers to nil
    @north = nil
    @south = nil
    @east = nil
    @west = nil
    # record whether this cell is vacant
    # default is not vacant i.e is a wall.
    @vacant = false
    # this stops cycles - set when you travel through a cell
    @visited = false
    @on_path = false
  end
end

# Instructions:
# Left click on cells to create a maze with at least one path moving from
# left to right.  The right click on a cell for the program to find a path
# through the maze. When a path is found it will be displayed in red.
class GameWindow < Gosu::Window

  # initialize creates a window with a width an a height
  # and a caption. It also sets up any variables to be used.
  # This is procedure i.e the return value is 'undefined'
  def initialize
    super MAP_WIDTH, MAP_HEIGHT, false
    self.caption = "Map Creation"
    @path = nil

    x_cell_count = MAP_WIDTH / CELL_DIM
    y_cell_count = MAP_HEIGHT / CELL_DIM

    @columns = Array.new(x_cell_count)
    column_index = 0

    # first create cells for each position
    while (column_index < x_cell_count)
      row = Array.new(y_cell_count)
      @columns[column_index] = row
```

```
54        row_index = 0
55        while (row_index < y_cell_count)
56          cell = Cell.new()
57          @columns[column_index][row_index] = cell
58          row_index += 1
59        end
60        column_index += 1
61      end
62
63      # now set up the neighbour links
64      # You need to do this using a while loop with another
65      # nested while loop inside.
66      column_index=0
67
68      while (column_index < x_cell_count)
69        row = Array.new(y_cell_count)
70        @columns[column_index] = row
71        row_index = 0
72        while (row_index < y_cell_count)
73          cell = Cell.new()
74          @columns[column_index][row_index] = cell
75
76          @S = "Cell x: " + column_index.to_s + ", y: " + row_index.to_s
77
78              if(row_index==0)
79                  cell.north=0
80                  cell.south=1
81                  @S+= " north:0 " + "south: 1 "
82                      if(column_index==0)
83                          cell.west=0
84                          cell.east=1
85
86                          @S+= "east: 1 " + "west: 0 "
87
88
89                      elsif(column_index==9)
90                          cell.west=1
91                          cell.east=0
92                          @S+= "east: 0 " + "west: 1"
93
94                      else
95                          cell.west=1
96                          cell.east=1
97                          @S+= "east: 1 " + "west: 1"
98                      end
99
100
101
102             elsif(row_index==9)
103             cell.north=1
104                 cell.south=0
105                 @S+= " north:1 " + "south: 0 "
106                     if(column_index==0)
```

```
                    cell.west=0
                    cell.east=1

                    @S+= "east: 1 " + "west: 0 "


                elsif(column_index==9)
                    cell.west=1
                    cell.east=0
                    @S+= "east: 0 " + "west: 1"

                else
                    cell.west=1
                    cell.east=1
                    @S+= "east: 1 " + "west: 1"
                end
            else
                cell.north=1
                cell.south=1
                @S+= " north:1 " + "south: 1 "
                if(column_index==0)
                    cell.west=0
                    cell.east=1

                    @S+= "east: 1 " + "west: 0 "


                elsif(column_index==9)
                    cell.west=1
                    cell.east=0
                    @S+= "east: 0 " + "west: 1"

                else
                    cell.west=1
                    cell.east=1
                    @S+= "east: 1 " + "west: 1"
                end

            end
        puts @S
        row_index += 1

    end
    puts "------------ END of Column ------------"
    column_index += 1

end



#new plan. call all x and y. then say position of selected value.
#like for x==0, if x==1 is selected, say east:1
#use i and say for i less than x_cell_count, column[i], if
 ↪  column[i]+1==column[i+1], say east
```

```
    north=0
    south=0
    east=0
    west=0

    #while (mouse_over_cell(mouse_x, mouse_y))


    #end
    #problem, (value stored in x) +1 == to value stored in (x+1)?
    #will this stand true?
    i=0
    #=begin

    #while (cell_y[i] >= 0 && cell_y <= MAP_WIDTH)
    #    if cell_y[i]+1==cell_y[i+1]
    #        south=1
    #
    #    else if cell_y[i]-1==cell_y[i-1]
    #        north=1
        #else
            #south=0
            #north=0
        #end
    #end


    #north=0 for no cell, 1 for cell in north
    #south=0
    #east=0
    #west=0

    #puts( cell_x.to_s)
    #once array has been selected for selected cells,
    #has it been removed from x_cell_count?
    #or will x_cell_count still exist for it?
end


  # this is called by Gosu to see if should show the cursor (or mouse)
  def needs_cursor?
    true
  end







  # Returns an array of the cell x and y coordinates that were clicked on
  def mouse_over_cell(mouse_x, mouse_y)
    if mouse_x <= CELL_DIM
      cell_x = 0
    else
```

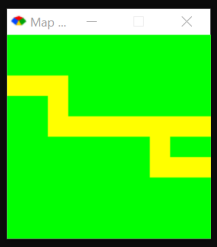
```
213       cell_x = (mouse_x / CELL_DIM).to_i
214     end
215
216     if mouse_y <= CELL_DIM
217       cell_y = 0
218     else
219       cell_y = (mouse_y / CELL_DIM).to_i
220     end
221
222     [cell_x, cell_y]
223
224 end
225
226
227   # start a recursive search for paths from the selected cell
228   # it searches till it hits the East 'wall' then stops
229   # it does not necessarily find the shortest path
230
231   # Completing this function is NOT NECESSARY for the Maze Creation task
232   # complete the following for the Maze Search task - after
233   # we cover Recusion in the lectures.
234
235   # But you DO need to complete it later for the Maze Search task
236   def search(cell_x ,cell_y)
237
238     dead_end = false
239     path_found = false
240
241     if (cell_x == ((MAP_WIDTH / CELL_DIM) - 1))
242       if (ARGV.length > 0) # debug
243         puts "End of one path x: " + cell_x.to_s + " y: " + cell_y.to_s
244       end
245       [[cell_x,cell_y]]  # We are at the east wall - exit
246     else
247
248       north_path = nil
249       west_path = nil
250       east_path = nil
251       south_path = nil
252
253       if (ARGV.length > 0) # debug
254         puts "Searching. In cell x: " + cell_x.to_s + " y: " + cell_y.to_s
255       end
256
257       # INSERT MISSING CODE HERE!! You need to have 4 'if' tests to
258       # check each surrounding cell. Make use of the attributes for
259       # cells such as vacant, visited and on_path.
260       # Cells on the outer boundaries will always have a nil on the
261       # boundary side
262
263       # pick one of the possible paths that is not nil (if any):
264       if (north_path != nil)
265         path = north_path
```

```
266       elsif (south_path != nil)
267         path = south_path
268       elsif (east_path != nil)
269         path = east_path
270       elsif (west_path != nil)
271         path = west_path
272       end
273
274       # A path was found:
275       if (path != nil)
276         if (ARGV.length > 0) # debug
277           puts "Added x: " + cell_x.to_s + " y: " + cell_y.to_s
278         end
279         [[cell_x,cell_y]].concat(path)
280       else
281         if (ARGV.length > 0) # debug
282           puts "Dead end x: " + cell_x.to_s + " y: " + cell_y.to_s
283         end
284         nil  # dead end
285       end
286     end
287   end
288
289   # Reacts to button press
290   # left button marks a cell vacant
291   # Right button starts a path search from the clicked cell
292   def button_down(id)
293     case id
294       when Gosu::MsLeft
295         cell = mouse_over_cell(mouse_x, mouse_y)
296         if (ARGV.length > 0) # debug
297           puts("Cell clicked on is x: " + cell[0].to_s + " y: " + cell[1].to_s)
298         end
299         @columns[cell[0]][cell[1]].vacant = true
300       when Gosu::MsRight
301         cell = mouse_over_cell(mouse_x, mouse_y)
302         @path = search(cell[0],cell[1])
303       end
304
305     #print  cell #this will print the array for selected cells.
306
307   end
308
309   # This will walk along the path setting the on_path for each cell
310   # to true. Then draw checks this and displays them a red colour.
311   def walk(path)
312       index = path.length
313       count = 0
314       while (count < index)
315         cell = path[count]
316         @columns[cell[0]][cell[1]].on_path = true
317         count += 1
318       end
```

```ruby
  end

  # Put any work you want done in update
  # This is a procedure i.e the return value is 'undefined'
  def update
    if (@path != nil)
      if (ARGV.length > 0) # debug
        puts "Displaying path"
        puts @path.to_s
      end
      walk(@path)
      @path = nil
    end
  end









 # Draw (or Redraw) the window
  # This is procedure i.e the return value is 'undefined'
  def draw
    index = 0
    x_loc = 0;
    y_loc = 0;

    x_cell_count = MAP_WIDTH / CELL_DIM
    y_cell_count = MAP_HEIGHT / CELL_DIM

    column_index = 0
    while (column_index < x_cell_count)
      row_index = 0
      while (row_index < y_cell_count)

        if (@columns[column_index][row_index].vacant)
          color = Gosu::Color::YELLOW
        else
          color = Gosu::Color::GREEN
        end
        if (@columns[column_index][row_index].on_path)
          color = Gosu::Color::RED
        end

        Gosu.draw_rect(column_index * CELL_DIM, row_index * CELL_DIM, CELL_DIM,
        ↪  CELL_DIM, color, ZOrder::TOP, mode=:default)

        row_index += 1
      end
      column_index += 1
    end
  end
end

```

```
371 window = GameWindow.new
372 window.show
```

```
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 4\4.4D-resources\Resources>ruby gosu_maze_creation.rb
Cell x: 0, y: 0 north:0 south: 1 east: 1 west: 0
Cell x: 0, y: 1 north:1 south: 1 east: 1 west: 0
Cell x: 0, y: 2 north:1 south: 1 east: 1 west: 0
Cell x: 0, y: 3 north:1 south: 1 east: 1 west: 0
Cell x: 0, y: 4 north:1 south: 1 east: 1 west: 0
Cell x: 0, y: 5 north:1 south: 1 east: 1 west: 0
Cell x: 0, y: 6 north:1 south: 1 east: 1 west: 0
Cell x: 0, y: 7 north:1 south: 1 east: 1 west: 0
Cell x: 0, y: 8 north:1 south: 1 east: 1 west: 0
Cell x: 0, y: 9 north:1 south: 0 east: 1 west: 0
------------ END of Column ------------
Cell x: 1, y: 0 north:0 south: 1 east: 1 west: 1
Cell x: 1, y: 1 north:1 south: 1 east: 1 west: 1
Cell x: 1, y: 2 north:1 south: 1 east: 1 west: 1
Cell x: 1, y: 3 north:1 south: 1 east: 1 west: 1
Cell x: 1, y: 4 north:1 south: 1 east: 1 west: 1
Cell x: 1, y: 5 north:1 south: 1 east: 1 west: 1
Cell x: 1, y: 6 north:1 south: 1 east: 1 west: 1
Cell x: 1, y: 7 north:1 south: 1 east: 1 west: 1
Cell x: 1, y: 8 north:1 south: 1 east: 1 west: 1
Cell x: 1, y: 9 north:1 south: 0 east: 1 west: 1
------------ END of Column ------------
Cell x: 2, y: 0 north:0 south: 1 east: 1 west: 1
Cell x: 2, y: 1 north:1 south: 1 east: 1 west: 1
Cell x: 2, y: 2 north:1 south: 1 east: 1 west: 1
Cell x: 2, y: 3 north:1 south: 1 east: 1 west: 1
Cell x: 2, y: 4 north:1 south: 1 east: 1 west: 1
Cell x: 2, y: 5 north:1 south: 1 east: 1 west: 1
Cell x: 2, y: 6 north:1 south: 1 east: 1 west: 1
Cell x: 2, y: 7 north:1 south: 1 east: 1 west: 1
Cell x: 2, y: 8 north:1 south: 1 east: 1 west: 1
Cell x: 2, y: 9 north:1 south: 0 east: 1 west: 1
------------ END of Column ------------
Cell x: 3, y: 0 north:0 south: 1 east: 1 west: 1
Cell x: 3, y: 1 north:1 south: 1 east: 1 west: 1
Cell x: 3, y: 2 north:1 south: 1 east: 1 west: 1
Cell x: 3, y: 3 north:1 south: 1 east: 1 west: 1
Cell x: 3, y: 4 north:1 south: 1 east: 1 west: 1
Cell x: 3, y: 5 north:1 south: 1 east: 1 west: 1
Cell x: 3, y: 6 north:1 south: 1 east: 1 west: 1
```



18 Music Records

Create a record for the Music Player

Outcome	Weight
Functional Decomposition	♦♦♦♦◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦♦◊◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Music Records

Submitted By:
S M Ragib Rezwan
103172423
2021/03/29 09:04

Tutor:
NAJAM NAZAR

March 29, 2021

```ruby
require './input_functions'

module Genre
  POP, CLASSIC, JAZZ, ROCK = *1..4
end

# Maybe the following needs to be changed? How?
$genre_names = ['Null', 'Pop', 'Classic', 'Jazz', 'Rock']

class Album
  attr_accessor :title, :artist, :genre

end

# This function Reads in and returns a single album from the given file, with all its tracks.
# ...for now however, take input from the terminal to enter just the album information.
# Complete the missing lines of code and change the functionality so that the hardcoded
# values are not used.

def read_album()

  # You could use get_integer_range below to get a genre.
  # You only the need validation if reading from the terminal
  # (i.e when using a file later, you can assume the data in
  # the file is correct)

  # insert lines here - use read_integer_in_range to get a genre
  puts("Enter Album")
  album_title = read_string("Enter album name:")
  album_artist = read_string("Enter artist name:")
  #album_genre = Genre::POP


  album_genre = read_integer_in_range("Enter Genre between 1 - 4:",0,5)

  album = Album.new()
  album.title = album_title
  album.artist = album_artist
  album.genre = album_genre
  return album
end

# Takes a single album and prints it to the terminal
# complete the missing lines:

def print_album(album)
  puts('Album information is: ')
  puts album.title
  puts album.artist
  puts album.genre
```

```
51      # insert lines here
52      puts('Genre is ' + album.genre.to_s)
53      puts($genre_names[album.genre])
54  end
55
56  # Reads in an Album then prints it to the terminal
57
58  def main()
59      album = read_album()
60      print_album(album)
61  end
62
63  main()
```

```
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 5\5.2P-resources\Resources>music_record.rb
Enter Album
Enter album name:
Greatest Hits
Enter artist name:
Don McClean
Enter Genre between 1 - 4:
2
Album information is:
Greatest Hits
Don McClean
2
Genre is 2
Classic

C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 5\5.2P-resources\Resources>
```

19 Hover Button

Create reactive button in GOSU

Outcome	Weight
Functional Decomposition	♦♦◊◊◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦♦♦◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Hover Button

Submitted By:
S M Ragib Rezwan
103172423
2021/03/29 09:47

Tutor:
NAJAM NAZAR

March 29, 2021

```ruby
require 'rubygems'
require 'gosu'

# Instructions:  This code also needs to be fixed and finished!
# As in the earlier button tasks the "Click Me" text is not appearing
# on the button, also both the mouse_x and mouse_ co-ordinate should
# be shown, regardless of whether the mouse has been clicked or not.
# The button should be highlighted when the mouse moves over it
# (i.e it should have a black border around the outside)
# finally, a user has noticed that in this version also sometimes the
# button action occurs when you click outside the button area and vice-versa.


# FOR THE CREDIT VERSION:
# display a colored border that 'highlights' the button when the mouse moves over it

# determines whether a graphical widget is placed over others or not
module ZOrder
  BACKGROUND, MIDDLE, TOP = *0..2
end

# Global constants
WIN_WIDTH = 640
WIN_HEIGHT = 400

class DemoWindow < Gosu::Window

  # set up variables and attributes
  def initialize
    super(WIN_WIDTH, WIN_HEIGHT, false)
    @background = Gosu::Color::WHITE
    @button_font = Gosu::Font.new(20)
    @info_font = Gosu::Font.new(10)
    @locs = [60,60]
  end


  # Draw the background, the button with 'click me' text and text
  # showing the mouse coordinates
  def draw
    # Draw background color
    Gosu.draw_rect(0, 0, WIN_WIDTH, WIN_HEIGHT, @background, ZOrder::BACKGROUND,
     ↪  mode=:default)
    # Draw the rectangle that provides the background.
    # ????
    # Draw the button
    Gosu.draw_rect(50, 50, 100, 50, Gosu::Color::GREEN, ZOrder::TOP, mode=:default)

    if  mouse_over_button(mouse_x, mouse_y)
    Gosu.draw_rect(45, 45, 110, 60, Gosu::Color::BLACK, ZOrder::MIDDLE,
     ↪  mode=:default)
    else
    end
```

```ruby
    # Draw the button text
    @button_font.draw("Click me", 60, 60, ZOrder::TOP, 1.0, 1.0, Gosu::Color::BLACK)
    # Draw the mouse_x position
    @info_font.draw("mouse_x: #{mouse_x}", 0, 350, ZOrder::TOP, 1.0, 1.0,
    ↪ Gosu::Color::BLACK)
    # Draw the mouse_y position
    # @info_font.draw("mouse_y: #{mouse_y}", ......... )
    @info_font.draw("mouse_y: #{mouse_y}", 100, 350, ZOrder::TOP, 1.0, 1.0,
    ↪ Gosu::Color::BLACK)
  end

  # this is called by Gosu to see if it should show the cursor (or mouse)
  def needs_cursor?; true; end

 # This still needs to be fixed!

  def mouse_over_button(mouse_x, mouse_y)
    if ((mouse_x >= 50 && mouse_x <= 150) && (mouse_y >= 50 && mouse_y <= 100))
      true

    else
      false
    end
  end

  # If the button area (rectangle) has been clicked on change the background color
  # also store the mouse_x and mouse_y attributes that we 'inherit' from Gosu
  # you will learn about inheritance in the OOP unit - for now just accept that
  # these are available and filled with the latest x and y locations of the mouse
  ↪ click.
  def button_down(id)
    case id
    when Gosu::MsLeft
      if mouse_over_button(mouse_x, mouse_y)
        @background = Gosu::Color::YELLOW
      else
        @background = Gosu::Color::WHITE
      end
    end
  end
end




# Lets get started!
DemoWindow.new.show
```


Click me
mouse_x: 69.0
mouse_y: 77.0

20 Track File Handling

Read tracks from a file

Outcome	Weight
Functional Decomposition	♦♦♦◊◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦◊◊◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Track File Handling

Submitted By:
S M Ragib Rezwan
103172423
2021/03/28 21:20

Tutor:
NAJAM NAZAR

March 28, 2021

```ruby
class Track
    attr_accessor :name, :location

    def initialize (name, location)
        @name = name
        @location = location
    end
end

# Returns an array of tracks read from the given file
def read_tracks(music_file)

  count = music_file.gets().to_i()
  tracks = Array.new()

  # Put a while loop here which increments an index to read the tracks
  i=0
    while i < count do
      track = read_track(music_file)
      tracks << track
        i = i+1
    end
  return tracks

end

# reads in a single track from the given file.
def read_track(a_file)
    name = a_file.gets()
    location = a_file.gets()
    track = Track.new(name, location)
    return track
  # complete this function
    # you need to create a Track here
end


# Takes an array of tracks and prints them to the terminal
def print_tracks(tracks)

    i=0
    while (i < tracks.length) do
    print_track(tracks[i])
    i = i +1
    end
  # Use a while loop with a control variable index
  # to print each track. Use tracks.length to determine how
  # many times to loop.

  # Print each track use: tracks[index] to get each track record
end

# Takes a single track and prints it to the terminal
```

```
54 def print_track(track)
55   puts('Track title is: ' + track.name)
56     puts('Track file location is: ' + track.location)
57 end
58 
59 # Open the file and read in the tracks then print them
60 def main()
61   a_file = File.new("input.txt", "r") # open for reading
62   tracks = read_tracks(a_file)
63 
64   # Print all the tracks
65   print_tracks(tracks)
66 end
67 
68 main
```

```
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 5\5.1T-resources\Resources>track_file_handling.rb
Track title is: Crackling Rose
Track file location is: sounds/01-Cracklin-rose.wav
Track title is: Soolaimon
Track file location is: sounds/06-Soolaimon.wav
Track title is: Sweet Caroline
Track file location is: sounds/20-Sweet_Caroline.wav

C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 5\5.1T-resources\Resources>
```

21 Array Search

Search an array for an item.

Outcome	Weight
Functional Decomposition	♦♦♦◊◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦◊◊◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Array Search

Submitted By:
S M Ragib Rezwan
103172423
2021/04/10 09:41

Tutor:
NAJAM NAZAR

April 10, 2021

```ruby
require './input_functions'

# Task 6.1 T - use the code from 5.1 and 5.2 T to help with this



class Track
    attr_accessor :name, :location

    def initialize (name, location)
        @name = name
        @location = location
    end
end

# Reads in and returns a single track from the given file

def read_track(music_file)
    # fill in the missing code
    name = music_file.gets()
    location = music_file.gets()
    track = Track.new(name, location)
    return track
end

# Returns an array of tracks read from the given file

def read_tracks(music_file)
     count = music_file.gets().to_i()
  tracks = Array.new()

  # Put a while loop here which increments an index to read the tracks
  i=0
    while i < count do
      track = read_track(music_file)
      tracks << track
        i = i+1
    end
  return tracks

end

# Takes an array of tracks and prints them to the terminal

def print_tracks(tracks)
    # print all the tracks use: tracks[x] to access each track.
    i=0
    while (i < tracks.length) do
    print_track(tracks[i])
    i = i +1
    end
end

```

```ruby
# Takes a single track and prints it to the terminal
def print_track(track)
      puts(track.name)
    puts(track.location)
end


# search for track by name.
# Returns the index of the track or -1 if not found
def search_for_track_name(tracks, search_string)

    index = 0
    found_index = -1
    while (index < tracks.length )


        if(tracks[index].name.chomp == search_string)
            found_index = index

        end
        index = index + 1
    end

# Put a while loop here that searches through the tracks
# Use the read_string() function from input_functions.
# NB: you might need to use .chomp to compare the strings correctly

# Put your code here.

  return found_index
end


# Reads in an Album from a file and then prints all the album
# to the terminal

def main()
      music_file = File.new("album.txt", "r")
    tracks = read_tracks(music_file)
      music_file.close()
    print_tracks(tracks)


      search_string = read_string("Enter the track name you wish to find: ")
      index = search_for_track_name(tracks, search_string)
      if index > -1
           puts "Found " + tracks[index].name + " at " + index.to_s()
      else
        puts "Entry not Found"
      end
end

main()
```

```
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week6\6.1T-resources\Resources>track_search.rb
Crackling Rose
sounds/01-Cracklin-rose.wav
Soolaimon
sounds/06-Soolaimon.wav
Sweet Caroline
sounds/20-Sweet_Caroline.wav
Enter the track name you wish to find:
Soolaimon
Found Soolaimon
 at 1
```

22 Album File Handling

Read Album information for the Music Player

Outcome	Weight
Functional Decomposition	♦♦♦♦◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦♦◊◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Album File Handling

Submitted By:
S M Ragib Rezwan
103172423
2021/04/12 19:50

Tutor:
NAJAM NAZAR

April 12, 2021

```ruby
# Task 6.2 T - use the code from 5.1 and 5.2 T to help with this

module Genre
  POP, CLASSIC, JAZZ, ROCK = *1..4
end

$genre_names = ['Null', 'Pop', 'Classic', 'Jazz', 'Rock']

class Album
  # you will need to add tracks to the following and the initialize()
    attr_accessor :title, :artist, :genre, :tracks

# complete the missing code:
    def initialize (title, artist, genre, tracks)
        # insert lines here
        @title = title

        @artist = artist
        @genre = genre
        @tracks = tracks

    end
end

class Track
    attr_accessor :name, :location

    def initialize (name, location)
        @name = name
        @location = location
    end
end

# Reads in and returns a single track from the given file

def read_track(music_file)
    # fill in the missing code
    name = music_file.gets()
    location = music_file.gets()
    track = Track.new(name, location)

    return track
end

# Returns an array of tracks read from the given file

def read_tracks(music_file)
    tracks = Array.new()
    count = music_file.gets().to_i()


  # Put a while loop here which increments an index to read the tracks
i=0
```

```ruby
    while i < count do
      track = read_track(music_file)

      tracks << track
        i = i+1
    end

  return tracks
end

# Takes an array of tracks and prints them to the terminal

def print_tracks(tracks)
    # print all the tracks use: tracks[x] to access each track.
    i=0

    while (i < tracks.length) do
    print_track(tracks[i])
    i = i +1
    end
end

# Reads in and returns a single album from the given file, with all its tracks

def read_album(music_file)

  # read in all the Album's fields/attributes including all the tracks
  # complete the missing code
    album_artist = music_file.gets()
    album_title = music_file.gets()
    album_genre = music_file.gets()
    tracks = read_tracks (music_file)
    album = Album.new(album_title, album_artist, album_genre, tracks)

    return album

end


# Takes a single album and prints it to the terminal along with all its tracks
def print_album(album)

  # print out all the albums fields/attributes
  # Complete the missing code.
    puts (album.artist.to_s())
    puts (album.title.to_s())
    puts('Genre is ' + album.genre.to_s())
    puts($genre_names[album.genre.to_i])
    print_tracks(album.tracks)
    #album.track is must! otherwise it wont realise that we are
    #printing tracks and instead will make a new null array
```

```ruby
    # print out the tracks

end

# Takes a single track and prints it to the terminal
def print_track(track)
    puts(track.name)
    puts(track.location)
end


# Reads in an Album from a file and then prints all the album
# to the terminal

def main()
      music_file = File.new("album.txt", "r")
      album = read_album(music_file)
      music_file.close()
      print_album(album)
end

main()
```

```
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week6\6.2P-resources\Resources>album_file_handling.rb
Neil Diamond
Greatest Hits
Genre is 1
Pop
Crackling Rose
sounds/01-Cracklin-rose.wav
Soolaimon
sounds/06-Soolaimon.wav
Sweet Caroline
sounds/20-Sweet_Caroline.wav

C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week6\6.2P-resources\Resources>
```

23 Custom Program Design

Design for your Custom program

Outcome	Weight
Functional Decomposition	♦♦◊◊◊

Since it is saying about the name of functions used in the game, it can be counted as being relevant to decomposition.

Outcome	Weight
Structured Coding Principles	♦♦♦◊◊

Outcome	Weight
Programming	♦♦♦♦◊

Outcome	Weight
Apply Reading Techniques	♦♦♦♦♦

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Custom Program Design

Submitted By:
S M Ragib Rezwan
103172423
2021/05/31 20:08

Tutor:
NAJAM NAZAR

May 31, 2021

Design Overview for Orb Collector

Name: S M Ragib Rezwan
Student ID: 103172423

Summary of Program

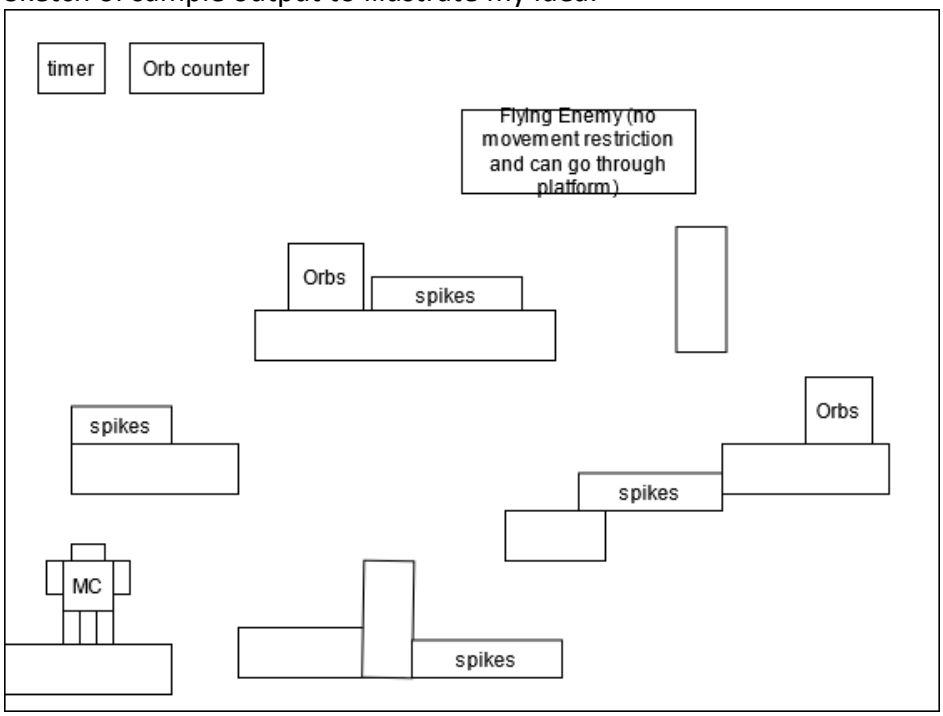
It is basically a platforming game where a character moves across the game map collecting orbs while avoiding falling into spikes and coming into contact with enemies.

The game's is backstory is something akin to this:

You are a world famous collector obsessed with collecting objects called orbs. In order to obtain them, you will go anywhere, even scouring though treacherous the terrain, just to obtain all of them. Even, if its kept high out of reach or barricaded by rows of spikes, you will still find a way to obtain them no matter what.

But unfortunately, collecting these orbs have ended up putting deadly mimics on your tail. Ones who can smell the orbs you have collected throughout time and won't ever stop coming until you're dead. Can you survive their pursuit, the deadly terrain and also fulfil your obsession in time?

Sketch of sample output to illustrate my idea:



Required Data Types

All records:

Table 1: GameMap details (take help from lect09 or ruby for this)

Field Name	Type	Notes
Width	Integer (basic)	Gives width of map
Height	Integer (basic)	Gives height of map
orb	Orb data (complex)	Links to orb data
spike	Spike data (complex)	Links to spike data
enemy	Enemy data (complex)	Links to enemy data
tileset	Image data for platform (complex)	Calls up image files and links pic to tile module
tiles	Array (basic)	It mainly refers to the module created and links it

Table 2: Player details (take help from lect09 or ruby for this)

Field Name	Type	Notes
x	Integer (basic)	Player's x position
y	Integer (basic)	Player's y position
dir	Left or right (complex)	Mainly it sets which direction player is facing and turns character.
vy	Integer (basic)	Player's vertical velocity
game_map	Game_map data (complex)	Links the data from map to player
standing	Image (complex)	Stores picture when player is standing
walk1	Image (complex)	Stores picture when player is walking (first frame)
walk2	Image (complex)	Stores picture when player is walking (second frame)
jump	Image (complex)	Stores picture when player is jumping
current_image	Image (complex)	Passed on the player's current image as that of standing, walk1, walk2 or jumping

Table 3: Orb details

Field Name	Type	Notes
x	Integer (basic)	Orb's x position
y	Integer (basic)	Orb's y position
image	Image (complex)	Stores Orb's picture

Table 4: Spike details

Field Name	Type	Notes
x	Integer (basic)	Spike's x position
y	Integer (basic)	Spike's y position
image	Image (complex)	Stores spike's image

Table 5: Enemy details

Field Name	Type	Notes
x	Integer (basic)	Enemy's x position
y	Integer (basic)	Enemy's y position
image	Image (complex)	Stores enemy's image

All Enumerations:

Table 1: Tiles details

Value	Notes
Land	Stores picture for border and land part and links to gamemap
Rock	Stores picture for platform and links it to gamemap

Table 2: Other constant values

Value	Notes
Width	Width of screen of game
Height	Height of screen of game

Overview of Program Structure

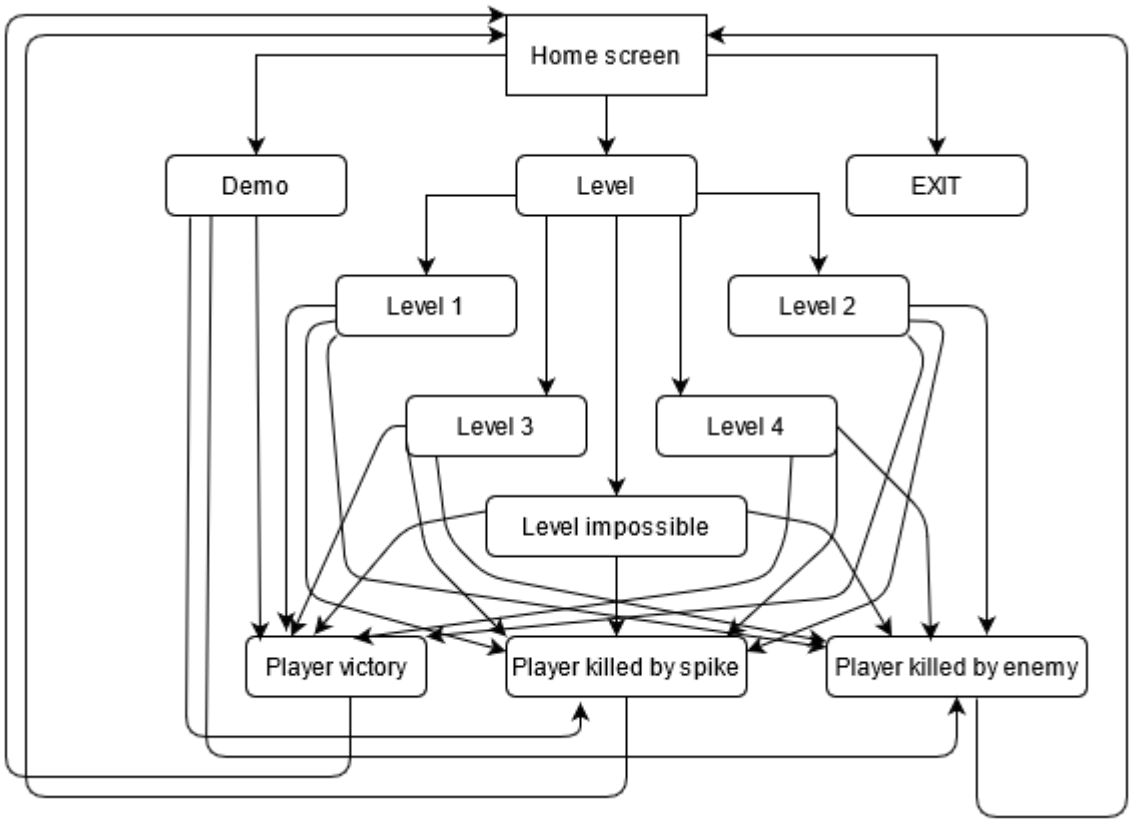
List of main functions/procedures needed to create this program (and brief description)

Name of procedure/ function	description
Setup_game_map(filename)	Helping code read symbols from a txt file and understanding where to place which picture in the main game's level and store in parameter gamemap
Draw_game_map (game_map)	Draws the entire level using data passed from game map and adding extra condition (like spinning for orb)
Solid?(game_map, x ,y)	Gives hitbox of platform for player and platform interaction

Setup_player(player, game_map, x, y)	Sets up all data for player class and links them to specific variables and images
draw_player(player)	Draws player. Also makes sure to differentiate which way it is facing and assigns appropriate condition to each
would_fit(player, offs_x, offs_y)	Sets up player hitbox for player and platform interaction
update_player(player, move_x)	Sets up all player movement, player platform interaction, vertical velocity
try_to_jump(player)	Sets up a condition to jump distance via vertical velocity
Enemy_Data(image, x, y)	Sets up all enemy data
enemy_draw(enemy)	Draws enemy
enemy_move(player, enemy)	Sets up the condition for enemy movement (ie the enemy AI) by using player position
enemy_touch(player, enemy)	Sets up condition for player to get killed by enemy
Spike_Data(image, x, y)	Sets up all spike data
spike_draw(spike)	Draws spike
spike_touch(player, spike)	Sets up condition for player to get killed by spike
Orb_Data(image, x, y)	Sets up all orb data
orb_spin(orb)	Sets up a special condition to make sure orb spins in a certain way to differentiate it from other objects in game and attract focus and attention
orb_gone(player, orb)	Sets up condition for player to take in orb and also sets up condition for player to be victorious (ie all orbs collected)
draw_start_screen_elements(z_order)	draws up all start screen elements with appropriate z order level allocated to each
start_button_elements(z_order)	Draws start (ie the “demo”) button elements
levels(z_order)	Draws level button elements
quit_button_elements(z_order)	Draws quit button elements
level_1_button_elements(z_order)	Draws level 1 button elements
level_2_button_elements(z_order)	Draws level 2 button elements
level_3_button_elements(z_order)	Draws level 3 button elements

level_4_button_elements(z_order)	Draws level 4 button elements
level_impossible_button_elements(z_order)	Draws level impossible button elements
draw_level_extra_elements(z_order)	Calls up the extra level items to be drawn (ie timer and orb counter)
draw_level_timer(z_order)	Draws timer elements on each levels
draw_orb_counter(z_order)	Draws orb counter elements on each levels
draw_victory_end(z_order)	Draws up victory screen text and elements, alongside a comment about player's game play via timer count
draw_defeat_spike_end(z_order)	Draws up player killed by spike screen and elements
draw_defeat_enemy_end(z_order)	Draws up player killed by enemy screen and elements
return_button_elements(z_order)	Draws up all return to home button elements for all end screens.
mouse_over_button_start(mouse_x, mouse_y)	Setting a clickable area for start (demo) button
mouse_over_button_level_1(mouse_x, mouse_y)	Setting a clickable area for start level 1 button
mouse_over_button_level_2(mouse_x, mouse_y)	Setting a clickable area for start level 2 button
mouse_over_button_level_3(mouse_x, mouse_y)	Setting a clickable area for start level 3 button
mouse_over_button_level_4(mouse_x, mouse_y)	Setting a clickable area for start level 4 button
mouse_over_button_level_impossible(mouse_x, mouse_y)	Setting a clickable area for start level impossible button
mouse_over_button_level(mouse_x, mouse_y)	Setting a clickable area for level button
mouse_over_button_end(mouse_x, mouse_y)	Setting a clickable area for return home button
mouse_over_button_quit(mouse_x, mouse_y)	Setting a clickable area for quit button

Basic Structure Chart:



24 Maze Search

A path-searching program

Outcome	Weight
Functional Decomposition	♦♦♦♦◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦♦♦♦♦

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Maze Search

Submitted By:
S M Ragib Rezwan
103172423
2021/05/14 10:36

Tutor:
NAJAM NAZAR

May 14, 2021

```ruby
require 'gosu'

module ZOrder
  BACKGROUND, MIDDLE, TOP = *0..2
end

MAP_WIDTH = 200
MAP_HEIGHT = 200
CELL_DIM = 20

class Cell
  # have a pointer to the neighbouring cells
  attr_accessor :north, :south, :east, :west, :vacant, :visited, :on_path

  def initialize()
    # Set the pointers to nil
    @north = nil
    @south = nil
    @east = nil
    @west = nil
    # record whether this cell is vacant
    # default is not vacant i.e is a wall.
    @vacant = false
    # this stops cycles - set when you travel through a cell
    @visited = false
    @on_path = false
  end
end

# Instructions:
# Left click on cells to create a maze with at least one path moving from
# left to right.  The right click on a cell for the program to find a path
# through the maze. When a path is found it will be displayed in red.
class GameWindow < Gosu::Window

  # initialize creates a window with a width an a height
  # and a caption. It also sets up any variables to be used.
  # This is procedure i.e the return value is 'undefined'
  def initialize
    super MAP_WIDTH, MAP_HEIGHT, false
    self.caption = "Map Creation"
    @path = nil
    empty = Cell.new()#needed to add this part as vacant and visited didnt know
     ↪  what to do with integers(1 and 0 given before as border condition) or even
     ↪  with nil

    x_cell_count = MAP_WIDTH / CELL_DIM
    y_cell_count = MAP_HEIGHT / CELL_DIM

    @columns = Array.new(x_cell_count)
    column_index = 0

    # first create cells for each position
```

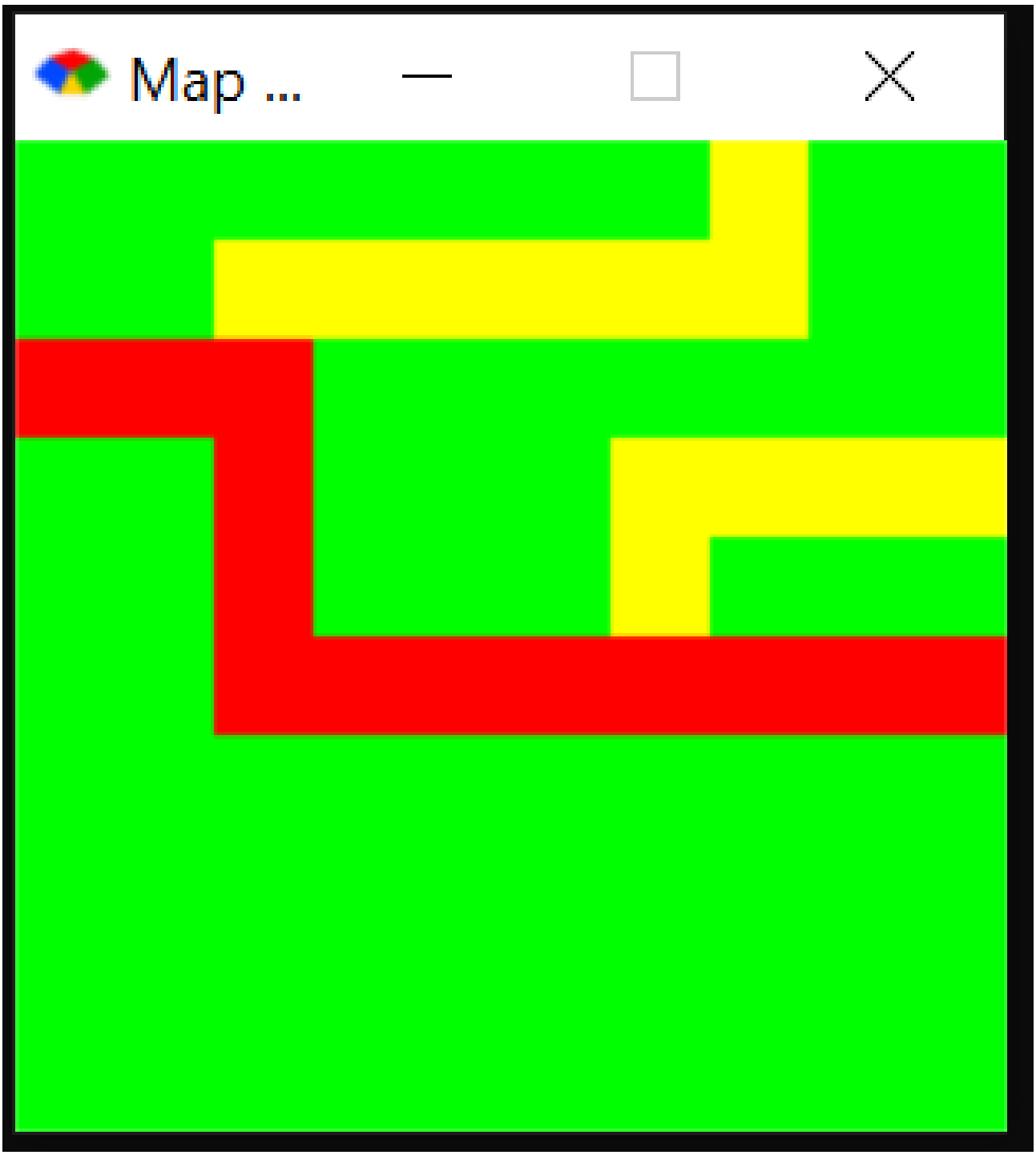
```
52      while (column_index < x_cell_count)
53        row = Array.new(y_cell_count)
54        @columns[column_index] = row
55        row_index = 0
56        while (row_index < y_cell_count)
57          cell = Cell.new()
58          @columns[column_index][row_index] = cell
59          row_index += 1
60        end
61        column_index += 1
62      end
63
64      # now set up the neighbour links
65      # You need to do this using a while loop with another
66      # nested while loop inside.
67
68      #this is a cleaner version of the loop done previously
69      #will give the same output as before and will also be able to be used in the
        ↪  recursion part now due to "empty"
70      column_index = 0
71      while column_index < x_cell_count
72          row_index = 0
73          while row_index < y_cell_count
74              cell = @columns[column_index][row_index]
75
76              @S = "Cell x: " + column_index.to_s + ", y: " + row_index.to_s
77              if row_index == 0
78                  cell.north = empty
79                  @S+= " north =1 "
80              else
81                  cell.north = @columns[column_index][row_index-1]
82                  @S+= " north =0 "
83              end
84              if row_index == y_cell_count-1
85                  cell.south = empty
86                  @S+= "south =1 "
87              else
88                  cell.south = @columns[column_index][row_index+1]
89                  @S+= "south =0 "
90              end
91
92              if column_index == x_cell_count-1
93                  cell.east = empty
94                  @S+= "east =1 "
95              else
96                  cell.east = @columns[column_index+1][row_index]
97                  @S+= "east =0 "
98              end
99              if column_index == 0
100                 cell.west = empty
101                 @S+= "west =1 "
102             else
103                 cell.west = @columns[column_index-1][row_index]
```

```
104                 @S+= "west =0 "
105             end
106             puts @S
107             row_index += 1
108
109             end
110             puts "------------ END of Column ------------"
111         column_index += 1
112         end
113     end
114
115   # this is called by Gosu to see if should show the cursor (or mouse)
116   def needs_cursor?
117     true
118   end
119
120   # Returns an array of the cell x and y coordinates that were clicked on
121   def mouse_over_cell(mouse_x, mouse_y)
122     if mouse_x <= CELL_DIM
123       cell_x = 0
124     else
125       cell_x = (mouse_x / CELL_DIM).to_i
126     end
127
128     if mouse_y <= CELL_DIM
129       cell_y = 0
130     else
131       cell_y = (mouse_y / CELL_DIM).to_i
132     end
133
134     [cell_x, cell_y]
135   end
136
137   # start a recursive search for paths from the selected cell
138   # it searches till it hits the East 'wall' then stops
139   # it does not necessarily find the shortest path
140
141   # Completing this function is NOT NECESSARY for the Maze Creation task
142   # complete the following for the Maze Search task - after
143   # we cover Recusion in the lectures.
144
145   # But you DO need to complete it later for the Maze Search task
146   def search(cell_x ,cell_y)
147     cell = @columns[cell_x][cell_y]
148
149     if (cell_x == ((MAP_WIDTH / CELL_DIM) - 1))
150       if (ARGV.length > 0) # debug
151         puts "End of one path x: " + cell_x.to_s + " y: " + cell_y.to_s
152       end
153       return [[cell_x,cell_y]]  # We are at the east wall - exit
154     else
155
156       if (ARGV.length > 0) # debug
```

```
157         puts "Searching. In cell x: " + cell_x.to_s + " y: " + cell_y.to_s
158       end
159
160       # INSERT MISSING CODE HERE!! You need to have 4 'if' tests to
161       # check each surrounding cell. Make use of the attributes for
162       # cells such as vacant, visited and on_path.
163       # Cells on the outer boundaries will always have a nil on the
164       # boundary side
165
166     cell.visited = true
167     if cell.north.vacant && cell.north.visited == false
168         north_path = search(cell_x, cell_y-1)
169     end
170     if cell.east.vacant && cell.east.visited == false
171         east_path = search(cell_x+1, cell_y)
172     end
173     if cell.south.vacant && cell.south.visited == false
174         south_path = search(cell_x, cell_y+1)
175     end
176     if cell.west.vacant && cell.west.visited == false
177         west_path = search(cell_x-1, cell_y)
178     end
179
180       # pick one of the possible paths that is not nil (if any):
181      if east_path != nil
182         path = east_path
183     elsif north_path != nil
184         path = north_path
185     elsif south_path != nil
186         path = south_path
187     elsif west_path != nil
188         path = west_path
189     end
190
191       # A path was found:
192       if (path != nil)
193         if (ARGV.length > 0) # debug
194           puts "Added x: " + cell_x.to_s + " y: " + cell_y.to_s
195         end
196         return [[cell_x,cell_y]].concat(path)
197       else
198         if (ARGV.length > 0) # debug
199           puts "Dead end x: " + cell_x.to_s + " y: " + cell_y.to_s
200         end
201         return(nil)  # dead end
202       end
203     end
204   end
205
206   # Reacts to button press
207   # left button marks a cell vacant
208   # Right button starts a path search from the clicked cell
209   def button_down(id)
```

```
210       case id
211         when Gosu::MsLeft
212           cell = mouse_over_cell(mouse_x, mouse_y)
213           if (ARGV.length > 0) # debug
214             puts("Cell clicked on is x: " + cell[0].to_s + " y: " + cell[1].to_s)
215           end
216           @columns[cell[0]][cell[1]].vacant = true
217         when Gosu::MsRight
218           cell = mouse_over_cell(mouse_x, mouse_y)
219           @path = search(cell[0],cell[1])
220       end
221     end
222
223     # This will walk along the path setting the on_path for each cell
224     # to true. Then draw checks this and displays them a red colour.
225     def walk(path)
226       index = path.length
227       count = 0
228       while (count < index)
229         cell = path[count]
230         @columns[cell[0]][cell[1]].on_path = true
231         count += 1
232       end
233     end
234
235     # Put any work you want done in update
236     # This is a procedure i.e the return value is 'undefined'
237     def update
238       if (@path != nil)
239         if (ARGV.length > 0) # debug
240           puts "Displaying path"
241           puts @path.to_s
242         end
243         walk(@path)
244         @path = nil
245       end
246     end
247
248     # Draw (or Redraw) the window
249     # This is procedure i.e the return value is 'undefined'
250     def draw
251       index = 0
252       x_loc = 0;
253       y_loc = 0;
254
255       x_cell_count = MAP_WIDTH / CELL_DIM
256       y_cell_count = MAP_HEIGHT / CELL_DIM
257
258       column_index = 0
259       while (column_index < x_cell_count)
260         row_index = 0
261         while (row_index < y_cell_count)
262
```

```ruby
        if (@columns[column_index][row_index].vacant)
          color = Gosu::Color::YELLOW
        else
          color = Gosu::Color::GREEN
        end
        if (@columns[column_index][row_index].on_path)
          color = Gosu::Color::RED
        end

        Gosu.draw_rect(column_index * CELL_DIM, row_index * CELL_DIM, CELL_DIM,
        ↪  CELL_DIM, color, ZOrder::TOP, mode=:default)

        row_index += 1
      end
      column_index += 1
    end
  end
end

window = GameWindow.new
window.show
```


Map ...

25 Test 1

Do this in Ed. Your tutor will mark it there and post the feedback here.

Outcome	Weight
Functional Decomposition	♦♦♦♦◊

Here i had to marge 5.1 and 5.2, so gave it the same rating as that given for those two tasks

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦♦♦◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Test 1

Submitted By:
S M Ragib Rezwan
103172423
2021/04/28 14:37

Tutor:
NAJAM NAZAR

April 28, 2021

```ruby
require './input_functions'

class Bird
    attr_accessor :id, :location, :species, :cage

    def initialize (id, location, species, cage)
        @id = id
        @location = location
        @species = species
        @cage = cage
    end
end
# Complete the code below
# Use input_functions to read the data from the user

def read_a_bird(bird)
    id = read_string("Enter bird id: ")
    location = read_string("Enter bird location: ")
    species = read_string("Enter bird species: ")
    cage= read_string("Enter bird cage number: ")

  bird = Bird.new(id, location, species, cage)
  return bird
end

def read_birds()
count = read_integer("How many birds are you entering: ")
birds = Array.new
i=0
    while i < count do
      bird = read_a_bird(bird)
      birds << bird
        i = i+1
    end
  return birds

end

def print_a_bird(bird)
    puts('Id ' + bird.id)
    puts('Location ' + bird.location)
    puts('Species ' + bird.species)
    puts('Cage Number ' + bird.cage)
end

def print_birds(birds)
    i=0
    while (i < birds.length) do
    print_a_bird(birds[i])
    i = i +1
    end
end

```

```
54 def main()
55     birds = read_birds()
56     print_birds(birds)
57 end
58
59 main()
```

```
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\test>bird.rb
How many birds are you entering:
3
Enter bird id:
100
Enter bird location:
American
Enter bird species:
Pelican
Enter bird cage number:
30
Enter bird id:
200
Enter bird location:
Australia
Enter bird species:
Kookaburra
Enter bird cage number:
30
Enter bird id:
300
Enter bird location:
England
Enter bird species:
Sparrow
Enter bird cage number:
43
Id 100
Location American
Species Pelican
Cage Number 30
Id 200
Location Australia
Species Kookaburra
Cage Number 30
Id 300
Location England
Species Sparrow
Cage Number 43
```

26 Concept Map

Produce a concept map.

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦♦◊◊◊

There is only a concept here related to this task. So not giving it too high of a rating

Outcome	Weight
Functional Decomposition	♦◊◊◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

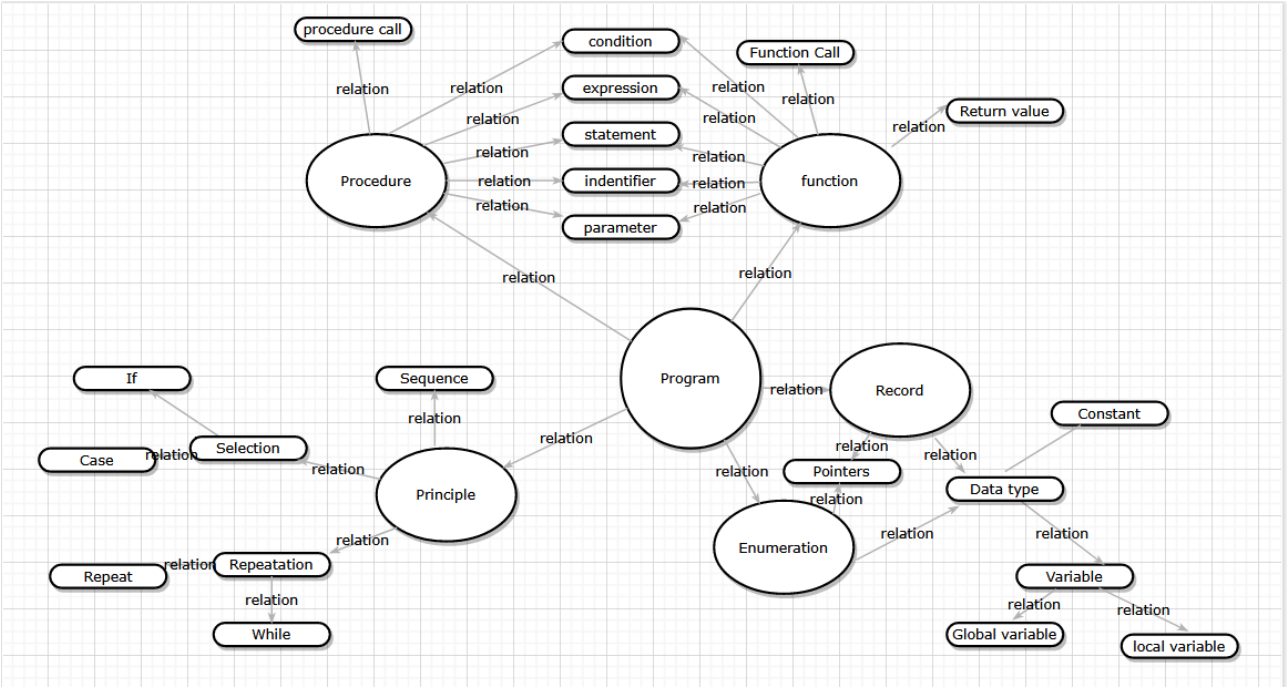
Doubtfire Submission

Concept Map

Submitted By:
S M Ragib Rezwan
103172423
2021/04/30 10:13

Tutor:
NAJAM NAZAR

April 30, 2021


procedure call
condition
Function Call
relation
expression
statement
indentifier
parameter
Procedure
function
Return value
Program
Record
Constant
If
Sequence
Selection
Case
Principle
Pointers
Data type
Enumeration
Repeat
Repeatation
While
Variable
Global variable
local variable

27 Food Hunter

A simple Food Hunter game

Outcome	Weight
Functional Decomposition	♦♦♦♦◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦♦♦♦◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Food Hunter

Submitted By:
S M Ragib Rezwan
103172423
2021/05/13 12:35

Tutor:
NAJAM NAZAR

May 13, 2021

```ruby
# Encoding: UTF-8

require 'rubygems'
require 'gosu'

# Create some constants for the screen width and height

# The following determines which layers things are placed on on the screen
# background is the lowest layer (drawn over by other layers), user interface objects are highest.

module ZOrder
  BACKGROUND, FOOD, PLAYER, UI = *0..3
end


SCREEN_HEIGHT = 800
SCREEN_WIDTH = 600

# Note: There is one class for each record in the Pascal Food Hunter Game

class Hunter
  attr_accessor :score, :image, :yuk, :yum, :hunted, :hunted_image, :vel_x, :vel_y, :angle, :x, :y, :score

  def initialize(hunted)
    @image = Gosu::Image.new("media/Hunter.PNG")
    @yuk = Gosu::Sample.new("media/Yuk.wav")
    @yum = Gosu::Sample.new("media/Yum.wav")

    @hunted = hunted  # default
    @hunted_image = Gosu::Image.new("media/SmallIcecream.png")

    @vel_x = @vel_y = 3.0
    @x = @y = @angle = 0.0
    @score = 0
  end
end

def set_hunted(hunter, hunted)
  hunter.hunted = hunted
  case hunted
  when :chips
    hunted_string = "media/" + "SmallChips.png"
  when :icecream
    hunted_string = "media/" + "SmallIcecream.png"
  when :burger
    hunted_string = "media/" + "SmallBurger.png"
  when :pizza
    hunted_string = "media/" + "SmallPizza.png"
#when :smoke
#    hunted_string = "media/" + "smoke.png"
  end
```

```
52    hunter.hunted_image = Gosu::Image.new(hunted_string)
53  end
54
55  def warp(hunter, x, y)
56    hunter.x, hunter.y = x, y
57  end
58
59  def move_left hunter
60    hunter.x -= hunter.vel_x
61    hunter.x %= SCREEN_WIDTH
62  end
63
64  def move_right hunter
65    hunter.x += hunter.vel_x
66    hunter.x %= SCREEN_WIDTH
67  end
68
69  def move_up hunter
70    hunter.y -= hunter.vel_y
71    hunter.y %= SCREEN_HEIGHT
72
73  end
74
75  def move_down hunter
76    hunter.y += hunter.vel_y
77    hunter.y %= SCREEN_HEIGHT
78  end
79
80  def draw_hunter hunter
81    hunter.image.draw_rot(hunter.x, hunter.y, ZOrder::PLAYER, hunter.angle)
82    hunter.hunted_image.draw_rot(hunter.x, hunter.y, ZOrder::PLAYER, hunter.angle)
83  end
84
85  def collect_food(all_food, hunter)
86    all_food.reject! do |food|
87      if Gosu.distance(hunter.x, hunter.y, food.x, food.y) < 80 # an arbitrary
    ↪  distance - could be improved!!!
88        if (food.type == hunter.hunted)
89          hunter.score += 1
90          hunter.yum.play
91        else
92          hunter.score += -1
93          hunter.yuk.play
94        end
95        true
96      else
97        false
98      end
99    end
100 end
101
102
103 class Food
```

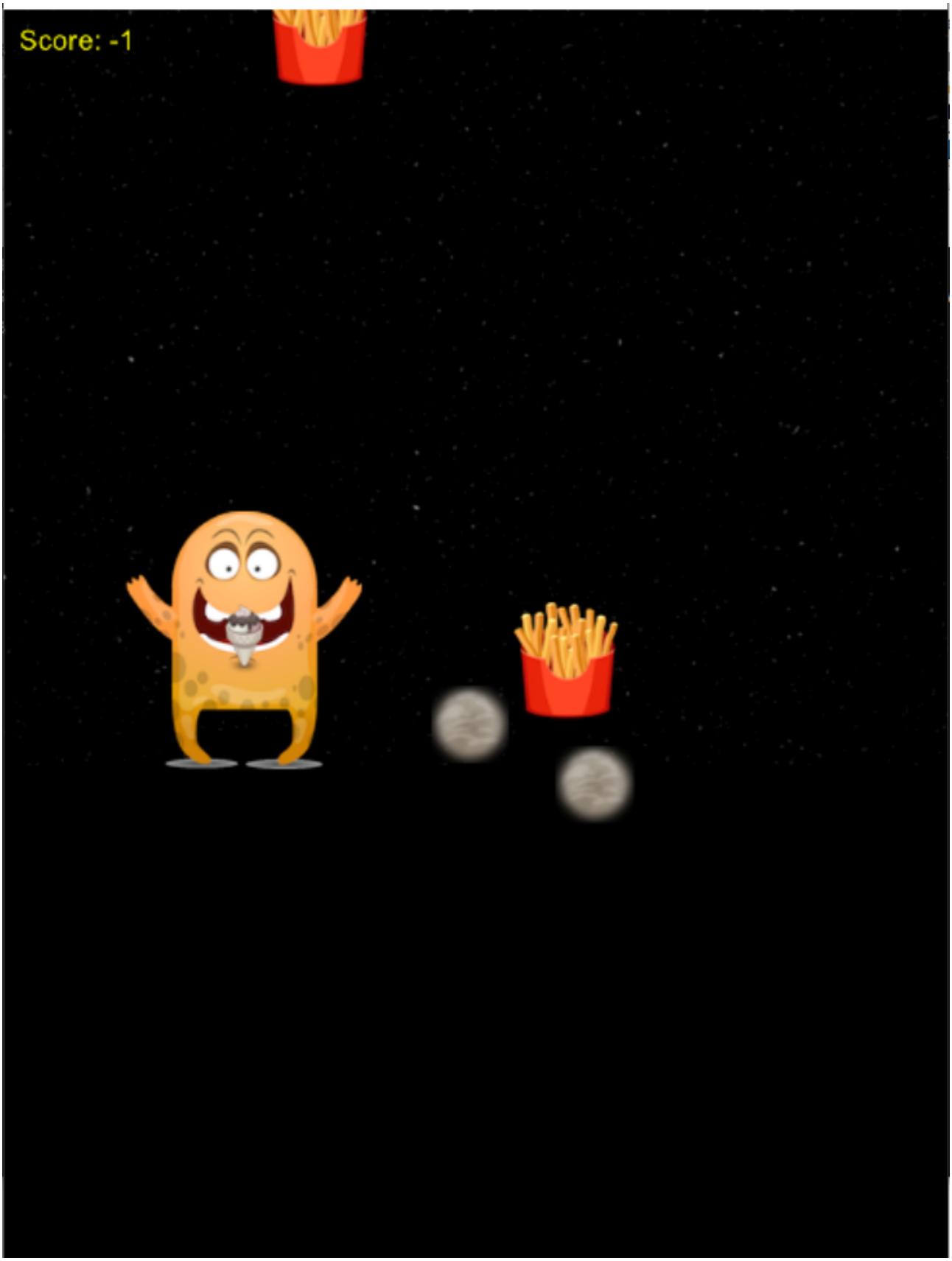
```ruby

  attr_accessor :x, :y, :type, :image, :vel_x, :vel_y, :angle, :x, :y, :score,
   ↪  :trigger, :count, :change_image

  def initialize(image, type)
    @type = type;
    @image = Gosu::Image.new(image);


#@image_flash = Gosu::Image.new("media/smoke.png");
    @vel_x = rand(-2 .. 2)  # rand(1.2 .. 2.0)
    @vel_y = rand(-2 .. 2)
    @angle = 0.0
    @trigger = false
    @count = count
    @change_image = Gosu::Image.new("media/smoke.png", :tileable =>false)
    # @time_to_flash = nil
# @flash_duration = 1 # 1 second as smoke


  # replace hard coded values with global constants:

    @x = rand * SCREEN_WIDTH
    @y = rand * SCREEN_HEIGHT
    @score = 0
  end


def move food
  food.x += food.vel_x
  food.x %= SCREEN_WIDTH
  food.y += food.vel_y
  food.y %=SCREEN_HEIGHT

  if (food.trigger ==false)
  if Gosu.milliseconds/100 % rand(1..1000) ==1 #random time to become smoke
    food.trigger = true
    food.count = 20    #number of steps for smoke to stay
    end
  else

    if food.count > 0
    food.count = food.count - 1 #
    else
    food.trigger = false #smoke time up, new image ad triggered
    #after smoke time up, velocity will change for x and y
    food.vel_x = rand(-2..2)
    food.vel_y = rand(-2..2)
  end
  end
end


```

```ruby
def draw_food food#if true then changes food to smoke
    if food.trigger
        food.change_image.draw_rot(@x, @y, ZOrder::FOOD, @angle)
    else
        food.image.draw_rot(@x, @y, ZOrder::FOOD, @angle)
    end
end
end
#draw smoke here
#def draw_smoke(x,y)
#@smoke.draw(x, y, ZOrder::FOOD, 1.5, 1.5)
#@frame -= 1
#end



 # def draw_food food

 # if food.trigger == true

  # draw_smoke(food.x,food.y)
  # food.trigger = false

 # else
 # food.image.draw_rot(food.x, food.y, ZOrder::FOOD, food.angle)
  # end
 # end
# end

class FoodHunterGame < (Example rescue Gosu::Window)
  def initialize

    # replace hard coded values with global constants:

    super SCREEN_WIDTH, SCREEN_HEIGHT
    self.caption = "Food Hunter Game"


@start_time = Gosu.milliseconds

    @background_image = Gosu::Image.new("media/space.png", :tileable => true)
    @smoke = Gosu::Image.new("media/smoke.png", :tileable => true)

    @all_food = Array.new

    # Food is created later in generate-food

    @player = Hunter.new(:icecream)

    warp(@player, SCREEN_WIDTH/2, SCREEN_HEIGHT/2)

    @font = Gosu::Font.new(20)
    @frame = 0
```

```ruby
  end

  def update        #maybe add a function for random case then smoke and if smoke then sleep for some time and then re call move food?

    # For key mappings see https://www.libgosu.org/cpp/namespace_gosu.html#enum-members

    if Gosu.button_down? Gosu::KB_LEFT or Gosu.button_down? Gosu::GP_LEFT
      move_left @player
    end
    if Gosu.button_down? Gosu::KB_RIGHT or Gosu.button_down? Gosu::GP_RIGHT
      move_right @player
    end
    if Gosu.button_down? Gosu::KB_UP or Gosu.button_down? Gosu::GP_BUTTON_0
      move_up @player
    end
    if Gosu.button_down? Gosu::KB_DOWN or Gosu.button_down? Gosu::GP_BUTTON_9
      move_down @player
    end

     @all_food.each {  |food| food.move food }
    # if rand(400) == 0
# if rand(2) == 0
 #it randomises the speed and direction of object, then it calls the if trigger= true part now once random number is equal to 0
 #oh ok np
 #but smoke still loads
 #and object velocity and direction still changes as excepted
 #only problem is duration of smoke
 # @trigger = true
# food.vel_x = -food.vel_x
# food.vel_y = -food.vel_y
# end

# end









    self.remove_food

    collect_food(@all_food, @player)

    # the following will generate new food randomly as update is called each timestep


   if rand(100) < 2 and @all_food.size < 4
      @all_food.push(generate_food)
   end
```

```
258
259     # change the hunted food randomly:
260     if rand(400) == 0
261       change = rand(4)
262       case change
263        when 0
264          set_hunted(@player, :icecream)
265        when 1
266          set_hunted(@player, :chips)
267        when 2
268          set_hunted(@player, :burger)
269        when 3
270          set_hunted(@player, :pizza)
271       end
272
273
274
275     end
276   end
277
278 # #draw smoke here
279 # def draw_smoke(x,y)
280 # @smoke.draw(x, y, ZOrder::FOOD, 1.5, 1.5)
281 # @frame -= 1
282 # end
283
284   def draw
285     @background_image.draw(0, 0, ZOrder::BACKGROUND)
286     draw_hunter @player
287     @all_food.each { |food| food.draw_food food}
288     @font.draw("Score: #{@player.score}", 10, 10, ZOrder::UI, 1.0, 1.0,
      ↪  Gosu::Color::YELLOW)
289   end
290
291   def generate_food
292    case rand(4)
293     when 0
294       Food.new("media/Chips.png", :chips)
295     when 1
296       Food.new("media/Burger.png", :burger)
297     when 2
298       Food.new("media/IceCream.png", :icecream)
299     when 3
300       Food.new("media/Pizza.png", :pizza)
301       end
302   end
303
304
305   def remove_food
306     @all_food.reject! do |food|
307       # Replace the following hard coded values with global constants:
308       if food.x > SCREEN_WIDTH || food.y > SCREEN_HEIGHT  || food.x < 0 || food.y <
       ↪  0
```

```
309         true
310       else
311         false
312       end
313     end
314   end
315
316   def button_down(id)
317     if id == Gosu::KB_ESCAPE
318       close
319     end
320     end
321 end
322
323 FoodHunterGame.new.show if __FILE__ == $0
```


Score: -1

Faculty of Science, Engineering and Technology

Introduction to Programming

Distinction Task 8.3: Food Hunter

Principles question and answer sheet.

Write up to one page (maximum 500 words) addressing the following question:

During this course we have covered some design principles for programming. These include:

- Coupling
- Cohesion
- Information hiding
- Modular design

Among others.

Compare and contrast how at least two (2) of the structured programming principles we have covered relate to theory and practice of object oriented programming. i.e what are the similarities and/or differences between structured programming and object oriented programming in relation to at least 2 design principles,

Answer:

Structured Programming is basically a programming paradigm that improves the quality by making extensive use of control structures like sequence, selection, etc. (focuses mainly on making procedures, functions and code parts).

On the other hand, object oriented programming is a different programming paradigm that focuses on the concepts of objects which can contain data and code instead. (focuses on making objects that contain both data and function).

Since they are both following different programming paradigms, it's quite obvious that their programming principles will also be quite different. So here, I'll compare and contrast between the 2 structured principle [a) information hiding, and b) modular design] with respect to object oriented programming.

a) **Information hiding** is a way to separate design decisions in a program that may change, and thus prevents other parts of the program from being altered alongside it.

   In structured program, it is done by 2 ways:

   1) using black box method, where internal components (code parts) are hidden and the focus is on how to use rather than how it works
   2) Information hiding, where the data parts are hidden and only accessed when needed.

   Similar thing can be achieved in OOP by using encapsulation by making use of 3 access specifiers of class:

   1) public (no access restriction),
   2) private (only be used by members and friends of that class)
   3) protected (used by members and friends of class and also members and friends derived from that class)

b) **Modular design** is a way to separate a big program into smaller, independent parts that, usually, contains everything needed to execute a specific, small task, in the entire program. It increases the functionality (as programmer can easily say what it does) and also reusability of that part of the code (as it is independent from the rest)

In structured program, we go through the entire code and mark out functions that do a specific task (like only print certain album or something). Then we remove the function from the main code into a block outside main code and link it to the main via the function name.

In object oriented programming, it is done in a similar way, except here, the program is divided into objects and entities. Thus it is easier to reuse as we don't need to worry about the linking mechanism between the modules (unlike in case of structural programming)

End of Task.

28 Fix-it

Fix a Ruby program.

Outcome	Weight
Functional Decomposition	♦♦♦◊◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦◊◊◊◊

Outcome	Weight
Apply Reading Techniques	♦♦♦♦♦

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Fix-it

Submitted By:
S M Ragib Rezwan
103172423
2021/05/09 20:22

Tutor:
NAJAM NAZAR

May 9, 2021

```ruby
# takes a number and writes that number to a file then on each line
# increments from zero to the number passed
def write(aFile, number)

  aFile.puts(number)

  index = 0
  while (index < number)
   aFile.puts(index.to_s)
   index += 1
  end
end

# Read the data from the file and print out each line
def read(aFile)

  # Defensive programming:
  count = aFile.gets.to_i
  validate (count)


  index = 0
  while (index < count)
    line = aFile.gets
    puts "Line read: " + line
    index += 1
  end
end

#called a new function for validation to improve modularity
def validate(count)
    if (is_numeric?(count))
    count = count.to_i
  else
    count = 0
    puts "Error: first line of file is not a number"
  end
  return count
end


# returns true if a string contains only digits
def is_numeric?(obj)
  if /^[0-9]+$/.match(obj.to_s) == nil
  #we know match cant be nil if it is number. so if it comes nil then it is false!
    false
  else
  true
  end
end

# Write data to a file then read it in and print it out
def main
```

```
54    aFile = File.new("mydata.txt", "w") # open for writing
55    write(aFile, 10)
56    aFile.close
57    #must close, or else it will read from 2nd line  instead of starting line!
58
59   bFile = File.new("mydata.txt", "r")
60
61    read(bFile)
62
63    bFile.close
64  end
65
66
67
68  main
```

```
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 9\9.1T-resources\Resources>fix_it.rb
Line read: 0
Line read: 1
Line read: 2
Line read: 3
Line read: 4
Line read: 5
Line read: 6
Line read: 7
Line read: 8
Line read: 9

C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 9\9.1T-resources\Resources>
```

29 Distinction Custom Code

A custom program option (instead of the D level Music Player)

Outcome	Weight
Functional Decomposition	♦♦♦♦♦

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦♦♦♦♦

Outcome	Weight
Apply Reading Techniques	♦♦♦♦♦

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Distinction Custom Code

Submitted By:
S M Ragib Rezwan
103172423
2021/05/31 14:37

Tutor:
NAJAM NAZAR

May 31, 2021

```ruby
require 'rubygems'
require 'gosu'

WIDTH, HEIGHT = 500, 500

module Tiles
  Land = 0
  Rock = 1

end

# Map class holds and draws tiles and gems.
class GameMap
  attr_accessor :width, :height, :orb, :spike, :enemy, :tile_set, :tiles
end

# Player class(scope benefit)
class Player
  attr_accessor :x, :y, :dir, :vy, :game_map, :standing, :walk1, :walk2, :jump,
   ↪  :current_image
end

#orb class(scope benefit)
class Orb
  attr_accessor :x, :y, :image
end

#spike class(scope benefit)
class Spike
  attr_accessor :x, :y, :image
end

#enemy (ie the mimic) class (scope benefit)
class Enemy
  attr_accessor :x, :y, :image
end

def setup_game_map(filename)
#taken from Captain Ruby code with slight
 ↪  modification
#basically making map understand that it should take that symbol as position of
 ↪  that specific obj
  game_map = GameMap.new

  game_map.tile_set = Gosu::Image.load_tiles("image/tileset0.png", 50, 50,
   ↪  :tileable => true)


  orb = Gosu::Image.new("image/orb1.png")
    game_map.orb = []
    enemy = Gosu::Image.new("image/enemyflying.png") #add a different pic here
    game_map.enemy = []

```

```
50      #calling spike images in the same way player was called
51      spike_up, spike_left, spike_right, spike_down =
        ↪ Gosu::Image.load_tiles("image/spike_all2.png", 50, 50)
52
53
54    game_map.spike = []
55
56    lines = File.readlines(filename).map { |line| line.chomp }
57    game_map.height = lines.size
58    game_map.width = lines[0].size
59    game_map.tiles = Array.new(game_map.width) do |x|
60      Array.new(game_map.height) do |y|
61        case lines[y][x, 1]
62        when '"'
63          Tiles::Land
64        when '#'
65          Tiles::Rock
66          when '+'
67          game_map.spike.push(Spike_Data(spike_up, x * 50 + 25, y * 50 + 25))
68          nil
69          when 'l'
70          game_map.spike.push(Spike_Data(spike_left, x * 50 + 25, y * 50 + 25))
71          nil
72          when 'r'
73          game_map.spike.push(Spike_Data(spike_right, x * 50 + 25, y * 50 + 25))
74          nil
75          when '-'
76          game_map.spike.push(Spike_Data(spike_down, x * 50 + 25, y * 50 + 25))
77          nil
78        when 'x'
79          game_map.orb.push(Orb_Data(orb, x * 50 + 25, y * 50 + 25))
80          nil
81          when 'c'
82          game_map.enemy.push(Enemy_Data(enemy, x * 50 + 25, y * 50 + 25))
83          nil
84        else
85          nil
86        end
87      end
88    end
89    game_map
90  end
91
92  def draw_game_map(game_map)     #taken from Captain Ruby code with slight
    ↪ modification                                                          #
93    # Very primitive drawing function:
94    # Draws all the tiles, some off-screen, some on-screen.
95    game_map.height.times do |y|
96      game_map.width.times do |x|
97        tile = game_map.tiles[x][y]
98        if tile
99          #made all 50px, so removed overlap from captian ruby's one as redundant now
100         game_map.tile_set[tile].draw(x * 50 , y * 50 , 0)
```

```ruby
      end
    end
  end
  #drawing out all of them
  game_map.orb.each { |c| orb_spin(c) }
  game_map.spike.each { |c| spike_draw(c) }
  game_map.enemy.each { |c| enemy_draw(c) }

end


#taken from Captain Ruby code
# Solid at a given pixel position
def solid?(game_map, x, y)
  y < 0 || game_map.tiles[x / 50][y / 50]
end



#taken from Captain Ruby code
def setup_player(player, game_map, x, y)
  player = Player.new()
  player.x = x
  player.y = y
  player.dir = :left
  player.vy = 0 # Vertical velocity
  player.game_map = game_map

  # Load all animation frames
  player.standing, player.walk1, player.walk2, player.jump =
  ↪ Gosu::Image.load_tiles("image/bot5.png", 50, 50)
  # This always points to the frame that is currently drawn.
  # This is set in update, and used in draw.
  player.current_image = player.standing
  player
end

#taken from Captain Ruby code
def draw_player(player)
  # Flip vertically when facing to the left.
  if player.dir == :left
    offs_x = -25
    factor = 1.0
  else
    offs_x = 25
    factor = -1.0
  end
  player.current_image.draw(player.x + offs_x, player.y - 49, 0, factor, 1.0)
end

#taken from Captain Ruby code
# Could the object be placed at x + offs_x/y + offs_y without being stuck?
def would_fit(player, offs_x, offs_y)
```

```
153   # Check at the center/top and center/bottom for game_map collisions
154   not solid?(player.game_map, player.x + offs_x, player.y + offs_y) and
155     not solid?(player.game_map, player.x + offs_x, player.y + offs_y - 45)
156 end
157
158 #taken from Captain Ruby code
159 def update_player(player, move_x)
160   # Select image depending on action
161
162   if (move_x == 0)
163     player.current_image = player.standing
164   else
165     player.current_image = (Gosu.milliseconds / 175 % 2 == 0) ? player.walk1 :
      ↪  player.walk2
166   end
167   if (player.vy < 0)
168     player.current_image = player.jump
169   end
170
171   # Directional walking, horizontal movement
172   if move_x > 0
173     player.dir = :right
174     move_x.times { if would_fit(player, 1, 0) then player.x += 1 end }
175   end
176   if move_x < 0
177     player.dir = :left
178     (-move_x).times { if would_fit(player, -1, 0) then player.x -= 1 end }
179   end
180
181   # Acceleration/gravity
182   # By adding 1 each frame, and (ideally) adding vy to y, the player's
183   # jumping curve will be the parabole we want it to be.
184   player.vy += 1
185   # Vertical movement
186   if player.vy > 0
187     player.vy.times { if would_fit(player, 0, 1) then player.y += 1 else player.vy
      ↪  = 0 end }
188   end
189   if player.vy < 0
190     (-player.vy).times { if would_fit(player, 0, -1) then player.y -= 1 else
      ↪  player.vy = 0 end }
191   end
192 end
193
194 #taken from Captain Ruby code
195 def try_to_jump(player)
196   if solid?(player.game_map, player.x, player.y + 1)
197     player.vy = -18
198   end
199 end
200
201
202 def Enemy_Data(image, x, y)
```

```
203   enemy = Enemy.new()
204   enemy.image = image
205   enemy.x = x
206   enemy.y = y
207   enemy
208 end
209 
210 def enemy_draw(enemy)
211   enemy.image.draw_rot(enemy.x, enemy.y, 0, 0)
212 end
213 
214 #set up the enemy ai in here. It is a decision tree and probabilistic algorithm
    ↪  merged together
215 def enemy_move(player, enemy)
216   enemy.each do |c|
217         if (c.x < player.x)
218             c.x += rand(0.0..3.0)
219         elsif(c.x > player.x)
220             c.x -= rand(0.0..3.0)
221         else
222             c.x = 0.0
223         end
224 
225 
226         if (c.y < player.y)
227             c.y += rand(0.0..3.0)
228         elsif(c.y > player.y)
229             c.y -= rand(0.0..3.0)
230         else
231             c.y = 0.0
232         end
233 
234   end
235 
236 end
237 
238 #This is an example of FSM type AI
239 def enemy_touch(player, enemy)#reduced enemy hitbox to make it easier for player to
    ↪  dodge it
240   enemy.each do |c|
241       if ((c.x - player.x).abs < 25 and (c.y - player.y).abs < 25)
242 
243           @trigger = false
244           @trigger_Enemy_Death = true
245       end
246   end
247 end
248 
249 
250 
251 def Spike_Data(image, x, y)
252   spike = Spike.new()
253   spike.image = image
```

```
254   spike.x = x
255   spike.y = y
256   spike
257 end
258
259 def spike_draw(spike)
260   spike.image.draw_rot(spike.x, spike.y, 0, 0)
261 end
262
263
264 #This is an example of FSM type AI, even though object is stationary here!
265 def spike_touch(player, spike)#reduced spike hitbox to make it more playable
266   spike.each do |c|
267       if ((c.x - player.x).abs < 40 and (c.y - player.y).abs < 40)
268
269           @trigger = false
270           @trigger_Spike_Death = true
271       end
272   end
273 end
274
275
276 def Orb_Data(image, x, y)
277   orb = Orb.new()
278   orb.image = image
279   orb.x = x
280   orb.y = y
281   orb
282 end
283
284 def orb_spin(orb)#give different effect by playing around
285   # Draw, slowly rotating
286   #math.cos mainly gives multiple values making it rotate to different degrees
287   orb.image.draw_rot(orb.x, orb.y, 0, 25 * Math.cos(Gosu.milliseconds / 133.7))
288 end
289
290 def orb_gone(player, orb)
291   # Same as in the tutorial game.
292   #checksif object x and y values match with character or not. if matches then
      ↪  removes object from array "orb"
293   orb.reject! do |c|
294     (c.x - player.x).abs < 50 and (c.y - player.y).abs < 50
295
296   end #end for do condition
297
298   if (orb[0] == nil)
299       @trigger = false
300       @trigger_endscreen = true
301
302   #these two lines are for testing stuff:
303   # puts "testing nothing remains"
304   # puts "#{@time_now}"
305
```

```ruby
  end

end


#draws all start screen elements
def draw_start_screen_elements(z_order)
    start_button_elements(z_order)
    levels(z_order)
    quit_button_elements(z_order)

    #loads up all the levels when clicked on "level"
    if (@trigger_level==true)
        level_1_button_elements(z_order)
        level_2_button_elements(z_order)
        level_3_button_elements(z_order)
        level_4_button_elements(z_order)
        level_impossible_button_elements(z_order)
    end
end

def quit_button_elements(z_order)
    # Draw the button
    Gosu.draw_rect(50, 450, 80, 30, Gosu::Color::BLACK, z_order, mode=:default)

    #hover technique from hover button
    if  mouse_over_button_quit(mouse_x, mouse_y)
        Gosu.draw_rect(45, 445, 90, 40, Gosu::Color::WHITE, z_order - 1,
        ↪  mode=:default)
    end

    # Draw the button text
    @button_start_font.draw("QUIT", 60, 460, z_order, 1.0, 1.0, Gosu::Color::WHITE)
end

def start_button_elements(z_order)
    # Draw the button
    Gosu.draw_rect(50, 250, 80, 30, Gosu::Color::BLACK, z_order, mode=:default)

    #hover technique from hover button
    if  mouse_over_button_start(mouse_x, mouse_y)
        Gosu.draw_rect(45, 245, 90, 40, Gosu::Color::WHITE, z_order - 1,
        ↪  mode=:default)
    end

    # Draw the button text
    @button_start_font.draw("DEMO", 60, 260, z_order, 1.0, 1.0, Gosu::Color::WHITE)
end

def levels(z_order)
    # Draw the button
    Gosu.draw_rect(50, 305, 90, 30, Gosu::Color::BLACK, z_order, mode=:default)

```

```ruby
    #hover technique from hover button
    if  mouse_over_button_level(mouse_x, mouse_y)
        Gosu.draw_rect(45, 300, 100, 40, Gosu::Color::WHITE, z_order-1,
        ↪  mode=:default)
    end

    # Draw the button text
    @button_start_font.draw("LEVELS:", 60, 310, z_order, 1.0, 1.0,
    ↪  Gosu::Color::WHITE)
end


def level_1_button_elements(z_order)
    # Draw the button
    Gosu.draw_rect(50, 350, 80, 30, Gosu::Color::BLACK, z_order, mode=:default)

    #hover technique from hover button
    if  mouse_over_button_level_1(mouse_x, mouse_y)
        Gosu.draw_rect(45, 345, 90, 40, Gosu::Color::WHITE, z_order-1,
        ↪  mode=:default)
    end

    # Draw the button text
    @button_start_font.draw("Level 1", 60, 360, z_order, 1.0, 1.0,
    ↪  Gosu::Color::WHITE)
end

def level_2_button_elements(z_order)
    # Draw the button
    Gosu.draw_rect(160, 350, 80, 30, Gosu::Color::BLACK, z_order, mode=:default)

    #hover technique from hover button
    if  mouse_over_button_level_2(mouse_x, mouse_y)
        Gosu.draw_rect(155, 345, 90, 40, Gosu::Color::WHITE, z_order-1,
        ↪  mode=:default)
    end

    # Draw the button text
    @button_start_font.draw("Level 2", 170, 360, z_order, 1.0, 1.0,
    ↪  Gosu::Color::WHITE)
end

def level_3_button_elements(z_order)
    # Draw the button
    Gosu.draw_rect(270, 350, 80, 30, Gosu::Color::BLACK, z_order, mode=:default)

    #hover technique from hover button
    if  mouse_over_button_level_3(mouse_x, mouse_y)
        Gosu.draw_rect(265, 345, 90, 40, Gosu::Color::WHITE, z_order-1,
        ↪  mode=:default)
    end

    # Draw the button text
```

```
403     @button_start_font.draw("Level 3", 280, 360, z_order, 1.0, 1.0,
        ↪ Gosu::Color::WHITE)
404 end
405
406 def level_4_button_elements(z_order)
407     # Draw the button
408     Gosu.draw_rect(380, 350, 80, 30, Gosu::Color::BLACK, z_order, mode=:default)
409
410     #hover technique from hover button
411     if  mouse_over_button_level_4(mouse_x, mouse_y)
412         Gosu.draw_rect(375, 345, 90, 40, Gosu::Color::WHITE, z_order-1,
            ↪ mode=:default)
413     end
414
415     # Draw the button text
416     @button_start_font.draw("Level 4", 390, 360, z_order, 1.0, 1.0,
        ↪ Gosu::Color::WHITE)
417 end
418
419 def level_impossible_button_elements(z_order)
420     # Draw the button
421     Gosu.draw_rect(160, 400, 170, 30, Gosu::Color::BLACK, z_order, mode=:default)
422
423     #hover technique from hover button
424     if  mouse_over_button_level_impossible(mouse_x, mouse_y)
425         Gosu.draw_rect(150, 395, 190, 40, Gosu::Color::WHITE, z_order-1,
            ↪ mode=:default)
426     end
427
428     # Draw the button text
429     @button_start_font.draw("IMPOSSIBLE LEVEL", 160, 410, z_order, 1.0, 1.0,
        ↪ Gosu::Color::WHITE)
430 end
431
432 #these load the level timer and orb counter elements
433 def    draw_level_extra_elements(z_order)
434     draw_level_timer(z_order)
435     draw_orb_counter(z_order)
436 end
437
438 def draw_level_timer(z_order)
439     @sky.draw 0, 0, 0
440     Gosu.draw_rect(0, 0, 175, 20, Gosu::Color::BLACK, z_order-1, mode=:default)
441     @end_font.draw("Timer: #{@time_now.round(1)} seconds", 5, 0, z_order, 1.0, 1.0,
        ↪ Gosu::Color::WHITE)
442 end
443
444 def draw_orb_counter(z_order)
445     Gosu.draw_rect(195, 0, 105, 20, Gosu::Color::BLACK, z_order -1, mode=:default)
446     @end_font.draw("Orbs left: #{@game_map.orb.length}", 200, 0, z_order, 1.0, 1.0,
        ↪ Gosu::Color::WHITE)
447 end
448
```

```ruby
#makes sure it draws these set of objects for victory screen
def draw_victory_end(z_order)
    @end_font.draw("GAME OVER! ", 60, 30, z_order, 1.0, 1.0, Gosu::Color::WHITE)
    @end_font.draw("Time Taken: #{@time_now.round(1)} seconds!", 60, 50, z_order, 1.0, 1.0, Gosu::Color::WHITE)
    return_button_elements(z_order)

    #gives out certain comments to player's gameplay depending on time taken by him or her
    if(@time_now.round(1) < 20)
        @end_font.draw("You know the map too well. Are you cheating? ", 60, 80, z_order, 1.0, 1.0, Gosu::Color::WHITE)
    end
    if(@time_now.round(1) > 20 && @time_now.round(1) < 60)
        @end_font.draw("Excellent! ", 60, 80, z_order, 1.0, 1.0, Gosu::Color::WHITE)
    end
    if(@time_now.round(1) > 60 && @time_now.round(1) < 80)
        @end_font.draw("Good Effort! But I know you can do better!", 60, 80, z_order, 1.0, 1.0, Gosu::Color::WHITE)
    end
    if(@time_now.round(1) > 80 && @time_now.round(1) < 100)
        @end_font.draw("You are very slow", 60, 80, z_order, 1.0, 1.0, Gosu::Color::WHITE)
    end
    if(@time_now.round(1) > 100)
        @end_font.draw("Are you sure you are playing the game?", 60, 80, z_order, 1.0, 1.0, Gosu::Color::WHITE)
    end

    @character_win.draw(0, 0, z_order)

end

#makes sure it draws these set of objects for killed by spike screen
def draw_defeat_spike_end(z_order)
    @end_font.draw("GAME OVER! ", 60, 60, z_order, 1.0, 1.0, Gosu::Color::WHITE)
    @end_font.draw("You have been killed by spikes!!", 60, 80, z_order, 1.0, 1.0, Gosu::Color::WHITE)
    return_button_elements(z_order)
    @character_dead.draw(0, 0, z_order)
end

#makes sure it draws these set of objects for killed by enemy (MIMIC) screen
def draw_defeat_enemy_end(z_order)
    @end_font.draw("GAME OVER! ", 60, 60, z_order, 1.0, 1.0, Gosu::Color::WHITE)
    @end_font.draw("You have been killed by MIMIC!!", 60, 80, z_order, 1.0, 1.0, Gosu::Color::WHITE)
    return_button_elements(z_order)
    @character_dead.draw(0, 0, z_order)
end

#draws return button elements on all end screens
def return_button_elements(z_order_level)
```

```ruby
    # Draw the button
    Gosu.draw_rect(55, 110, 140, 30, Gosu::Color::BLACK, z_order_level, mode=:default)

    #hover technique from hover button
    if  mouse_over_button_end(mouse_x, mouse_y)
        Gosu.draw_rect(50, 105, 150, 40, Gosu::Color::WHITE, z_order_level-1, mode=:default)
    end

    # Draw the button text
    @button_start_font.draw("Return Home", 60, 110, z_order_level, 1.0, 1.0, Gosu::Color::WHITE)
end




class CollectOrbGame < (Example rescue Gosu::Window)
  def initialize
  #all triggers and objects are initialized here and each new space means a different object from a different group. I didnt put them in different functions here as i want to keep all triggers and initialize all objetcs in the same place.
    super WIDTH, HEIGHT

    self.caption = "Collect all orbs!"

    #all start screen object and z orders
    @start_screen = Gosu::Image.new("image/start1.png")
    @button_start_font = Gosu::Font.new(20)
    @start_screen_z_order = 1
    @start_button_z_order = 3

    @trigger_map = true
    #for background picture during level play through
    @sky = Gosu::Image.new("image/background2.jpg")

    if (@trigger_map == true)
    @game_map = setup_game_map("image/gamemapdemo.txt")    #kept it so that things all load smth at first. This will get overridden when other levels load

    #puts "demo map start" #testing level loads or not
    end


    @bot = setup_player(@bot, @game_map, 2500, 950)
    #kept it so that player things all load smth at first, this will get overridden when other levels load

    #all level triggers
    @trigger_level_1 = false
    @trigger_level_2 = false
```

```ruby
    @trigger_level_3 = false
    @trigger_level_4 = false
    @trigger_level_impossible = false
    @trigger_level_demo = false

    @camera_x = @camera_y = 0

    #for the z level of timer and orb remaining number
    @counter_display_information_z_order = -1


    @trigger_level=false #for "click level at start to show all level options"
    @trigger = false    #all @trigger is for timer and also enemy movement
     ↪  generation

    #triggers for all endscrren variations and elements
    @trigger_endscreen = false
    @trigger_Spike_Death = false
    @trigger_Enemy_Death = false
    @trigger_return_home = false

    #z_order level for all end screen elements
    @end_z_order = -1
    @end_z_order_victory = -1
    @end_z_order_spike_death = -1
    @end_z_order_enemy_death = -1

    #end screen font size
    @end_font = Gosu::Font.new(20)

    #endscreen images
    @character_dead = Gosu::Image.new("image/character_death3.png")
    @character_win = Gosu::Image.new("image/character_win.png")

    #time counter initialized. I used this as I far more custom control over it.
    @time_now = 0

    #all song elements used in his game
    @song = Gosu::Song.new("image/game_background_music2.wav")
    @start_song = false    #says whether song is not played for not

  end

 def update
    if(@trigger_level_demo ==true)
        @game_map = setup_game_map("image/gamemapdemo.txt")
        @bot = setup_player(@bot, @game_map, 2500, 950)
        @trigger_map = false

        @trigger_return_home = false
        @trigger_level_demo =false
        #puts "demo map select"    #testing level loads or not
    end
```

```
591
592     if(@trigger_level_1 ==true)
593         @game_map = setup_game_map("image/gamemap1.txt")
594         @bot = setup_player(@bot, @game_map, 200, 1100)
595         @trigger_map = false
596
597         @trigger_return_home = false
598         #puts "lvl1 map select"    #testing level loads or not
599         @trigger_level_1 =false
600     end
601
602     if(@trigger_level_2 ==true)
603         @game_map = setup_game_map("image/gamemap2.txt")
604         @bot = setup_player(@bot, @game_map, 200, 850)
605         @trigger_map = false
606
607         @trigger_return_home = false
608         #puts "lvl2 map select"    #testing level loads or not
609         @trigger_level_2 =false
610     end
611
612     if(@trigger_level_3 ==true)
613         @game_map = setup_game_map("image/gamemap3.txt")
614         @bot = setup_player(@bot, @game_map, 1500, 950)
615         @trigger_map = false
616
617         @trigger_return_home = false
618         #puts "lvl3 map select"    #testing level loads or not
619         @trigger_level_3 =false
620     end
621
622     if(@trigger_level_4 ==true)
623         @game_map = setup_game_map("image/gamemap4.txt")
624         @bot = setup_player(@bot, @game_map, 2500, 950)
625         @trigger_map = false
626
627         @trigger_return_home = false
628         #puts "lvl4 map select"    #testing level loads or not
629         @trigger_level_4 =false
630     end
631
632     if(@trigger_level_impossible ==true)
633         @game_map = setup_game_map("image/gamemapimpossible.txt")
634         @bot = setup_player(@bot, @game_map, 2500, 950)
635         @trigger_map = false
636
637         @trigger_return_home = false
638         #puts "lvl5 map select"    #testing level loads or not
639         @trigger_level_impossible =false
640     end
641     #player movement data. I made it so that player can move faster then enemy
      ↪  movement in order to give the player a better chance of surviving
642     move_x = 0
```

```
643     move_x -= 5 if Gosu.button_down? Gosu::KB_LEFT
644     move_x += 5 if Gosu.button_down? Gosu::KB_RIGHT
645     update_player(@bot, move_x)
646 
647     #objects and player interaction
648     orb_gone(@bot, @game_map.orb)#reject orb inside this
649     spike_touch(@bot, @game_map.spike)#spike kill condition inside this
650     enemy_touch(@bot, @game_map.enemy)#enemy kill condition inside this
651 
652 
653 
654     # Scrolling follows player
655     #needed for camera translation in draw part
656     @camera_x = [[@bot.x - WIDTH / 2, 0].max, @game_map.width * 50 - WIDTH].min
657     @camera_y = [[@bot.y - HEIGHT / 2, 0].max, @game_map.height * 50 - HEIGHT].min
658 
659 
660     if (@trigger == true)#for all levels
661         #move enemy that spawned with map
662         enemy_move(@bot, @game_map.enemy)
663 
664         @time_now += 0.0175 #(will now give second's value)
665 
666         #these will remove all start menu objects
667         @start_button_z_order = -1
668         @start_screen_z_order = -1
669 
670         #these will load all game counter objects
671         @counter_display_information_z_order = 2
672 
673         #this will trigger the play song
674         @start_song = true
675     end
676 
677     #plays song in a loop
678     if(@start_song == true)
679         @song.play(looping = true)
680     end
681 
682     if(@trigger_endscreen == true)
683         #these will load all victory screen objects
684         @end_z_order = 3
685         @end_z_order_victory = 4
686 
687         @trigger =false
688 
689         #stops song
690         @song.stop
691     end
692 
693     if(@trigger_Spike_Death == true)
694         #these will load all spike death screen objects
695         @end_z_order = 3
```

```
696         @end_z_order_spike_death = 4
697
698         @trigger =false
699
700         #stops song
701         @song.stop
702     end
703
704     if(@trigger_Enemy_Death == true)
705         #these will load all spike death screen objects
706         @end_z_order = 3
707         @end_z_order_enemy_death = 4
708
709         @trigger =false
710
711         #stops song
712         @song.stop
713     end
714
715     if (@trigger_return_home==true)
716         #makes everything else false, and z_order -1
717         #makes z_order to start screen normal now
718         @trigger_endscreen = false
719         @trigger_Spike_Death = false
720         @trigger_Enemy_Death = false
721
722         #loads up all start screen objects
723         @start_screen_z_order = 1
724         @start_button_z_order = 3
725
726         #remove unnecesary objects
727         @counter_display_information_z_order = -1
728
729         @end_z_order = -1
730         @end_z_order_victory = -1
731         @end_z_order_spike_death = -1
732         @end_z_order_enemy_death = -1
733
734         #stops song
735         @song.stop
736
737         #resets time counter
738         @time_now = 0
739     end
740
741
742   end
743
744
745
746
747   def draw
748     @start_screen.draw 0, 0, @start_screen_z_order
```

```
749
750     draw_start_screen_elements(@start_button_z_order)
751     draw_level_extra_elements(@counter_display_information_z_order)
752
753     #this part prints camera position, for testing stuff
754     # puts "Camera X is #{@camera_x}"
755     # puts "Camera Y is #{@camera_y}"
756
757     Gosu.draw_rect(0, 0, WIDTH, HEIGHT, Gosu::Color::BLACK, @end_z_order,
        ↪  mode=:default)
758
759     draw_victory_end(@end_z_order_victory)    #called if all orb collected
760     draw_defeat_spike_end(@end_z_order_spike_death)    #called if death to spike
761     draw_defeat_enemy_end(@end_z_order_enemy_death)    #called if killed by
        ↪  enemy(ie the mimic)
762
763
764     #this is for testing stuff to see the timer and to help finetune it
765     #puts "time from start is #{@time_now.round(2)} seconds"
766
767
768     Gosu.translate(-@camera_x, -@camera_y) do
769       draw_game_map(@game_map)
770       draw_player(@bot)
771     end
772   end
773
774     #all button clicking locations
775     def mouse_over_button_start(mouse_x, mouse_y)
776         if ((mouse_x >= 40 && mouse_x <= 135) && (mouse_y >= 245 && mouse_y <= 285))
777           true
778
779         else
780           false
781         end
782     end
783
784     def mouse_over_button_level_1(mouse_x, mouse_y)
785         if ((mouse_x >= 45 && mouse_x <= 135) && (mouse_y >= 345 && mouse_y <= 385))
786           true
787
788         else
789           false
790         end
791     end
792
793     def mouse_over_button_level_2(mouse_x, mouse_y)
794         if ((mouse_x >= 155 && mouse_x <= 245) && (mouse_y >= 345 && mouse_y <=
            ↪  385))
795           true
796
797         else
798           false
```

```
799         end
800     end
801
802     def mouse_over_button_level_3(mouse_x, mouse_y)
803         if ((mouse_x >= 265 && mouse_x <= 355) && (mouse_y >= 345 && mouse_y <=
        ↪  385))
804           true
805
806         else
807           false
808         end
809     end
810
811     def mouse_over_button_level_4(mouse_x, mouse_y)
812         if ((mouse_x >= 375 && mouse_x <= 465) && (mouse_y >= 345 && mouse_y <=
        ↪  385))
813           true
814
815         else
816           false
817         end
818     end
819
820     def mouse_over_button_level_impossible(mouse_x, mouse_y)
821         if ((mouse_x >= 150 && mouse_x <= 340) && (mouse_y >= 395 && mouse_y <=
        ↪  435))
822           true
823
824         else
825           false
826         end
827     end
828
829     def mouse_over_button_level(mouse_x, mouse_y)
830         if ((mouse_x >= 45 && mouse_x <= 145) && (mouse_y >= 300 && mouse_y <= 340))
831           true
832
833         else
834           false
835         end
836     end
837
838     def mouse_over_button_end(mouse_x, mouse_y)
839         if ((mouse_x >= 50 && mouse_x <= 200) && (mouse_y >= 105 && mouse_y <= 145))
840           true
841
842         else
843           false
844         end
845     end
846
847     def mouse_over_button_quit(mouse_x, mouse_y)
848         if ((mouse_x >= 45 && mouse_x <= 135) && (mouse_y >= 445 && mouse_y <= 485))
```

```
849           true
850
851         else
852           false
853         end
854       end
855
856   #gave z order as another restriction along side all button location to make sure
      ↪  only the button loaded with a specific z order will be selected and not the one
      ↪  behind it
857     def button_down(id)
858     case id
859     when Gosu::MsLeft
860         if mouse_over_button_start(mouse_x, mouse_y) && @start_button_z_order == 3
861           @trigger_level_demo = true
862           @trigger = true
863           @trigger_level = false
864         end
865         if mouse_over_button_level_1(mouse_x, mouse_y) && @start_button_z_order == 3
866           @trigger_level_1 = true
867           @trigger = true
868           @trigger_level = false
869         end
870          if mouse_over_button_level_2(mouse_x, mouse_y) && @start_button_z_order == 3
871           @trigger_level_2 = true
872           @trigger = true
873           @trigger_level = false
874         end
875         if mouse_over_button_level_3(mouse_x, mouse_y) && @start_button_z_order == 3
876           @trigger_level_3 = true
877           @trigger = true
878           @trigger_level = false
879         end
880         if mouse_over_button_level_4(mouse_x, mouse_y) && @start_button_z_order == 3
881           @trigger_level_4 = true
882           @trigger = true
883           @trigger_level = false
884         end
885         if mouse_over_button_level_impossible(mouse_x, mouse_y) &&
          ↪  @start_button_z_order == 3
886           @trigger_level_impossible = true
887           @trigger = true
888           @trigger_level = false
889         end
890         if mouse_over_button_level(mouse_x, mouse_y) && @start_button_z_order == 3
891           @trigger_level = true
892
893         end
894
895         if mouse_over_button_end(mouse_x, mouse_y) &&(@end_z_order_spike_death ==4 ||
          ↪  @end_z_order_enemy_death==4 || @end_z_order_victory==4)
896           @trigger_return_home = true
897           @trigger = false
```

```
898       end
899       if   mouse_over_button_quit(mouse_x, mouse_y) && @start_button_z_order == 3
900         close
901       end
902
903     when Gosu::KB_UP
904       try_to_jump(@bot)
905     when Gosu::KB_ESCAPE
906         close
907     else
908       super
909     end
910   end
911
912 end
913
914 CollectOrbGame.new.show if __FILE__ == $0
```


C:\Users\SAIFUL>C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\6.3D-custom program\game things\my game\game stuff\orb_collector_final
'C:\Users\SAIFUL\Desktop\swinburne' is not recognized as an internal or external command,
operable program or batch file.
C:\Users\SAIFUL>cd C:\Users\SAIFUL\Desktop\swinburne
al
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS1000
Volume in drive C has no label.
Volume Serial Number is BED8-11C3
Directory of C:\Users\SAIFUL\Desktop\swinburne class
05/31/2021 10:27 AM <DIR> .
05/31/2021 10:27 AM <DIR> ..
05/31/2021 10:26 AM <DIR> image
05/31/2021 10:19 AM 25,309 orb_collector.
1 File(s) 25,309 bytes
3 Dir(s) 8,553,603,072 bytes free
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS1000
DEPRECATION WARNING: draw is deprecated; use Font#dra
draw called from C:/Users/SAIFUL/Desktop/swinburne cl
/orb_collector.rb:351:in `start_button_elements'.
Collect all orbs!
Timer: 10.3 seconds
Orbs left: 7

30 High Distinction Custom Code

A high distinction level custom program

Outcome	Weight
Functional Decomposition	♦♦♦♦♦

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦♦♦♦♦

Outcome	Weight
Apply Reading Techniques	♦♦♦♦♦

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

High Distinction Custom Code

Submitted By:
S M Ragib Rezwan
103172423
2021/05/31 14:38

Tutor:
NAJAM NAZAR

May 31, 2021

```ruby
require 'rubygems'
require 'gosu'

WIDTH, HEIGHT = 500, 500

module Tiles
  Land = 0
  Rock = 1

end

# Map class holds and draws tiles and gems.
class GameMap
  attr_accessor :width, :height, :orb, :spike, :enemy, :tile_set, :tiles
end

# Player class(scope benefit)
class Player
  attr_accessor :x, :y, :dir, :vy, :game_map, :standing, :walk1, :walk2, :jump,
  ↪ :current_image
end

#orb class(scope benefit)
class Orb
  attr_accessor :x, :y, :image
end

#spike class(scope benefit)
class Spike
  attr_accessor :x, :y, :image
end

#enemy (ie the mimic) class (scope benefit)
class Enemy
  attr_accessor :x, :y, :image
end

def setup_game_map(filename)
#taken from Captain Ruby code with slight
↪ modification
#basically making map understand that it should take that symbol as position of
↪ that specific obj
  game_map = GameMap.new

  game_map.tile_set = Gosu::Image.load_tiles("image/tileset0.png", 50, 50,
  ↪ :tileable => true)


  orb = Gosu::Image.new("image/orb1.png")
    game_map.orb = []
    enemy = Gosu::Image.new("image/enemyflying.png") #add a different pic here
    game_map.enemy = []

```

```
50      #calling spike images in the same way player was called
51      spike_up, spike_left, spike_right, spike_down =
        ↪ Gosu::Image.load_tiles("image/spike_all2.png", 50, 50)
52
53
54    game_map.spike = []
55
56    lines = File.readlines(filename).map { |line| line.chomp }
57    game_map.height = lines.size
58    game_map.width = lines[0].size
59    game_map.tiles = Array.new(game_map.width) do |x|
60      Array.new(game_map.height) do |y|
61        case lines[y][x, 1]
62        when '"'
63          Tiles::Land
64        when '#'
65          Tiles::Rock
66          when '+'
67          game_map.spike.push(Spike_Data(spike_up, x * 50 + 25, y * 50 + 25))
68          nil
69          when 'l'
70          game_map.spike.push(Spike_Data(spike_left, x * 50 + 25, y * 50 + 25))
71          nil
72          when 'r'
73          game_map.spike.push(Spike_Data(spike_right, x * 50 + 25, y * 50 + 25))
74          nil
75          when '-'
76          game_map.spike.push(Spike_Data(spike_down, x * 50 + 25, y * 50 + 25))
77          nil
78        when 'x'
79          game_map.orb.push(Orb_Data(orb, x * 50 + 25, y * 50 + 25))
80          nil
81          when 'c'
82          game_map.enemy.push(Enemy_Data(enemy, x * 50 + 25, y * 50 + 25))
83          nil
84        else
85          nil
86        end
87      end
88    end
89    game_map
90  end
91
92  def draw_game_map(game_map)     #taken from Captain Ruby code with slight
    ↪ modification                                                          #
93    # Very primitive drawing function:
94    # Draws all the tiles, some off-screen, some on-screen.
95    game_map.height.times do |y|
96      game_map.width.times do |x|
97        tile = game_map.tiles[x][y]
98        if tile
99          #made all 50px, so removed overlap from captian ruby's one as redundant now
100         game_map.tile_set[tile].draw(x * 50 , y * 50 , 0)
```

```ruby
      end
    end
  end
  #drawing out all of them
  game_map.orb.each { |c| orb_spin(c) }
  game_map.spike.each { |c| spike_draw(c) }
  game_map.enemy.each { |c| enemy_draw(c) }

end


#taken from Captain Ruby code
# Solid at a given pixel position
def solid?(game_map, x, y)
  y < 0 || game_map.tiles[x / 50][y / 50]
end



#taken from Captain Ruby code
def setup_player(player, game_map, x, y)
  player = Player.new()
  player.x = x
  player.y = y
  player.dir = :left
  player.vy = 0 # Vertical velocity
  player.game_map = game_map

  # Load all animation frames
  player.standing, player.walk1, player.walk2, player.jump =
  ↪ Gosu::Image.load_tiles("image/bot5.png", 50, 50)
  # This always points to the frame that is currently drawn.
  # This is set in update, and used in draw.
  player.current_image = player.standing
  player
end

#taken from Captain Ruby code
def draw_player(player)
  # Flip vertically when facing to the left.
  if player.dir == :left
    offs_x = -25
    factor = 1.0
  else
    offs_x = 25
    factor = -1.0
  end
  player.current_image.draw(player.x + offs_x, player.y - 49, 0, factor, 1.0)
end

#taken from Captain Ruby code
# Could the object be placed at x + offs_x/y + offs_y without being stuck?
def would_fit(player, offs_x, offs_y)
```

```ruby
  # Check at the center/top and center/bottom for game_map collisions
  not solid?(player.game_map, player.x + offs_x, player.y + offs_y) and
    not solid?(player.game_map, player.x + offs_x, player.y + offs_y - 45)
end

#taken from Captain Ruby code
def update_player(player, move_x)
  # Select image depending on action

  if (move_x == 0)
    player.current_image = player.standing
  else
    player.current_image = (Gosu.milliseconds / 175 % 2 == 0) ? player.walk1 :
    ↪  player.walk2
  end
  if (player.vy < 0)
    player.current_image = player.jump
  end

  # Directional walking, horizontal movement
  if move_x > 0
    player.dir = :right
    move_x.times { if would_fit(player, 1, 0) then player.x += 1 end }
  end
  if move_x < 0
    player.dir = :left
    (-move_x).times { if would_fit(player, -1, 0) then player.x -= 1 end }
  end

  # Acceleration/gravity
  # By adding 1 each frame, and (ideally) adding vy to y, the player's
  # jumping curve will be the parabole we want it to be.
  player.vy += 1
  # Vertical movement
  if player.vy > 0
    player.vy.times { if would_fit(player, 0, 1) then player.y += 1 else player.vy
    ↪  = 0 end }
  end
  if player.vy < 0
    (-player.vy).times { if would_fit(player, 0, -1) then player.y -= 1 else
    ↪  player.vy = 0 end }
  end
end

#taken from Captain Ruby code
def try_to_jump(player)
  if solid?(player.game_map, player.x, player.y + 1)
    player.vy = -18
  end
end


def Enemy_Data(image, x, y)
```

```
203   enemy = Enemy.new()
204   enemy.image = image
205   enemy.x = x
206   enemy.y = y
207   enemy
208 end
209 
210 def enemy_draw(enemy)
211   enemy.image.draw_rot(enemy.x, enemy.y, 0, 0)
212 end
213 
214 #set up the enemy ai in here. It is a decision tree and probabilistic algorithm
    ↪  merged together
215 def enemy_move(player, enemy)
216   enemy.each do |c|
217         if (c.x < player.x)
218             c.x += rand(0.0..3.0)
219         elsif(c.x > player.x)
220             c.x -= rand(0.0..3.0)
221         else
222             c.x = 0.0
223         end
224 
225 
226         if (c.y < player.y)
227             c.y += rand(0.0..3.0)
228         elsif(c.y > player.y)
229             c.y -= rand(0.0..3.0)
230         else
231             c.y = 0.0
232         end
233 
234   end
235 
236 end
237 
238 #This is an example of FSM type AI
239 def enemy_touch(player, enemy)#reduced enemy hitbox to make it easier for player to
    ↪  dodge it
240   enemy.each do |c|
241       if ((c.x - player.x).abs < 25 and (c.y - player.y).abs < 25)
242 
243           @trigger = false
244           @trigger_Enemy_Death = true
245       end
246   end
247 end
248 
249 
250 
251 def Spike_Data(image, x, y)
252   spike = Spike.new()
253   spike.image = image
```

```
254   spike.x = x
255   spike.y = y
256   spike
257 end
258
259 def spike_draw(spike)
260   spike.image.draw_rot(spike.x, spike.y, 0, 0)
261 end
262
263
264 #This is an example of FSM type AI, even though object is stationary here!
265 def spike_touch(player, spike)#reduced spike hitbox to make it more playable
266   spike.each do |c|
267       if ((c.x - player.x).abs < 40 and (c.y - player.y).abs < 40)
268
269           @trigger = false
270           @trigger_Spike_Death = true
271       end
272   end
273 end
274
275
276 def Orb_Data(image, x, y)
277   orb = Orb.new()
278   orb.image = image
279   orb.x = x
280   orb.y = y
281   orb
282 end
283
284 def orb_spin(orb)#give different effect by playing around
285   # Draw, slowly rotating
286   #math.cos mainly gives multiple values making it rotate to different degrees
287   orb.image.draw_rot(orb.x, orb.y, 0, 25 * Math.cos(Gosu.milliseconds / 133.7))
288 end
289
290 def orb_gone(player, orb)
291   # Same as in the tutorial game.
292   #checksif object x and y values match with character or not. if matches then
      ↪  removes object from array "orb"
293   orb.reject! do |c|
294     (c.x - player.x).abs < 50 and (c.y - player.y).abs < 50
295
296   end #end for do condition
297
298   if (orb[0] == nil)
299       @trigger = false
300       @trigger_endscreen = true
301
302   #these two lines are for testing stuff:
303   # puts "testing nothing remains"
304   # puts "#{@time_now}"
305
```

```ruby
  end

end


#draws all start screen elements
def draw_start_screen_elements(z_order)
    start_button_elements(z_order)
    levels(z_order)
    quit_button_elements(z_order)

    #loads up all the levels when clicked on "level"
    if (@trigger_level==true)
        level_1_button_elements(z_order)
        level_2_button_elements(z_order)
        level_3_button_elements(z_order)
        level_4_button_elements(z_order)
        level_impossible_button_elements(z_order)
    end
end

def quit_button_elements(z_order)
    # Draw the button
    Gosu.draw_rect(50, 450, 80, 30, Gosu::Color::BLACK, z_order, mode=:default)

    #hover technique from hover button
    if  mouse_over_button_quit(mouse_x, mouse_y)
        Gosu.draw_rect(45, 445, 90, 40, Gosu::Color::WHITE, z_order - 1,
        ↪  mode=:default)
    end

    # Draw the button text
    @button_start_font.draw("QUIT", 60, 460, z_order, 1.0, 1.0, Gosu::Color::WHITE)
end

def start_button_elements(z_order)
    # Draw the button
    Gosu.draw_rect(50, 250, 80, 30, Gosu::Color::BLACK, z_order, mode=:default)

    #hover technique from hover button
    if  mouse_over_button_start(mouse_x, mouse_y)
        Gosu.draw_rect(45, 245, 90, 40, Gosu::Color::WHITE, z_order - 1,
        ↪  mode=:default)
    end

    # Draw the button text
    @button_start_font.draw("DEMO", 60, 260, z_order, 1.0, 1.0, Gosu::Color::WHITE)
end

def levels(z_order)
    # Draw the button
    Gosu.draw_rect(50, 305, 90, 30, Gosu::Color::BLACK, z_order, mode=:default)

```

```ruby
    #hover technique from hover button
    if  mouse_over_button_level(mouse_x, mouse_y)
        Gosu.draw_rect(45, 300, 100, 40, Gosu::Color::WHITE, z_order-1,
        ↪  mode=:default)
    end

    # Draw the button text
    @button_start_font.draw("LEVELS:", 60, 310, z_order, 1.0, 1.0,
    ↪  Gosu::Color::WHITE)
end


def level_1_button_elements(z_order)
    # Draw the button
    Gosu.draw_rect(50, 350, 80, 30, Gosu::Color::BLACK, z_order, mode=:default)

    #hover technique from hover button
    if  mouse_over_button_level_1(mouse_x, mouse_y)
        Gosu.draw_rect(45, 345, 90, 40, Gosu::Color::WHITE, z_order-1,
        ↪  mode=:default)
    end

    # Draw the button text
    @button_start_font.draw("Level 1", 60, 360, z_order, 1.0, 1.0,
    ↪  Gosu::Color::WHITE)
end

def level_2_button_elements(z_order)
    # Draw the button
    Gosu.draw_rect(160, 350, 80, 30, Gosu::Color::BLACK, z_order, mode=:default)

    #hover technique from hover button
    if  mouse_over_button_level_2(mouse_x, mouse_y)
        Gosu.draw_rect(155, 345, 90, 40, Gosu::Color::WHITE, z_order-1,
        ↪  mode=:default)
    end

    # Draw the button text
    @button_start_font.draw("Level 2", 170, 360, z_order, 1.0, 1.0,
    ↪  Gosu::Color::WHITE)
end

def level_3_button_elements(z_order)
    # Draw the button
    Gosu.draw_rect(270, 350, 80, 30, Gosu::Color::BLACK, z_order, mode=:default)

    #hover technique from hover button
    if  mouse_over_button_level_3(mouse_x, mouse_y)
        Gosu.draw_rect(265, 345, 90, 40, Gosu::Color::WHITE, z_order-1,
        ↪  mode=:default)
    end

    # Draw the button text
```

```ruby
    @button_start_font.draw("Level 3", 280, 360, z_order, 1.0, 1.0, Gosu::Color::WHITE)
end

def level_4_button_elements(z_order)
    # Draw the button
    Gosu.draw_rect(380, 350, 80, 30, Gosu::Color::BLACK, z_order, mode=:default)

    #hover technique from hover button
    if  mouse_over_button_level_4(mouse_x, mouse_y)
        Gosu.draw_rect(375, 345, 90, 40, Gosu::Color::WHITE, z_order-1, mode=:default)
    end

    # Draw the button text
    @button_start_font.draw("Level 4", 390, 360, z_order, 1.0, 1.0, Gosu::Color::WHITE)
end

def level_impossible_button_elements(z_order)
    # Draw the button
    Gosu.draw_rect(160, 400, 170, 30, Gosu::Color::BLACK, z_order, mode=:default)

    #hover technique from hover button
    if  mouse_over_button_level_impossible(mouse_x, mouse_y)
        Gosu.draw_rect(150, 395, 190, 40, Gosu::Color::WHITE, z_order-1, mode=:default)
    end

    # Draw the button text
    @button_start_font.draw("IMPOSSIBLE LEVEL", 160, 410, z_order, 1.0, 1.0, Gosu::Color::WHITE)
end

#these load the level timer and orb counter elements
def    draw_level_extra_elements(z_order)
    draw_level_timer(z_order)
    draw_orb_counter(z_order)
end

def draw_level_timer(z_order)
    @sky.draw 0, 0, 0
    Gosu.draw_rect(0, 0, 175, 20, Gosu::Color::BLACK, z_order-1, mode=:default)
    @end_font.draw("Timer: #{@time_now.round(1)} seconds", 5, 0, z_order, 1.0, 1.0, Gosu::Color::WHITE)
end

def draw_orb_counter(z_order)
    Gosu.draw_rect(195, 0, 105, 20, Gosu::Color::BLACK, z_order -1, mode=:default)
    @end_font.draw("Orbs left: #{@game_map.orb.length}", 200, 0, z_order, 1.0, 1.0, Gosu::Color::WHITE)
end

```

```ruby
#makes sure it draws these set of objects for victory screen
def draw_victory_end(z_order)
    @end_font.draw("GAME OVER! ", 60, 30, z_order, 1.0, 1.0, Gosu::Color::WHITE)
    @end_font.draw("Time Taken: #{@time_now.round(1)} seconds!", 60, 50, z_order, 1.0, 1.0, Gosu::Color::WHITE)
    return_button_elements(z_order)

    #gives out certain comments to player's gameplay depending on time taken by him or her
    if(@time_now.round(1) < 20)
        @end_font.draw("You know the map too well. Are you cheating? ", 60, 80, z_order, 1.0, 1.0, Gosu::Color::WHITE)
    end
    if(@time_now.round(1) > 20 && @time_now.round(1) < 60)
        @end_font.draw("Excellent! ", 60, 80, z_order, 1.0, 1.0, Gosu::Color::WHITE)
    end
    if(@time_now.round(1) > 60 && @time_now.round(1) < 80)
        @end_font.draw("Good Effort! But I know you can do better!", 60, 80, z_order, 1.0, 1.0, Gosu::Color::WHITE)
    end
    if(@time_now.round(1) > 80 && @time_now.round(1) < 100)
        @end_font.draw("You are very slow", 60, 80, z_order, 1.0, 1.0, Gosu::Color::WHITE)
    end
    if(@time_now.round(1) > 100)
        @end_font.draw("Are you sure you are playing the game?", 60, 80, z_order, 1.0, 1.0, Gosu::Color::WHITE)
    end

    @character_win.draw(0, 0, z_order)

end

#makes sure it draws these set of objects for killed by spike screen
def draw_defeat_spike_end(z_order)
    @end_font.draw("GAME OVER! ", 60, 60, z_order, 1.0, 1.0, Gosu::Color::WHITE)
    @end_font.draw("You have been killed by spikes!!", 60, 80, z_order, 1.0, 1.0, Gosu::Color::WHITE)
    return_button_elements(z_order)
    @character_dead.draw(0, 0, z_order)
end

#makes sure it draws these set of objects for killed by enemy (MIMIC) screen
def draw_defeat_enemy_end(z_order)
    @end_font.draw("GAME OVER! ", 60, 60, z_order, 1.0, 1.0, Gosu::Color::WHITE)
    @end_font.draw("You have been killed by MIMIC!!", 60, 80, z_order, 1.0, 1.0, Gosu::Color::WHITE)
    return_button_elements(z_order)
    @character_dead.draw(0, 0, z_order)
end

#draws return button elements on all end screens
def return_button_elements(z_order_level)
```

```ruby
    # Draw the button
    Gosu.draw_rect(55, 110, 140, 30, Gosu::Color::BLACK, z_order_level, mode=:default)

    #hover technique from hover button
    if  mouse_over_button_end(mouse_x, mouse_y)
        Gosu.draw_rect(50, 105, 150, 40, Gosu::Color::WHITE, z_order_level-1, mode=:default)
    end

    # Draw the button text
    @button_start_font.draw("Return Home", 60, 110, z_order_level, 1.0, 1.0, Gosu::Color::WHITE)
end




class CollectOrbGame < (Example rescue Gosu::Window)
  def initialize
  #all triggers and objects are initialized here and each new space means a different object from a different group. I didnt put them in different functions here as i want to keep all triggers and initialize all objetcs in the same place.
    super WIDTH, HEIGHT

    self.caption = "Collect all orbs!"

    #all start screen object and z orders
    @start_screen = Gosu::Image.new("image/start1.png")
    @button_start_font = Gosu::Font.new(20)
    @start_screen_z_order = 1
    @start_button_z_order = 3

    @trigger_map = true
    #for background picture during level play through
    @sky = Gosu::Image.new("image/background2.jpg")

    if (@trigger_map == true)
    @game_map = setup_game_map("image/gamemapdemo.txt")    #kept it so that things all load smth at first. This will get overridden when other levels load

    #puts "demo map start" #testing level loads or not
    end


    @bot = setup_player(@bot, @game_map, 2500, 950)
    #kept it so that player things all load smth at first, this will get overridden when other levels load

    #all level triggers
    @trigger_level_1 = false
    @trigger_level_2 = false
```

```ruby
    @trigger_level_3 = false
    @trigger_level_4 = false
    @trigger_level_impossible = false
    @trigger_level_demo = false

    @camera_x = @camera_y = 0

    #for the z level of timer and orb remaining number
    @counter_display_information_z_order = -1


    @trigger_level=false #for "click level at start to show all level options"
    @trigger = false    #all @trigger is for timer and also enemy movement
     ↪  generation

    #triggers for all endscrren variations and elements
    @trigger_endscreen = false
    @trigger_Spike_Death = false
    @trigger_Enemy_Death = false
    @trigger_return_home = false

    #z_order level for all end screen elements
    @end_z_order = -1
    @end_z_order_victory = -1
    @end_z_order_spike_death = -1
    @end_z_order_enemy_death = -1

    #end screen font size
    @end_font = Gosu::Font.new(20)

    #endscreen images
    @character_dead = Gosu::Image.new("image/character_death3.png")
    @character_win = Gosu::Image.new("image/character_win.png")

    #time counter initialized. I used this as I far more custom control over it.
    @time_now = 0

    #all song elements used in his game
    @song = Gosu::Song.new("image/game_background_music2.wav")
    @start_song = false    #says whether song is not played for not

  end

 def update
    if(@trigger_level_demo ==true)
        @game_map = setup_game_map("image/gamemapdemo.txt")
        @bot = setup_player(@bot, @game_map, 2500, 950)
        @trigger_map = false

        @trigger_return_home = false
        @trigger_level_demo =false
        #puts "demo map select"    #testing level loads or not
    end
```

```
591
592     if(@trigger_level_1 ==true)
593         @game_map = setup_game_map("image/gamemap1.txt")
594         @bot = setup_player(@bot, @game_map, 200, 1100)
595         @trigger_map = false
596
597         @trigger_return_home = false
598         #puts "lvl1 map select"    #testing level loads or not
599         @trigger_level_1 =false
600     end
601
602     if(@trigger_level_2 ==true)
603         @game_map = setup_game_map("image/gamemap2.txt")
604         @bot = setup_player(@bot, @game_map, 200, 850)
605         @trigger_map = false
606
607         @trigger_return_home = false
608         #puts "lvl2 map select"    #testing level loads or not
609         @trigger_level_2 =false
610     end
611
612     if(@trigger_level_3 ==true)
613         @game_map = setup_game_map("image/gamemap3.txt")
614         @bot = setup_player(@bot, @game_map, 1500, 950)
615         @trigger_map = false
616
617         @trigger_return_home = false
618         #puts "lvl3 map select"    #testing level loads or not
619         @trigger_level_3 =false
620     end
621
622     if(@trigger_level_4 ==true)
623         @game_map = setup_game_map("image/gamemap4.txt")
624         @bot = setup_player(@bot, @game_map, 2500, 950)
625         @trigger_map = false
626
627         @trigger_return_home = false
628         #puts "lvl4 map select"    #testing level loads or not
629         @trigger_level_4 =false
630     end
631
632     if(@trigger_level_impossible ==true)
633         @game_map = setup_game_map("image/gamemapimpossible.txt")
634         @bot = setup_player(@bot, @game_map, 2500, 950)
635         @trigger_map = false
636
637         @trigger_return_home = false
638         #puts "lvl5 map select"    #testing level loads or not
639         @trigger_level_impossible =false
640     end
641     #player movement data. I made it so that player can move faster then enemy
      ↪  movement in order to give the player a better chance of surviving
642     move_x = 0
```

```
643     move_x -= 5 if Gosu.button_down? Gosu::KB_LEFT
644     move_x += 5 if Gosu.button_down? Gosu::KB_RIGHT
645     update_player(@bot, move_x)
646 
647     #objects and player interaction
648     orb_gone(@bot, @game_map.orb)#reject orb inside this
649     spike_touch(@bot, @game_map.spike)#spike kill condition inside this
650     enemy_touch(@bot, @game_map.enemy)#enemy kill condition inside this
651 
652 
653 
654     # Scrolling follows player
655     #needed for camera translation in draw part
656     @camera_x = [[@bot.x - WIDTH / 2, 0].max, @game_map.width * 50 - WIDTH].min
657     @camera_y = [[@bot.y - HEIGHT / 2, 0].max, @game_map.height * 50 - HEIGHT].min
658 
659 
660     if (@trigger == true)#for all levels
661         #move enemy that spawned with map
662         enemy_move(@bot, @game_map.enemy)
663 
664         @time_now += 0.0175 #(will now give second's value)
665 
666         #these will remove all start menu objects
667         @start_button_z_order = -1
668         @start_screen_z_order = -1
669 
670         #these will load all game counter objects
671         @counter_display_information_z_order = 2
672 
673         #this will trigger the play song
674         @start_song = true
675     end
676 
677     #plays song in a loop
678     if(@start_song == true)
679         @song.play(looping = true)
680     end
681 
682     if(@trigger_endscreen == true)
683         #these will load all victory screen objects
684         @end_z_order = 3
685         @end_z_order_victory = 4
686 
687         @trigger =false
688 
689         #stops song
690         @song.stop
691     end
692 
693     if(@trigger_Spike_Death == true)
694         #these will load all spike death screen objects
695         @end_z_order = 3
```

```
696         @end_z_order_spike_death = 4
697
698         @trigger =false
699
700         #stops song
701         @song.stop
702     end
703
704     if(@trigger_Enemy_Death == true)
705         #these will load all spike death screen objects
706         @end_z_order = 3
707         @end_z_order_enemy_death = 4
708
709         @trigger =false
710
711         #stops song
712         @song.stop
713     end
714
715     if (@trigger_return_home==true)
716         #makes everything else false, and z_order -1
717         #makes z_order to start screen normal now
718         @trigger_endscreen = false
719         @trigger_Spike_Death = false
720         @trigger_Enemy_Death = false
721
722         #loads up all start screen objects
723         @start_screen_z_order = 1
724         @start_button_z_order = 3
725
726         #remove unnecesary objects
727         @counter_display_information_z_order = -1
728
729         @end_z_order = -1
730         @end_z_order_victory = -1
731         @end_z_order_spike_death = -1
732         @end_z_order_enemy_death = -1
733
734         #stops song
735         @song.stop
736
737         #resets time counter
738         @time_now = 0
739     end
740
741
742   end
743
744
745
746
747   def draw
748     @start_screen.draw 0, 0, @start_screen_z_order
```

```
749
750     draw_start_screen_elements(@start_button_z_order)
751     draw_level_extra_elements(@counter_display_information_z_order)
752
753     #this part prints camera position, for testing stuff
754     # puts "Camera X is #{@camera_x}"
755     # puts "Camera Y is #{@camera_y}"
756
757     Gosu.draw_rect(0, 0, WIDTH, HEIGHT, Gosu::Color::BLACK, @end_z_order,
        ↪  mode=:default)
758
759     draw_victory_end(@end_z_order_victory)    #called if all orb collected
760     draw_defeat_spike_end(@end_z_order_spike_death)    #called if death to spike
761     draw_defeat_enemy_end(@end_z_order_enemy_death)    #called if killed by
        ↪  enemy(ie the mimic)
762
763
764     #this is for testing stuff to see the timer and to help finetune it
765     #puts "time from start is #{@time_now.round(2)} seconds"
766
767
768     Gosu.translate(-@camera_x, -@camera_y) do
769       draw_game_map(@game_map)
770       draw_player(@bot)
771     end
772   end
773
774     #all button clicking locations
775     def mouse_over_button_start(mouse_x, mouse_y)
776         if ((mouse_x >= 40 && mouse_x <= 135) && (mouse_y >= 245 && mouse_y <= 285))
777           true
778
779         else
780           false
781         end
782     end
783
784     def mouse_over_button_level_1(mouse_x, mouse_y)
785         if ((mouse_x >= 45 && mouse_x <= 135) && (mouse_y >= 345 && mouse_y <= 385))
786           true
787
788         else
789           false
790         end
791     end
792
793     def mouse_over_button_level_2(mouse_x, mouse_y)
794         if ((mouse_x >= 155 && mouse_x <= 245) && (mouse_y >= 345 && mouse_y <=
            ↪  385))
795           true
796
797         else
798           false
```

```
799         end
800     end
801
802     def mouse_over_button_level_3(mouse_x, mouse_y)
803         if ((mouse_x >= 265 && mouse_x <= 355) && (mouse_y >= 345 && mouse_y <=
        ↪  385))
804           true
805
806         else
807           false
808         end
809     end
810
811     def mouse_over_button_level_4(mouse_x, mouse_y)
812         if ((mouse_x >= 375 && mouse_x <= 465) && (mouse_y >= 345 && mouse_y <=
        ↪  385))
813           true
814
815         else
816           false
817         end
818     end
819
820     def mouse_over_button_level_impossible(mouse_x, mouse_y)
821         if ((mouse_x >= 150 && mouse_x <= 340) && (mouse_y >= 395 && mouse_y <=
        ↪  435))
822           true
823
824         else
825           false
826         end
827     end
828
829     def mouse_over_button_level(mouse_x, mouse_y)
830         if ((mouse_x >= 45 && mouse_x <= 145) && (mouse_y >= 300 && mouse_y <= 340))
831           true
832
833         else
834           false
835         end
836     end
837
838     def mouse_over_button_end(mouse_x, mouse_y)
839         if ((mouse_x >= 50 && mouse_x <= 200) && (mouse_y >= 105 && mouse_y <= 145))
840           true
841
842         else
843           false
844         end
845     end
846
847     def mouse_over_button_quit(mouse_x, mouse_y)
848         if ((mouse_x >= 45 && mouse_x <= 135) && (mouse_y >= 445 && mouse_y <= 485))
```

```
849           true
850
851         else
852           false
853         end
854       end
855
856   #gave z order as another restriction along side all button location to make sure
      ↪  only the button loaded with a specific z order will be selected and not the one
      ↪  behind it
857     def button_down(id)
858     case id
859     when Gosu::MsLeft
860         if mouse_over_button_start(mouse_x, mouse_y) && @start_button_z_order == 3
861           @trigger_level_demo = true
862           @trigger = true
863           @trigger_level = false
864         end
865         if mouse_over_button_level_1(mouse_x, mouse_y) && @start_button_z_order == 3
866           @trigger_level_1 = true
867           @trigger = true
868           @trigger_level = false
869         end
870          if mouse_over_button_level_2(mouse_x, mouse_y) && @start_button_z_order == 3
871           @trigger_level_2 = true
872           @trigger = true
873           @trigger_level = false
874         end
875         if mouse_over_button_level_3(mouse_x, mouse_y) && @start_button_z_order == 3
876           @trigger_level_3 = true
877           @trigger = true
878           @trigger_level = false
879         end
880         if mouse_over_button_level_4(mouse_x, mouse_y) && @start_button_z_order == 3
881           @trigger_level_4 = true
882           @trigger = true
883           @trigger_level = false
884         end
885         if mouse_over_button_level_impossible(mouse_x, mouse_y) &&
          ↪  @start_button_z_order == 3
886           @trigger_level_impossible = true
887           @trigger = true
888           @trigger_level = false
889         end
890         if mouse_over_button_level(mouse_x, mouse_y) && @start_button_z_order == 3
891           @trigger_level = true
892
893         end
894
895         if mouse_over_button_end(mouse_x, mouse_y) &&(@end_z_order_spike_death ==4 ||
          ↪  @end_z_order_enemy_death==4 || @end_z_order_victory==4)
896           @trigger_return_home = true
897           @trigger = false
```

```
898       end
899       if   mouse_over_button_quit(mouse_x, mouse_y) && @start_button_z_order == 3
900         close
901       end
902
903     when Gosu::KB_UP
904       try_to_jump(@bot)
905     when Gosu::KB_ESCAPE
906         close
907     else
908       super
909     end
910   end
911
912 end
913
914 CollectOrbGame.new.show if __FILE__ == $0
```


C:\Users\SAIFUL>C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\6.3D-custom program\game things\my game\game stuff\orb_collector_final
'C:\Users\SAIFUL\Desktop\swinburne' is not recognized as an internal or external command,
operable program or batch file.
C:\Users\SAIFUL>cd C:\Users\SAIFUL\Desktop\swinburne
al
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS1000
Volume in drive C has no label.
Volume Serial Number is BED8-11C3
Directory of C:\Users\SAIFUL\Desktop\swinburne class
05/31/2021 10:27 AM <DIR> .
05/31/2021 10:27 AM <DIR> ..
05/31/2021 10:26 AM <DIR> image
05/31/2021 10:19 AM 25,309 orb_collector.
1 File(s) 25,309 bytes
3 Dir(s) 8,553,603,072 bytes free
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS1000
DEPRECATION WARNING: draw is deprecated; use Font#dra
draw called from C:/Users/SAIFUL/Desktop/swinburne cl
/orb_collector.rb:351:in `start_button_elements'.
Collect all orbs!
Timer: 10.3 seconds
Orbs left: 7

31 Recursive Factorial

A simple recursive program to calculate Factorials.

Outcome	Weight
Functional Decomposition	♦♦♦◊◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦♦♦◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Recursive Factorial

Submitted By:
S M Ragib Rezwan
103172423
2021/05/09 20:43

Tutor:
NAJAM NAZAR

May 9, 2021

```ruby
# Recursive Factorial

# Complete the following
def factorial(n)
    if (n<=1)
    return n
    else
    n = n * factorial(n-1)
    end
end
def read_integer prompt
    value = read_string(prompt)
    return value
end

def read_string prompt
    puts prompt
    value = gets.chomp
end


# returns true if a string contains only digits
def is_numeric?(obj)
  if /^[0-9]+$/.match(obj.to_s) == nil

  #we know match cant be nil if it is number. so if it comes nil then it is false!
    false

  else
  true

  end
end


def main
    ARGV[0]=read_integer("")
    if (is_numeric?(ARGV[0]))
    ARGV[0] = ARGV[0].to_i
        if(ARGV[0] < 0)
        #if negative number inputted
        puts "Incorrect arguement -need a single arguement with ruby value of 0 or more"

        else
        #if number is 0 or more, run factorial
         puts factorial(ARGV[0].to_i)
        end
  else
    #if letter inputted
    puts "Incorrect arguement -need a single arguement with ruby value of 0 or more"
  end

```

```
53  end
54
55 main
```

```
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week10\10.2C-resources\Resources>recursive_factorial_incomplete.rb


Incorrect arguement -need a single arguement with ruby value of 0 or more

C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week10\10.2C-resources\Resources>recursive_factorial_incomplete.rb

-1
Incorrect arguement -need a single arguement with ruby value of 0 or more

C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week10\10.2C-resources\Resources>recursive_factorial_incomplete.rb

4
24

C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week10\10.2C-resources\Resources>
```

32 C Hello World

Write and compile a simple C program.

Outcome	Weight
Functional Decomposition	♦♦◊◊◊

Outcome	Weight
Structured Coding Principles	♦◊◊◊◊

Outcome	Weight
Programming	♦◊◊◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

C Hello World

Submitted By:
S M Ragib Rezwan
103172423
2021/05/09 20:09

Tutor:
NAJAM NAZAR

May 9, 2021

```c
#include <stdio.h>

void print_message(char *message)
{
    printf("%s", message);
}

int main()
{
  print_message("Hello World!\n");

  return 0;
}
```


Directory of D:\mingw-w64\i686-8.1.0-posix-dwarf-rt_v6-rev0
05/09/2021 03:52 PM <DIR> .
05/09/2021 03:52 PM <DIR> ..
01/17/2019 03:30 PM 6,148 .DS_Store
05/09/2021 03:52 PM 48,467 binary_program.exe
01/17/2019 03:11 PM 160 hello_world.c
02/28/2021 10:19 AM 253 mingw-w64.bat
12/28/2015 01:00 AM 67 mingw-w64.url
03/06/2021 08:19 AM <DIR> mingw32
02/28/2021 10:19 AM 958,464 uninstall.exe
02/28/2021 10:02 AM 1,525,085 uninstall.ini
7 File(s) 2,538,644 bytes
3 Dir(s) 48,175,984,640 bytes free
D:\mingw-w64\i686-8.1.0-posix-dwarf-rt_v6-rev0>binary_program.exe
Hello World!

33 Text Music Player with Menu

Text Music Player

Outcome	Weight
Functional Decomposition	♦♦♦♦♦

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦♦◊◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Text Music Player with Menu

Submitted By:
S M Ragib Rezwan
103172423
2021/04/22 08:03

Tutor:
NAJAM NAZAR

April 22, 2021

```ruby
require './input_functions'


module Genre
  POP, CLASSIC, JAZZ, ROCK = *1..4
end

$genre_names = ['Null', 'Pop', 'Classic', 'Jazz', 'Rock']

#adding objects to class and ensuring they can be called anywhere, in any function.
class Album

    attr_accessor :title, :artist, :genre, :tracks


    def initialize (title, artist, genre, tracks)

        @title = title

        @artist = artist
        @genre = genre
        @tracks = tracks

    end
end

#adding objects to class and ensuring they can be called anywhere, in any function.
class Track

    attr_accessor :name, :location

    def initialize (name, location)
        @name = name
        @location = location
    end
end

# Reads in and returns a single track from the given file (called in
 ↪  read_tracks(music_file))
def read_track(music_file)

    name = music_file.gets()
    location = music_file.gets()
    track = Track.new(name, location)

    return track
end

# Returns an array of tracks read from the given file (called in
 ↪  read_album(music_file))
def read_tracks(music_file)
    tracks = Array.new()
    count = music_file.gets().to_i()
```

```
52
53      i=0
54      while i < count do
55        track = read_track(music_file)
56
57        tracks << track
58          i = i+1
59      end
60
61    return tracks
62  end
63
64  # Takes an array of tracks and prints them to the terminal (called in
    ↪  print_album(album))
65  def print_tracks(tracks)
66
67      i=0
68      while (i < tracks.length) do
69          print_track(tracks[i])
70          i = i +1
71      end
72  end
73
74  # Reads in and returns a single album from the given file, with all its tracks
    ↪  (called in read_albums(music_file))
75
76  def read_album(music_file)
77        album_artist = music_file.gets()
78      album_title = music_file.gets()
79      album_genre = music_file.gets()
80      tracks = read_tracks (music_file)
81
82      album = Album.new(album_title, album_artist, album_genre, tracks)
83      return album
84
85  end
86  #stores all the data into albums (called in load_read_album_close_file(file_name))
87  def read_albums(music_file)
88
89    albums = Array.new
90    count = music_file.gets().to_i
91    i=0
92    while (i < count)
93        album = read_album(music_file)
94
95        albums << album
96        i = i + 1
97    end
98    return albums
99
100 end
101
102 #prints all albums with album id added to each to make easy reference(called in
    ↪  display_album(file_name))
```

```
103 def print_albums(albums)
104     i=0
105     while (i < albums.length) do
106         print "Album ID:" + i.to_s + "\n"
107         print_album(albums[i])
108         i = i +1
109     end
110 
111 end
112 
113 # Takes a single album and prints it to the terminal along with all its fields and
    ↪ attributes
114 #(called in print_albums(albums),print_specific_genre_album_details(albums,
    ↪ found_index))
115 def print_album(album)
116 
117     puts (album.artist.to_s())
118     puts (album.title.to_s())
119     puts('Genre is ' + album.genre.to_s())
120     puts($genre_names[album.genre.to_i])
121     print_tracks(album.tracks)
122     #Note:  (album.tracks)is must! otherwise it wont realise that we are
123     #        printing tracks and instead will make a new null array!
124 
125 
126     # print out the tracks
127 
128 end
129 
130 # Takes a single track and prints it to the terminal(called in
    ↪ print_tracks(tracks), print_tracks2(tracks))
131 def print_track(track)
132     puts(track.name)
133     puts(track.location)
134 end
135 
136 
137 
138 #reads in name of file containing albums (called in main)
139 def read_in_album()
140 
141     puts 'Read Album Submenu'
142    a = read_string( "Name of file:")
143    return a
144 
145 end
146 
147 
148 
149 
150 #searches album via genre (called in display_genre_specific_album(albums))
151 def search_for_genre(albums, search_string)
152     index = 0
```

```
153     found_index = -1
154     while (index < albums.length )
155
156
157         if(albums[index].genre.chomp == search_string)
158             found_index = index
159             print_specific_genre_album_details(albums, found_index)
160         end
161         index = index + 1
162     end
163
164 return found_index
165 end
166
167 #prints out all albums of that genre alongside their own unique album ids (called
    ↪  in search_for_genre (albums, search_string))
168 def print_specific_genre_album_details(albums, found_index)
169     print "Album ID:" + found_index.to_s
170     print "\n"
171     print_album(albums[found_index])
172 end
173
174 #reads in genre and calls the search function for genre in here(called in
    ↪  display_album(file_name))
175 def display_genre_specific_album(albums)
176     search_string = read_integer_in_range("Enter the Genre (number): ",1,4).to_s
177
178       index = search_for_genre(albums, search_string)
179
180     if index == -1
181           puts "Entry not Found"
182       end
183 end
184
185 #displays submenus and also calls functions to display all albums and specific
    ↪  genre albums (called in main)
186 def display_album(file_name)
187     albums = load_read_album_close_file(file_name)
188
189     puts 'Display Album Submenu'
190     puts '1 Display all albums'
191     puts '2 Display Albums of specific genre'
192
193     choice = read_integer_in_range("Please enter your choice:", 1, 2)
194     case choice
195     when 1
196       print_albums(albums)
197     when 2
198       display_genre_specific_album(albums)
199     else
200       puts 'Please select again'
201     end
202 end
```

```ruby

#selects certain album to be played using its unique id (called in main )
def select_album_to_play(file_name)
    albums = load_read_album_close_file(file_name)

    search_string = read_integer("Enter the Album number: ")
      index = search_for_album(albums, search_string)
      if index > -1
           print_tracks2(albums[index].tracks)
        print_specific_track(albums, index)
    else
        puts "entry not found"
    end

    #need to insert a delay here of a few seconds, so used sleep function (took idea from the gosu thing we did in tick timer one)
    sleep 5

end
#prints specific track of that album, including its spefic track id(called in select_album_to_play(file_name))
def print_specific_track(albums, index)
    search_string = read_integer("Enter the Track number: ")
    index2 = search_and_play_track(albums[index].tracks, search_string)
        if index2 > -1
            print ("Playing track #{albums[index].tracks[index2].name.to_s} from album  #{albums[index].title.to_s}")
        else
            print "track not found"
        end
end

#searches a track using certain id(called in print_specific_track(albums, index))
def search_and_play_track(tracks, search_string)
    index = 0
    found_index = -1
    while (index < tracks.length )


        if(index == search_string)
            found_index = index

        end
        index = index + 1
    end

return found_index
end
#searches a specific album using id(called in select_album_to_play(file_name) and update_existing_album(file_name))
def search_for_album(albums, search_string)
    index = 0
    found_index = -1
```

```
252     while (index < albums.length )
253
254
255         if(index == search_string)
256             found_index = index
257
258         end
259         index = index + 1
260     end
261
262 return found_index
263 end
264
265 #print tracks with track number or track id (called in
    ↪ select_album_to_play(file_name))
266 def print_tracks2(tracks)
267
268     i=0
269
270     while (i < tracks.length) do
271         print "Track Number:" + i.to_s + "\n"
272         print_track(tracks[i])
273         i = i +1
274     end
275 end
276
277
278 #Only updates data of file in program.
279 #Beware! It doesn't write the new update into the music file main file!
280
281
282
283
284 #updates existing album's either title or genre (called in main)
285 def update_existing_album(file_name)
286
287     albums = load_read_album_close_file(file_name)
288
289     search_string = read_integer("Enter the Album number: ")
290       index = search_for_album(albums, search_string)
291     while(index < albums.length) do
292         if (index > -1)
293
294             puts '1 Change title?'
295             puts '2 Chenge Genre?'
296
297             choice = read_integer_in_range("Please enter your choice:", 1, 2)
298
299             case choice
300             when 1
301                 return give_me_album_title(index, albums)
302
303             when 2
```

```
                return give_me_album_genre(index, albums)

            else
              puts 'Please select again'
            end


        else
        puts "invalid album"
        end
    end
end

#changes album genre (called in update_existing_album(file_name))
def give_me_album_genre(index, albums)
    albums[index].genre = read_integer_in_range("Enter the Genre (number):
    ↪  ",1,4).to_s
    print_album (albums[index])
    a = read_string("Press enter to return")
    return
end
#changes album title (called in update_existing_album(file_name))
def give_me_album_title(index, albums)
    albums[index].title = read_string("Please enter new title:")
    print_album (albums[index])
    a = read_string("Press enter to return")
    return
end

#opens files as a read only object, runs the read_albums function, closes file
#(used in update_existing_album(file_name), select_album_to_play(file_name),
↪  display_album(file_name))
def load_read_album_close_file(file_name)
    music_file = File.new(file_name, "r")
    albums = read_albums(music_file)
    music_file.close()
    return albums
end




#chooses options from main menu and loads up certain programs when certain numbers
↪  are pressed
def main()
  finished = false
  begin
    puts 'Main Menu:'
    puts '1 Read in Albums'
    puts '2 Display Albums'
    puts '3 Select an Album to play'
    puts '4 Update an existing Album'
    puts '5 Exit the application'
```

```
354
355
356     choice = read_integer_in_range("Please enter your choice:", 1, 5)
357     case choice
358     when 1
359       file_name = read_in_album()
360
361     when 2
362       display_album(file_name)
363     when 3
364       select_album_to_play(file_name)
365     when 4
366         update_existing_album(file_name)
367
368
369     when 5
370       finished = true
371     else
372       puts 'Please select again'
373     end
374   end until finished
375
376 end
377
378 main()
```

```
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 7\7.1P-resources\Resources>music_player_text_based.rb
Main Menu:
1 Read in Albums
2 Display Albums
3 Select an Album to play
4 Update an existing Album
5 Exit the application
Please enter your choice:
1
Read Album Submenu
Name of file:
album.txt
Main Menu:
1 Read in Albums
2 Display Albums
3 Select an Album to play
4 Update an existing Album
5 Exit the application
Please enter your choice:
2
Display Album Submenu
1 Display all albums
2 Display Albums of specific genre
Please enter your choice:
1
Album ID:0
Neil Diamond
Greatest Hits
Genre is 1
Pop
Crackling Rose
sounds/01-Cracklin-rose.wav
Soolaimon
sounds/06-Soolaimon.wav
Sweet Caroline
sounds/20-Sweet_Caroline.wav
```

34 Simple GUI Music Player

A simple GUI Music player.

Outcome	Weight
Functional Decomposition	♦♦♦♦♦

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦♦♦◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Simple GUI Music Player

Submitted By:
S M Ragib Rezwan
103172423
2021/04/28 16:16

Tutor:
NAJAM NAZAR

April 28, 2021

```ruby
require 'rubygems'
require 'gosu'

TOP_COLOR = Gosu::Color.new(0xFF1EB1FA)
BOTTOM_COLOR = Gosu::Color.new(0xFF1D4DB5)

TrackLeftX = 400

module ZOrder
  BACKGROUND2, BACKGROUND, PLAYER, UI = *0..3
end

module Genre
  POP, CLASSIC, JAZZ, ROCK = *1..4
end

GENRE_NAMES = ['Null', 'Pop', 'Classic', 'Jazz', 'Rock']

class ArtWork
    attr_accessor :bmp

    def initialize (file)
        @bmp = Gosu::Image.new(file)
    end
end

# Put your record definitions here

class Album

    attr_accessor :title, :artist, :genre, :tracks, :artwork


    def initialize (title, artist, genre, tracks, artwork)

        @title = title
        @artist = artist
        @genre = genre
        @tracks = tracks
        @artwork = artwork

    end
end

#adding objects to class and ensuring they can be called anywhere, in any function.
class Track

    attr_accessor :name, :location, :id

    def initialize (name, location, id)
        @name = name
        @location = location
        @id = id
```

```
54      end
55  end
56
57  class MusicPlayerMain < Gosu::Window
58
59      def initialize
60          super 600, 800
61          self.caption = "Music Player"
62          @album_image = Gosu::Image.new("neil2.png")
63          @track_font = Gosu::Font.new(20)
64          @font = Gosu::Font.new(20)
65          @No_of_items_in_tracks
66
67          @z_level = ZOrder::BACKGROUND2
68
69          @count = 0
70          @yposition1 = 15
71
72
73          # Reads in an array of albums from a file and then prints all the albums in
            ↪  the
74          # array to the terminal
75          music_file = File.new("album.txt", "r") #currently hardcoding file name
76          @albums = read_albums(music_file)
77          music_file.close()
78
79          @song = Array.new()
80          i=0
81          while(i < @tracks.length) do
82          @song.push << Gosu::Song.new(@album.tracks[i].location.chomp)    #loads all
            ↪  songs and inputs it to @song using push
83          i = i + 1
84          end
85          @index4 = 0
86          @start=false    #says whether song is not played for not
87
88
89          print_albums(@albums)
90
91      end
92
93    # Put in your code here to load albums and tracks
94    def read_albums(music_file)
95
96    @albums = Array.new
97    count = music_file.gets().to_i
98    i=0
99    while (i < count)
100         @album = read_album(music_file)
101
102         @albums << @album
103         i = i + 1
104   end
```

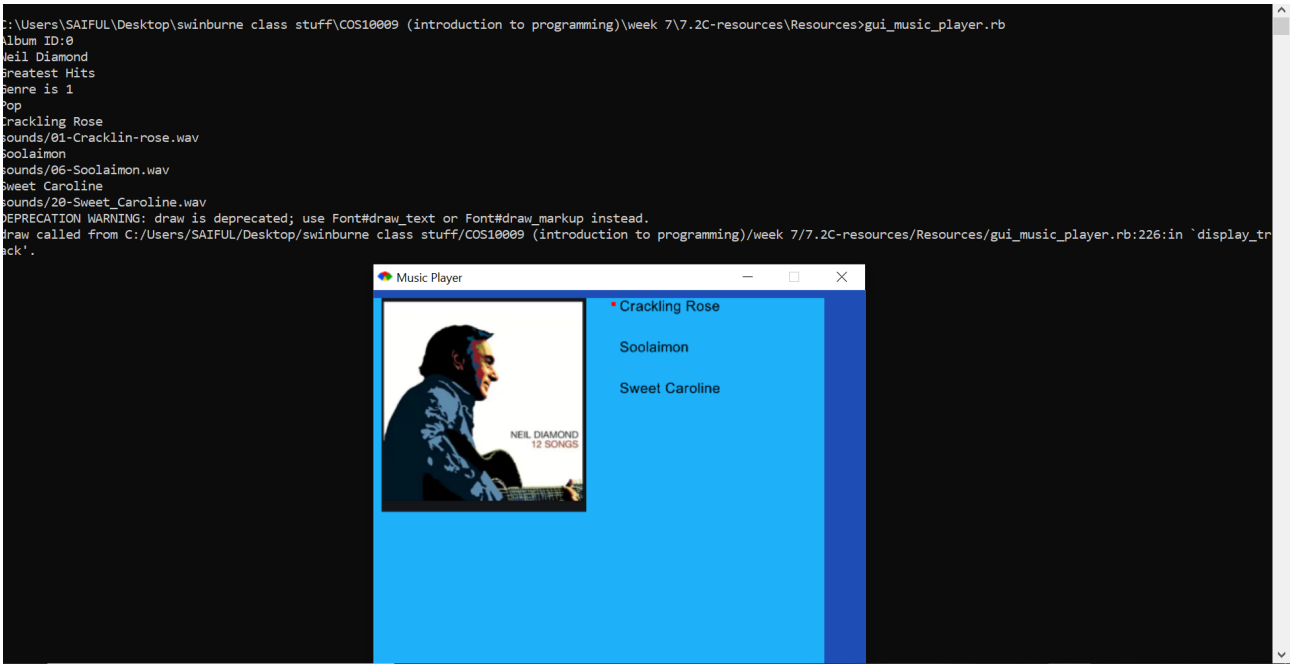
```
105   return @albums
106
107 end
108
109 def read_album(music_file)
110       album_artist = music_file.gets()
111     album_title = music_file.gets()
112     album_genre = music_file.gets()
113     artwork = ArtWork.new(music_file.gets().chomp())
114     @tracks = read_tracks (music_file)
115     @album = Album.new(album_title, album_artist, album_genre, @tracks, artwork)
116     return @album
117
118 end
119
120 def read_tracks(music_file)
121     @tracks = Array.new()
122     count = music_file.gets().to_i()
123
124     i=0
125     while i < count do
126       @track = read_track(music_file, i)
127
128       @tracks << @track
129         i = i+1
130     end
131
132   return @tracks
133 end
134
135 def read_track(music_file, i)
136
137     name = music_file.gets()
138     location = music_file.gets()
139     id = i
140     @track = Track.new(name, location, id)
141
142     return @track
143 end
144
145
146 #prints all albums with album id added to each to make easy reference(called in
    ↪  display_album(file_name))
147 def print_albums(albums)
148     i=0
149     while (i < albums.length) do
150         print "Album ID:" + i.to_s + "\n"
151
152         print_album(@albums[i])
153         i = i +1
154     end
155
156 end
```

```
157
158 def print_album(album)
159
160     puts (album.artist.to_s())
161     puts (album.title.to_s())
162     puts('Genre is ' + album.genre.to_s())
163     puts (GENRE_NAMES[album.genre.to_i])
164     print_tracks(album.tracks)
165     #Note:  (album.tracks)is must! otherwise it wont realise that we are
166     #        printing tracks and instead will make a new null array!
167
168
169     # print out the tracks
170
171 end
172
173 # Takes an array of tracks and prints them to the terminal (called in
    ↪ print_album(album))
174 def print_tracks(tracks)
175     @No_of_items_in_tracks = @tracks.length
176     i=0
177     while (i < @tracks.length) do
178         print_track(@tracks[i])
179
180         i = i +1
181     end
182 end
183
184 def print_track(track)
185     puts(track.name)
186     puts(track.location)
187 end
188
189
190
191
192   # Draws the artwork on the screen for all the albums
193
194   def draw_albums albums  #(album)
195
196
197
198
199     @album_image.draw(10, 10, z = ZOrder::UI)                                    #here i
      ↪ have hardcoded the ablum_image to that of "neil2.png"in initialize as it
      ↪ said only 1 album will be taken as condition of output
200
201
202
203   end
204
205   # Detects if a 'mouse sensitive' area has been clicked on
206   # i.e either an album or a track. returns true or false
```

```ruby

  def area_clicked(leftX, topY, rightX, bottomY)

     if ((mouse_x >= leftX && mouse_x <= rightX) && (mouse_y >= topY && mouse_y <= bottomY))
      true

    else
      false
    end
  end


  # Takes a String title and an Integer ypos
  # You may want to use the following:
  def display_track
    ypos = 10    #gives intial y position
    i = 0

    while (i < @tracks.length) do
        @track_font.draw(@tracks[i].name, 300, ypos, @z_level, 1.0, 1.0, Gosu::Color::BLACK)
        i = i + 1
        ypos = ypos + 50
    end


  end


 # Takes a track index and an Album and plays the Track from the Album

  def playTrack(song, index)

        @song[index].play
        if(index ==@song.length-1)            #whether last song played

        @start=false                      #commented this part so that start stays true
        end                                  #and fulfil update condition



  end

# Draw a coloured background using TOP_COLOR and BOTTOM_COLOR

    def draw_background
    Gosu.draw_rect(0, 0, 600, 800, BOTTOM_COLOR, ZOrder::BACKGROUND, mode=:default)
    # Draw the rectangle that provides the background.
    Gosu.draw_rect(0, 10, 550, 800, TOP_COLOR, ZOrder::BACKGROUND, mode=:default)


```

```
257     end
258
259
260
261
262 # Not used? Everything depends on mouse actions.
263
264
265 def update
266
267
268     if(!@song[@index4].playing? && @start==true )                #not playing but
        ↪  start is true( it means location has been clicked)
269
270         @yposition1 += 50            #red box moves 50 boxes down
271         @index4 = @index4 +1
272         playTrack(@song, @index4)    #calls the function again, sort of like
            ↪  recursive
273
274
275
276     end
277
278 end
279
280  # Draws the album images and the track list for the selected album
281
282     def draw
283
284         draw_background
285         draw_albums (@albums)
286
287         display_track
288
289
290 Gosu.draw_rect(290, @yposition1, 5, 5,Gosu::Color::RED, @z_level, mode=:default)
291
292     end
293
294      def needs_cursor?; true; end
295
296     # If the button area (rectangle) has been clicked on change the background
        ↪  color
297     # also store the mouse_x and mouse_y attributes that we 'inherit' from Gosu
298     # you will learn about inheritance in the OOP unit - for now just accept that
299     # these are available and filled with the latest x and y locations of the mouse
        ↪  click.
300
301     def button_down(id)
302         case id
303         when Gosu::MsLeft
304             if area_clicked(10, 10, 300, 400)
305             @z_level = ZOrder::UI                  #made it update to foreground "UI"
```

```
306             @count = 1
307
308
309             @start = true                           #2nd condition to load the
                ↪  playtrack in update
310             @index4 = 0
311             playTrack(@songs, @index4)
312
313
314
315
316             end
317
318         end
319     end
320
321 end
322
323 # Show is a method that loops through update and draw
324
325 MusicPlayerMain.new.show if __FILE__ == $0
```


C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 7\7.2C-resources\Resources>gui_music_player.rb
Album ID:0
Neil Diamond
Greatest Hits
Genre is 1
Pop
Crackling Rose
sounds/01-Cracklin-rose.wav
Soolaimon
sounds/06-Soolaimon.wav
Sweet Caroline
sounds/20-Sweet_Caroline.wav
DEPRECATION WARNING: draw is deprecated; use Font#draw_text or Font#draw_markup instead.
draw called from C:/Users/SAIFUL/Desktop/swinburne class stuff/COS10009 (introduction to programming)/week 7/7.2C-resources/Resources/gui_music_player.rb:226:in `display_tr
ack'.
Music Player
NEIL DIAMOND
12 SONGS
Crackling Rose
Soolaimon
Sweet Caroline

35 Learning Summary

Learning Summary

Outcome	Weight
Functional Decomposition	♦♦♦◊◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦◊◊◊◊

Outcome	Weight
Apply Reading Techniques	♦♦♦♦◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Learning Summary

Submitted By:
S M Ragib Rezwan
103172423
2021/06/02 20:59

Tutor:
NAJAM NAZAR

June 2, 2021

COS10009 – Introduction to Programming

Learning Summary Report

SM RAGIB REZWAN (103172423)

Self-Assessment Details

The following checklists provide an overview of my self-assessment for this unit.

	Pass (D)	Credit (C)	Distinction (B)	High Distinction (A)
Self-Assessment (please tick)				✓

Self-assessment Statement

	Included (please tick)
Learning Summary Report	✓
Test 1 and Test 2 are Compete in Ed	✓
All Pass level tasks completed (including tutorial tasks)	✓

Minimum Pass Checklist

	Included (please tick)
All Credit Tasks are Complete in Ed	✓

Minimum Credit Checklist, in addition to Pass Checklist

	Included (please tick)
Distinction tasks (other than Custom Program) are Complete	✓
Custom program meets Distinction criteria & Interview booked	✓
Design report has structure chart and screenshots of program	✓

Minimum Distinction Checklist, in addition to Credit Checklist

	Included (please tick)
HD Project included	✓
Custom project meets HD requirements	✓

Minimum High Distinction Checklist, in addition to Distinction Checklist

Declaration

I declare that this portfolio is my individual work. I have not copied from any other student's work or from any other source except where due acknowledgment is made explicitly in the text, nor has any part of this submission been written for me by another person.

Signature:_____

Portfolio Overview

This portfolio includes work that demonstrates that I have achieve all Unit Learning Outcomes for COS10009 Unit Title to a **High Distinction** level.

This course helped me understand how to design and code in a structural procedural way in depth, using not only the coding language “ruby”, but also other such similar languages like “C” and “python”. It has accomplished this by teaching me crucial aspects in programming itself, regarding both code and design, via the help of lecture material, reference books, Doubtfire and Ed tasks, help desk sessions and tutorials.

Although it’s a bit difficult to pick out pieces of work that focuses on any single specific Unit Learning Outcomes (as they are all sort of mixed and merged in between the allocated tasks), I will try my best to give you a list of ones that I deem as my best piece of work in each of those category. (giving in order followed in Doubtfire portfolio task)

1) Functional Decomposition: This is basically the ability to break a large problem down in to smaller problems or functions.

   This is beneficial in many cases, like debugging (as we can just test each component or procedure separately instead of entire code at once), reusability (as we can just remove the specific functions we need, instead of taking entire code), understanding of entire code (as we can easily see which small part of code is doing what task, what is being passed and how it is all linked together), etc.

   This has been demonstrated in almost all of the tasks. But the one where I focused on this most was on task 7.1 (where I had to merge information from task 5.1, 5.2, 6.2 and 3.2) and 7.2 (where I had to merge information from task 7.2 with task 3.3 and 4.3). Hence I am keeping those as max level here and graduating down the levels for the rest. (I am not mentioning Foodhunter and maze task as max level here as even though they also have high level of modularisation, these were already set up as such in sample code and the amount I added extra, with regard to modularisation, was very little).

2) Structured Coding principles: There are mainly 4 structured principles:
   a. Sequence: The code is written and executed following a certain order
   b. Repetition: Using for, while, etc. loops to iterate through a block of code until starting condition becomes false.
   c. Selection: Using if, elsif (ruby used this instead of else if), else, case (or switch), etc. to run only a specific case or condition out of the many options given.
   d. Modularisation: Breaking down a code into smaller parts and moving them out of the main code (and putting inside functions and procedures) with the main intention of reusing them in a similar code.

   This is a crucial aspect of structured programming and it is impossible to write a decent code without using any of these (unless the code is a “hello world” type or a poorly written, small code like printing a single line or value). Moreover, in case of big and complicated codes (like in making games, music player, network, etc), where many parts are working in unison and passing data to and from the functions, there is usually combination of all these principles being used there.

   Hence, after contemplating on this for some time, I have decided to give all codes a max rating (except the “hello world” type codes where I am giving a rating of 1) in this regard. That’s because the structured coding principles are like the building block for each and every code and hence I believe that all codes are highly relevant to it.

   I am also giving the task given in concept map as max rating as well as there I had to compare OOP with structural programming (although out of the 4 core principles,

I only focused on modularisation there and the other point was on information hiding which is not part of the core principles)

3) Programming: This basically means being able to code efficiently in the languages covered by utilizing all the techniques and functions provided in it (i.e. if, selection, array, looping, functions and procedures, data type, local and global scoping, classes, parameter, pointers, etc.)

   For this case, I will refer to a famous quote spoken by Henrik Ibsen “A thousand words will not leave so deep an impression as one’s deed”

   So, instead of speaking in detail about each and every one of the codes I have written throughout this semester and submitted on Doubtfire, I believe it will be better to have a glimpse though them and then decide whether I can code or not. (Here I am rating them with respect to type of task they are (T or P or C or D or HD) instead of their relevance as they are all programs and in a sense, they are all equally highly relevant!)

   Furthermore, if you look through it all chronologically (i.e. following the week order) instead of rating order, you will also be able to see how I have grown over time with regards to programming, from the ignorant fool (who couldn’t even input and output variables) to a decent coder (who can analyze and modify codes on the fly).

   [if you are short on time, you can quickly check my growth and understanding by comparing the code I had written at fist in 4.4 and one I wrote for 10.3 and seeing the change I made in the neighbor linking part in order to make sure the code executes as intended.]

.

4) Code reading or debugging: This is an absolutely crucial skill without which a person can’t really call himself to be programmer.

   Its sort like the fact that you need to both read and write to be able to call yourself literate. Till now, all the points we have looked at were mostly focused on writing the code itself and how it looked like. But without being able to understand what you are seeing (i.e. being able to read the code properly), a person will never be able to code programs in depth or help fellow programmers debug their code.

   Moreover, this is a must with respect to career as after joining a company for the first time, you will be following up with a senior programmer where your task will be to go through his code and help him debug it for errors. There if you can’t read code, you can’t properly debug them and hence might even lose the job.

   Technically, most of the tasks submitted on Doubtfire are part of code reading (as there I just had to add or alter a program given in order to make sure it does the desired work and hence provide the desired output). But debugging mainly on took place in tasks 2.1, 9.1, 12.1. So I am giving those specific task as max level rating and others as low level rating.

Since I have been aiming for HD level Band 2 (i.e. 90 to 100), I have completed all the T level, P level, C level, D level and HD level tasks. In case of music player, I have completed up 7.2 (ie the simple GUI one) as Matt had said those who are going for Custom Program and custom project for HD route, do not need to do 7.3 music player task.

Furthermore, as already mentioned, I have done a custom code where I not only used and linked everything I have learned so far from the course, but also used some interesting things

I have found while coding (like being able to restrict button selection to a specific z order level, adding in Enemy ai, creating a custom timer without using Gosu's inbuilt one, etc.). Furthermore, after implementing an Enemy Ai in the code, I had become really curious about the topic, leading me to research on the concept in depth and ended up writing a paper regarding what I have found on it. Thus, I believe I should achieve the HD level 2 band's mark.

Reflection

The most important things I learnt:

Well, truth to be told, I had never actually coded before coming to this course. That's because even though my school had ICT subject, it lacked programming topics in its syllabus and instead prioritized on general knowledge stuff. So everything I learnt here, from basic topics (like reading in values, code syntax, etc.) to the complicated ones (like debugging, 2D arrays, etc.) has been extremely important in my eyes.

But surprisingly, the most important thing I learnt here wasn't actually the key concepts. Instead it was the value of the following sequence of steps: keeping calm in frustrating situations (preferably by taking deep breaths), analysing the questions carefully, noting down possible solutions in copy, testing each and every one of the solutions until desired output has been obtained and cleaning up code and writing comments to make sure I don't fall into the same problem next time.

Although it might seem like a fairly simple sequence of steps, it has been critical in helping me solve the problems I had faced while coding. Thus I am ranking this at the very top.

The things that helped me most were:

1)Lectures (as they guided me though important key concepts and provided hint on tasks)

2) Doubtfire and Ed tasks (as they helped not only reinforce my knowledge but also let me test out what the functions can and can't do)

3) Ruby and Gosu documentations (as they helped me find out important syntax and in built functions and classes)

4)Tutors (as they helped fully explain what task really requires, break simplify it and provide necessary detailed feedback)

5)Helpdesks (as they gave a further opportunity to ask questions and thus solve confusions)

6)Doubtfire's inbuilt deadlines for each tasks (as it helped make sure I finished all my weekly tasks within the week, instead of letting them all pile up until the end)

I found the following topics particularly challenging:

Well, since it was my first time coding, I actually found all of them more or less challenging. But the most challenging ones were:

1)Multi-array (as I didn't understand how they were being set up at first. But then again, it can be because back then I hadn't really understood what a normal array did and hence seeing the code for 4.4 had crashed my brain at that time)

2)Pathfinding task in maze (as I had just come across details on that type of algorithm for the very first time in a different course and thus wasn't able to understand what was going on back then. But thankfully, by the week 10's end, I had been able to understand what the new code was actually supposed to do, modify my previous 4.4 code appropriately and could finally see the task for the simple thing it is, instead of over complicating it)

3) Solving "Rendering Error" in Gosu (as the solution was to separate the image of objects and motion of the object into two different parts which went against my plan of keeping all the information about the object in a single module for later reusability and code-snippet testing)

I found the following topics particularly interesting:

1) Classes (as it provided me with a brand new way of scoping (other than using local and global) which in turn, helped me code with further ease)
2) Recursion (as I had learnt about the concept in my high school mathematics class in induction topic and thus implementing in a code was both nostalgic and fun)

I feel I learnt these topics, concepts, and/or tools really well:

1) Looping with while, begin, for, etc.
2) Passing data into different classes and creating new arrays and objects to store them
3) Selection using if, case, etc.
4) Selecting specific elements in an array, add or removing data at certain parts or to all part of array, keeping count of no of elements in array, passing data by piping
5) Debugging using puts, testing code snippets, using test harness, defensive programming and validation, etc.

There are many more that can be added to the list. But then it would become too long and end up echoing out almost the entire syllabus. So instead I just listed my top 5 here.

I believe the evidence for these and more can be clearly seen in the codes I have submitted as my portfolio via Doubtfire and thus am not elaborating it again here.

I still need to work on the following areas:

Currently I am trying to work out when to use which level of coupling and cohesion and how far I should go in case of modularization in designing my codes.

Furthermore, I am also wondering whether it will be ok or not to use multiple levels of coupling and cohesion throughout in my code, and if so, I am curious of which level of coupling and cohesion I should call my entire code as.

This unit will help me in the future:

I believe this unit will help me make a decision on where I wish to go in life, how far I want to delve in the world of coding and thus, fix my career path. But of most of all, I believe it will work as an encouragement for me, showing me that as long as a I work hard enough, I can not only understand any new concepts but can also master them.

If I did this unit again I would do the following things differently:

1) From day 1, I would look up a table of all possible syntax and logical errors the program can output, what they each mean and how to resolve them. This would have helped me resolve lots of silly mistakes in short time, instead of spending hours and hours trying the modify a part of the code, only to realise in the end that all I am missing is just a bracket or an “end”.
2) Attend all helpdesk from the very first no matter what, not hesitate with my problems and not let embarrassment hold me back from resolving my problems
3) Attend every tutor’s classes (as long as they don’t mind me participating in it) from the very first week and listen to what they say there. That’s because different people have different views and thoughts about the same problem and listening to them can help find new insights and ideas about a problem or topic.
4) Not get too worried if unable to do a task at the first try. Instead, take breaks, think about what’s going wrong and try again and again.

Other:

36 Python Shape Moving

A shape moving game using pygame.

Outcome	Weight
Functional Decomposition	♦◊◊◊◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦♦♦◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

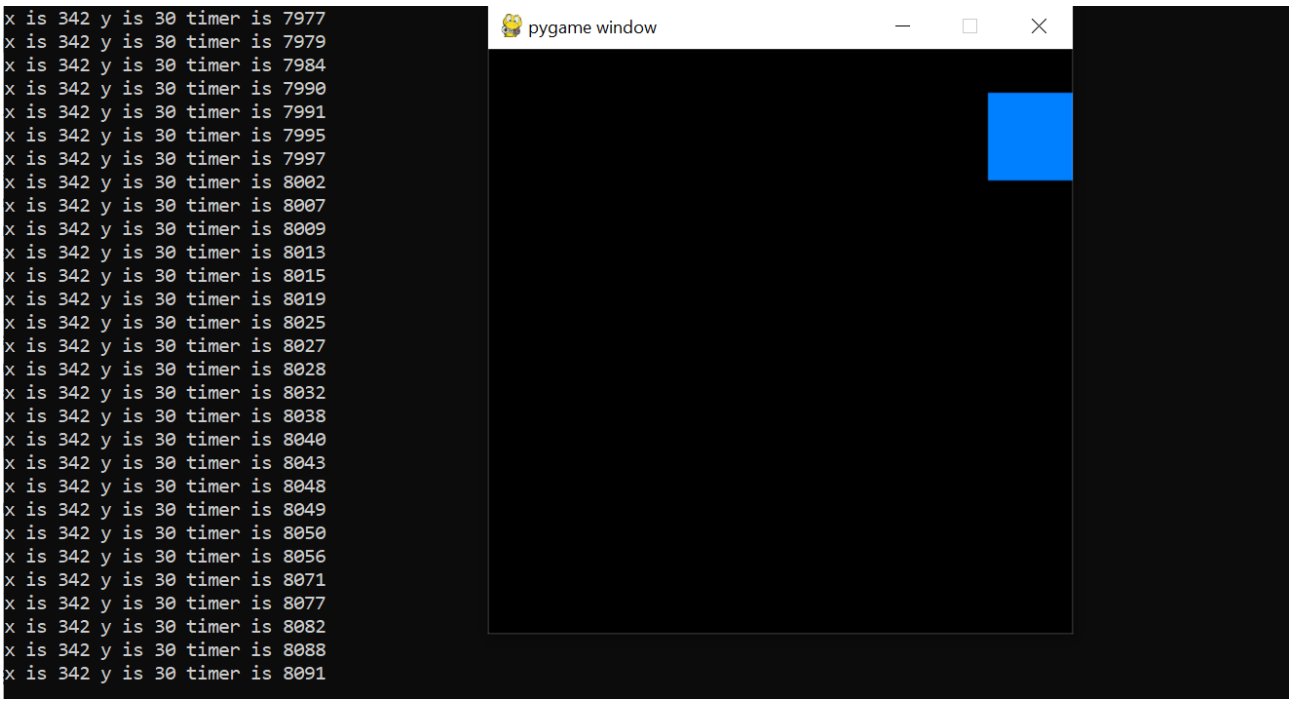
Python Shape Moving

Submitted By:
S M Ragib Rezwan
103172423
2021/05/17 10:10

Tutor:
NAJAM NAZAR

May 18, 2021

```
# Acknowledgement to the original authors of the code on which this
# example is based.
import pygame

pygame.init()

SCREEN_HEIGHT = 400
SCREEN_WIDTH = 400

screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
done = False
is_blue = True
x = 30
y = 30

time = pygame.time

while not done:
        for event in pygame.event.get():
                if event.type == pygame.QUIT:
                        done = True
                if event.type == pygame.KEYDOWN and event.key == pygame.K_SPACE:
                        is_blue = not is_blue

        pressed = pygame.key.get_pressed()

        # subtracted width and height of shape(60,60) from screen width and screen
        ↪  height to make sure entire object stays inside the screen
        if pressed[pygame.K_LEFT] and x >= 0 : x -= 3
        if pressed[pygame.K_RIGHT]and x <= (SCREEN_WIDTH -60): x += 3
        if pressed[pygame.K_UP] and y >=0: y -= 3
        if pressed[pygame.K_DOWN] and y <= (SCREEN_HEIGHT - 60): y += 3

        print(f"x is {x} y is {y} timer is {time.get_ticks()}")

        screen.fill((0, 0, 0))
        if is_blue: color = (0, 128, 255)
        else: color = (255, 100, 0)
        #the width and height of shape is 60 x 60 set here
        rect = pygame.Rect(x, y, 60, 60)
        pygame.draw.rect(screen, color, rect)

        pygame.display.flip()
```


x is 342 y is 30 timer is 7977
x is 342 y is 30 timer is 7979
x is 342 y is 30 timer is 7984
x is 342 y is 30 timer is 7990
x is 342 y is 30 timer is 7991
x is 342 y is 30 timer is 7995
x is 342 y is 30 timer is 7997
x is 342 y is 30 timer is 8002
x is 342 y is 30 timer is 8007
x is 342 y is 30 timer is 8009
x is 342 y is 30 timer is 8013
x is 342 y is 30 timer is 8015
x is 342 y is 30 timer is 8019
x is 342 y is 30 timer is 8025
x is 342 y is 30 timer is 8027
x is 342 y is 30 timer is 8028
x is 342 y is 30 timer is 8032
x is 342 y is 30 timer is 8038
x is 342 y is 30 timer is 8040
x is 342 y is 30 timer is 8043
x is 342 y is 30 timer is 8048
x is 342 y is 30 timer is 8049
x is 342 y is 30 timer is 8050
x is 342 y is 30 timer is 8056
x is 342 y is 30 timer is 8071
x is 342 y is 30 timer is 8077
x is 342 y is 30 timer is 8082
x is 342 y is 30 timer is 8088
x is 342 y is 30 timer is 8091
pygame window

37 Python Hello World

A Python program to print Hello World

Outcome	Weight
Functional Decomposition	♦◊◊◊◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦♦◊◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Python Hello World

Submitted By:
S M Ragib Rezwan
103172423
2021/05/11 12:08

Tutor:
NAJAM NAZAR

May 11, 2021

```
1 print ("Hello World!")
```

```
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week10\10.2T>hello_world.py
Hello World!
```

38 Python Silly Name Program

Complete a Python "Silly Name" program.

Outcome	Weight
Functional Decomposition	♦♦◊◊◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦♦◊◊◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Python Silly Name Program

Submitted By:
S M Ragib Rezwan
103172423
2021/05/17 09:30

Tutor:
NAJAM NAZAR

May 18, 2021

```
# write code that reads in a user's name from the terminal.  If the name matches
# your name or your tutor's name, then print out "<Name> is an awesome name!"
# Otherwise call a function called print_silly_name(name) - which you must write -
# that prints out "<Name> is a " then print 'silly' (60 times) on one long line
# then print ' name.'

def print_silly_name(name):
    # complete this
    i = 0
    print(f"{name} is a ", end='') # f allows {name}
    while(i<60):
        print("silly ", end='')  # end='' has no newline char
        i = i + 1
    print("name")
    return

def main():
    name = input("What is your name? ")
    if (name == "Ragib") or (name == "Nazam"):
        print(name + " is an awesome name!")
    else:
        print_silly_name(name)
    return

main()
```

```
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week11\11.2P-resources\Resources>silly-name.py
What is your name? jock
jock is a silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly
silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly sill
y silly silly silly silly name
```

39 C program

C program

Outcome	Weight
Functional Decomposition	♦♦◊◊◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦♦♦♦◊

Outcome	Weight
Apply Reading Techniques	♦◊◊◊◊

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

C program

Submitted By:
S M Ragib Rezwan
103172423
2021/05/17 11:26

Tutor:
NAJAM NAZAR

May 18, 2021

```c
#include <stdio.h>
#include <string.h>
#include "terminal_user_input.h"

#define LOOP_COUNT 60

void print_silly_name(my_string name){
    int index;
    for(index=0;index<LOOP_COUNT;index++) {

    printf(" a silly ");

  }
  printf(" name!\n");
}

int main()
{
  my_string name;


  name = read_string("What is your name? ");
    printf("\nYour name");


  // Move the following code into a procedure
  // ie:  void print_silly_name(my_string name)

  if(strcmp(name.str, "YOUR_TUTOR_NAME")==0)
  {
      printf(" is an AWESOME name!");
  }
  else{
      printf(" %s is ", name.str);
      print_silly_name(name);
  }



  return 0;
}
```

```
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week11\11.4D-resources\Resources>gcc terminal_user_input.h terminal_user_input.c name_t
ester.c

C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week11\11.4D-resources\Resources>a
What is your name? YOUR_TUTOR_NAME

Your name is an AWESOME name!
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week11\11.4D-resources\Resources>a
What is your name? Fred

Your name Fred is  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly
a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a
 silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a silly  a
silly  a silly  a silly  a silly  a silly  name!

C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week11\11.4D-resources\Resources>
```

40 Test Harness

Complete a test harness to test a simple program.

Outcome	Weight
Functional Decomposition	♦♦◊◊◊

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

Outcome	Weight
Programming	♦♦♦◊◊

Outcome	Weight
Apply Reading Techniques	♦♦♦♦♦

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Test Harness

Submitted By:
S M Ragib Rezwan
103172423
2021/05/26 10:18

Tutor:
NAJAM NAZAR

May 27, 2021

```ruby
# tester_demo.rb
require "minitest/autorun"
require_relative "student"

class StudentTest < Minitest::Test

  def test_student_id_is_integer
    assert_kind_of Integer, get_student_id(2)

  end



# insert a test here for the finding the correct student for id 300
def test_correct_student_for_id_300
    assert_equal( "Jill", get_student_name_for_id(300), "INVALID")
    end


# insert a test here for returning "Not Found" for student with id 800
def test_not_found_for_id_800

    assert_equal( "Not Found", get_student_name_for_id(800), "Invalid")
    end
# # insert a test here for finding the correct student name for array position 0
def test_first_student_name_in_array
    assert_equal( "Fred", get_student_name(0), "Invalid")
    end
end
```

```
C:\Users\SAIFUL\Desktop\swinburne class stuff\COS10009 (introduction to programming)\week 12\12.1C-resources\Resources>ruby tester_demo.rb
100
Jenny
Jill
Run options: --seed 27732

# Running:

....

Finished in 0.007168s, 558.0513 runs/s, 558.0513 assertions/s.

4 runs, 4 assertions, 0 failures, 0 errors, 0 skips
```

41 10.4HD Custom Project

Custom Project

Outcome	Weight
Functional Decomposition	♦♦♦◊◊

Outcome	Weight
Structured Coding Principles	♦♦◊◊◊

Outcome	Weight
Programming	♦♦♦♦◊

Outcome	Weight
Apply Reading Techniques	♦♦♦♦♦

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

10.4HD Custom Project

Submitted By:
S M Ragib Rezwan
103172423
2021/05/28 10:01

Tutor:
NAJAM NAZAR

May 28, 2021

Name: SM RAGIB REZWAN

ID: 103172423

Course: COS10009 (Introduction to Programming)

Research project title: Enemy/NPC AI

ENEMY/NPC AI:

Contents

What is AI?

In order to understand this, one must first understand what an AI is. Although multiple definitions have been given over time, the most accepted definition of AI is:

“the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings.”

Let us consider an example to understand this. Imagine you are writing an essay topic. There you are pondering about a topic, using your logic and assumptions to come to a conclusion that has a certain meaning for you. Then you go through it again (with help of internet or peers or a teacher), find your mistakes (if any any) and learn from it, so as not to repeat it.

This is an example of an action performed by an intelligent being. Basically it’s the ability to think, use logic, assign meaning, generalize and learn from past mistakes. And in AI, we are trying to replicate that intelligence, except in a machine instead of a human.

What is Enemy/NPC AI?

“Enemy/ NPC AI is the use of that AI system in game characters in order to make the game more fun and life like.”

This is the definition you will find in multiple gaming webpages. But in most cases, it’s actually a lie (like for the cases in old games). Instead it will be more accurate to say it as “pseudo AI” where the game characters are being moved in such a way that it seems as if they have their own intelligence.

It’s easier to understand if you consider a puppet show. There, dolls are being moved across the stage, using strings, in order to perform certain actions like walking. The puppets there have no intelligence, but because they are being moved in different ways with respect to time, alongside interesting narration and artwork, it seems as if they have intelligence.

This is what is being done in most games. Amazing graphics and narrations are being provided to make sure the game characters only need to perform certain actions (like only moving back and forth on a platform as they are guarding that area) and hence use it to trick the gamer into thinking that the Enemy or NPC has intelligence.

But that isn’t always the case, with modern games actually trying to implement AIs in the game characters’ movements with varying degree of success (unlike past ones that just used the “AI” word to increase the popularity of their game). But even so, Enemy/NPC AI still doesn’t have actual intelligence. Instead, it’s more accurate to say that it has “a pseudo AI that is extremely close to AI”, with each new game getting closer and closer (sort of like “fake it till you make it” type scenario).

How it has been implemented over the years?

Well, various games have implemented it in various ways; some simple, some complicated. But they all can be simplified into the following: (I have stated it in Gosu terms to make it easier to understand)

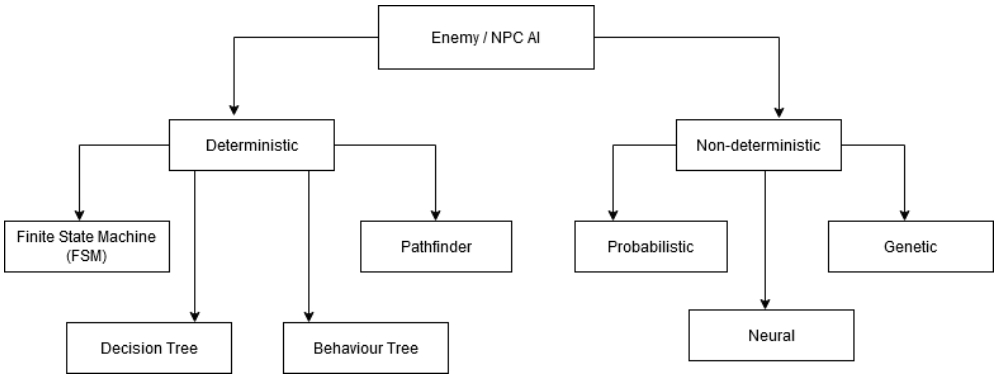
1. Outside initialize, draw and update:
   Create a class for the character containing “keywords” relevant to it
2. In initialize:
   Create variable or image file address to link with the “keywords” in order to call it throughout in the code and also set up triggers

3. In draw:
   Call up images or shapes to link with the character and its appearance when performing actions
4. In Update:
   Setup values to movement variables and give conditions to make the character perform certain actions in certain way at certain time, preferably with the help of algorithms.

So how to differentiate between different types of Enemy/NPC AI?

Well, if you ask a gamer, he will probably differentiate it via the actions performed by it (like talk, defend, attack, use weapons, use potions, barter, etc.) or the way it looks (i.e. the number of detailed graphics or sprites assignment to its movements). But in a programmer's eye, there is actually a simpler, more concrete way to differentiate between them, via comparing the algorithms they use.

Different types of Enemy/NPC AI: (using algorithms)



Overall there is 2 different types of such AI (with their own sub classifications):

1) Deterministic Algorithms:

These are the types of algorithms that use some underlying condition to produce the same output for a specific input (i.e. if player jumps, the enemy will jump as well).

Code wise, a simplified deterministic algorithm will look something like this (using FSM here):

```
Enemy.each do  |c|
    if((player.x-c.x).abs <50 && (player.y-c.y).abs <50)
            @trigger_Enemy_Death = true

    end

end
```

Basically, the following things happen here:
If (player x and y - enemy x and y < 50), then show animation of player death

There are usually 4 types of such algorithm used in Enemy/NPC AI:

a) Finite State Machine(FSM):

Definition: "A Finite-State Machine is a formal model of any system/machine/algorithm that can exist in a finite number of states and that transitions among those states based on sequence of input symbols"

Let us take an example to better understand this. Imagine you are talking with an NPC in a village. There, in most games, you will find the NPC can only answer certain questions where they tell you different information depending on what you choose and perform different sequence of movement actions (like waving hand while explaining or being extremely happy). But no matter what you do, for the same option selected, they repeat the same conversation and actions with no variation what so ever.

This is an example of a Finite State Machine algorithm where the programmer made sure that the bot can only experience a certain set of events and assigned specific set of responses to perform for each.

You can see this being performed in Wolfenstein 3D where they thought of all the situations an enemy solder can experience like a door being opened, player moving nearby, getting themselves shot, etc.

Code wise, a FSM will look something akin to this:
(although it looks very modular, keep in mind, this modularity is because of Gosu and not because of the FSM! And thus, when I am comparing modularity in the writing, I mean only the algorithm's modularity, not the ones that formed due to Gosu's syntax!)

```
Enemy.each do  |c|
    if((player.x-c.x).abs <50 && (player.y-c.y).abs <50)
            @trigger_Enemy_Death = true

    end

end
```

Basically, it will do the following:

If (player x and y - enemy x and y < 50), then show animation of player death

b) Decision Tree:

Definition: "It is a decision support tool that uses a tree-like model of decisions and their possible consequences. It is one way to display an algorithm that only contains conditional control statements and is a popular tool in machine learning algorithm"

This is similar to FSM, except here we can consider for a bit more complex situation.

For instance, let's go back to the scenario where we are talking with the NPC. Now consider the case where the background music changes (i.e., before you interact with NPC, there is a village song being played in background but during your interaction a different song is played).

If you try to do this using FSM, you need to set up two different FSM with two different condition. As more elements run in the background, these keep adding up and hence you end up an enormous amount of FSM to be used just to implement a small feature.

Hence we use Decision tree instead where we design a specific tree with a node for NPC which will make program run a specific set of commands (like turn off other music and play NPC music instead, selected option has certain outcome, blur background object or zoom into NPC, etc.) (basically Decision tree is like a hierarchical FSM where the codes are highly modular and hence can be reused more easily)

This type has been used extensively while coding for the AI of alien in Alien: Isolation game. (But it will be better to call it as behavior tree example instead as it is far more complex than that provided by decision tree)

In code you can implement this for Enemy/NPC AI by setting up some nested condition that calls up functions:

```
    Enemy.each do  |c|

    while((player.x-c.x).abs <250 && (player.y-c.y).abs <250)
    play_enemy_music
    ram_player(c)
    end

    enemy_movement_normal(c)
  end
```

```
    def ram_player(c)

        if((player.x-c.x).abs <50 && (player.y-c.y).abs <50)
        @trigger_Enemy_Death = true
        else
            if (player.x < c.x)
            c.x = c.x - 1
            elsif (player.x > c.x)
            c.x = c.x + 1
            end
        end

end
```

Basically, it will do the following:

while player x and y – enemy x and y <250 (for any enemy x and y) (since each tile is 50, it means 5 tiles)
Then play sound/music
Also call a function to “run at player and try to reach him)
Inside the function, if it touches player, then run player death scene function

End while

Run a function that calls up preset enemy movement for normal condition

Inside it, run another function for platform (ie standing?)

Where the condition is, if platform is there then running

Else fall down

c) Behavior Tree:

Definition: "A behavior tree is a mathematical model of plan execution used in computer science, robotics, control systems and video games. They describe switching between a finite set of tasks in a modular fashion. Their strength comes from their ability to create very complex tasks composed of simple tasks, without worrying how the simple tasks are implemented."

Its definition is also similar to that of decision tree, but it's a bit more complex than that.
That's because here, the order it uses to go through the nodes are different. In decision tree, it would go from root to child node and would continue until end is reached, upon which it will output condition given and stop.

But here in behavior tree, each child nodes are based on priority. So, it will go down each node with respect to order of priority until end and only if condition matched 100%, it will run condition. But if not matched completely, it will check other nodes to see which one has the fully matched condition.

Let us look back to the case with the NPC conversation in the village scenario. Imagine a raid occurred all of a sudden with Ogres attacking the village. This will take priority over the NPC conversation (as otherwise the player may be killed and game may end) and hence will stop the conversation with dialogue like "Oh no, the ogres are attacking! We can't converse now!". Instead the function for "village being attacked" will run, or more specifically, function for "village being attacked by orges" or the "orge" child node for the "village being attacked" branch, will run and function for player movement + fighting will be activated.

This has been used in games like Alien isolation (as mentioned before) Halo, Bioshock, Spore,etc

In code you can implement this for Enemy/NPC AI by setting up some nested condition that calls up functions, except this time we set up a simple priority system:

(it is similar to decision system as we used while and if in both. In game found online, there will be more complicated conditions that must be fulfilled before certain actions can be performed. That is where decision system can show its true worth)

```
  Enemy.each do  |c|

  while((player.x-c.x).abs <250 && (player.y-c.y).abs <250)
  play_enemy_music
  ram_player(c)
  end

  enemy_movement_normal(c)

  if ((player.x-c.x).abs >750 && (player.y-c.y).abs <750)
      enemy_idle(c)
  end
end
```

```
   def ram_player(c)

      if((player.x-c.x).abs <50 && (player.y-c.y).abs <50)
      @trigger_Enemy_Death = true
      else
          if (player.x < c.x)
          c.x = c.x - 1
          elsif (player.x > c.x)
          c.x = c.x + 1
          end
      end

end


def enemy_idle(c)
    c.x=0
    c.y=0
end
```

Basically, it will do the following,

while player x and y – enemy x and y <250 (for any enemy x and y) (since each pixel is 50, it means 5 tiles)
Then play sound/music

And then, call a function to (run at player and try to reach him)
Inside the function: if it touches player, then run player death scene function
End while

Run a function that calls up preset enemy movement for normal condition

Inside it, run another function for platform (ie standing?)

Where the condition is, if platform is there then running

Else fall down

If player x and y minus enemy x and y is greater than 750 (for any enemy x and y)) (since each pixel is 50, it means 15 tiles),
run enemy idle function
inside function: Enemy stationary as x and y increment is 0

d) Pathfinding:

Definition: these are algorithms that are used to find shortest path between two nodes (commonly used are **Dijkstra, A* algorithm, etc.**)

Basically the thought behind this concept is something akin to this, if an Enemy/NPC can't navigate the map on its own, what sort of intelligence does it have?

So, let's go back to our previously thought scenario. When the ogres attacked, the NPCs need to run away to safe spot. In order to do that they need to navigate through the map and find the shortest path to safe zone, without coming into collision with objects like walls, or falling into dead ends.

This has been mostly used in games where enemies chase or run away from you, like for creeper in Minecraft, bosses in terraria, etc.

This is done with the implementation of a path finding algorithm where basically the following is done: (written the one where NPC is finding a save path to escape)

1) An array of all cells are taken with conditions given to distinguish free cell from those which are walls or obstacles.
2) Then the neighboring cells are connected with "safe zone" cells being distinguished in some way (like using nil).
3) After this, denote cell as:
   a) vacant if free cell (i.e. not wall or object),
   b) visited if it has been in that cell before (preventing it from looping around in a circle),
   c) path_north if there is north path from cell,
   d) path_south if there is south path from cell,
   e) path_east if there is east path from cell,
   f) path_west if there is west path from cell.
4) Then call a recursive function that goes through the entire thing and making sure it is not a dead end and that there is a path to safe zone (i.e. group of cells which is safe zone)

This is an extremely large code so i just explained the concept behind it, instead of giving code. If you wish to see the code, it will be better to refer to task 10.3HD that has been submitted in Doubtfire.

2) Non-deterministic:

These are the types of algorithms that produces different outputs in different runs, even though input remains same. Hence its behavior differs each time it is run.

Code wise, a simplified non-deterministic algorithm will look something like this (using probabilistic one here):

```
    Enemy.each do  |c|
        while((player.x-c.x).abs <250 && (player.y-c.y <250))
        play_enemy_music
            if rand(100)<80
            stop_and_growl_at_player(c)
            else
            ram_player(c)
            end
        end

    enemy_movement_normal(c)
  end

  def ram_player(c)

      if((player.x-c.x).abs <50 && (player.y-c.y).abs <50)
      @trigger_Enemy_Death = true
      else
          if (player.x < c.x)
          c.x = c.x - 1
          elsif (player.x > c.x)
          c.x = c.x + 1
          end
      end

end

def stop_and_growl_at_player(c)
enemy_idle(c)
@growlsound=true
end

def enemy_idle(c)
    c.x=0
    c.y=0
end
```

Basically, it will do the following extra part:

If rand(100)<80
    Function for enemy to stop and growl at player
Else
        Function to Ram the player
End if

Usually there is 3 such types used in Enemy/NPC AI:

a) Probabilistic method:

Definition: "A probabilistic algorithm is an algorithm whose behavior is partly controlled by random events"

Basically, it's the case where randomness has been introduced in an Enemy/NPC actions to make it seem more natural, lifelike and unpredictable.
Let's come back to that scenario we were talking about previously with the orge attacking the village while the NPCs are running away. Now, you have run toward the nearest orge group to fight and kill them.

Here, if it is a modern game, you will see that the orges don't always give the same attack. Some start by swinging with an axe, some start by punching, some start by stomping. (even their appearance may be slightly different)

Even though they are all orges and have the same attacks coded in them, their attack patterns are slightly different. That's because each attacks have a certain probability coded in them, like 50% change for axe swing, 25% for punch, 15% for stomp, etc. Hence they don't always end up doing the same thing, even though the same input has been provided (i.e. you engaging them)

This can be best seen in movement pattern of pets in games, like movement of fairies in Zelda, Plessie in super Mario 3d World, etc.

In codes this can be easily done by introducing a random function as a condition before implementing a function to take action:

```
Enemy.each do  |c|
    while((player.x-c.x).abs <250 && (player.y-c.y <250))
    play_enemy_music
        if rand(100)<80
        stop_and_growl_at_player(c)
        else
        ram_player(c)
        end
    end

  enemy_movement_normal(c)
end
```

```
    def ram_player(c)

        if((player.x-c.x).abs <50 && (player.y-c.y).abs <50)
        @trigger_Enemy_Death = true
        else
            if (player.x < c.x)
            c.x = c.x - 1
            elsif (player.x > c.x)
            c.x = c.x + 1
            end
        end

end

def stop_and_growl_at_player(c)
enemy_idle(c)
@growlsound=true
end

def enemy_idle(c)
    c.x=0
    c.y=0
end
```

Basically, it will do the following extra part:

If rand(100)<80
    Function for enemy to stop and growl at player
Else
        Function to Ram the player
End if

b) Neural method (or artificial neural network (ANN)):

Definition: “A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates”

Simply said, here we try to use algorithms to recreate the way our brain (or mainly neurons in the brain) thinks in response to different objects and situations, in order to simulate intelligence. We do this with the help of weight directed graphs and nodes where the weighed input can only be passed if it crosses a certain threshold.

This is different from probabilistic one as here we are trying to make the program actually think in the way we do, instead of giving a random event that occurs with a specific probability.

Now, back to the scenario we were speaking about. If it is a really, really good game, then it will also include in the conditions that will make the other orges (ones who haven attacked yet) compare their peer’s battles and outcomes and use it to fine-tune or modify their strategy. This

will end up making the battle closer to reality and hence give you a feeling of accomplishment, once you defeat them.

This has been first used in the game Creatures (1996) (an artificial life simulation game) in the characters NORNs where either the user teaches them how to behave, or they learn in by themselves

This is a bit difficult to implement in games as you need to consider several things.
But if simplified, it basically comprises of the following (in case of enemy and player):
1) range of movement possible by Enemy,
2) conditions that can be thought of partial success (ie moving towards and damaging player),
3) condition for final success (player killed)
4) an algorithm that will promote the program into making the enemy hurt and kill the player (like by using point system and asking system to follow the path that leads to max point over time)

c) Genetic algorithm:

Definition: "A genetic algorithm is a search heuristic that is inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation."

This is same as what had been taught to all students in biology in topic of genetic, except here we have implemented it as a form of algorithm where it takes a group of solutions to a problem as a "population" and uses the genetic crossing and mutations in order to find the best possible solution to the problem.

This leads to similar output to that which occurs in case or neural method. But here the output is more fine-tuned as we are comparing several generations and selecting the best one.

So coming back to scenario when the orges are attacking the village. After defeating the group, you get a notice that first wave is over and second wave is beginning. Now when you fight against the new group of orges, you can see that their strategy has vastly changed and that your previously created pattern against them isn't working. So you decide to implement a new one, based on the enemies and barely manage to defeat them.

This leads to a message similar to the previous where it says 2nd wave is over and prepare for final wave. Now when the orges attack, they are using vastly different strategies when compared to previous generation (like all attacking together but with different moves covering different spots, making it impossible for you to escape). This drastic yet fine-tuned change over the waves of enemies is best implemented using genetic algorithm and leads to extreme gameplay levels (if implemented properly) and gives you a true sense of challenge.

This still hasn't been implemented in large scale games, but has been implemented in small scale ones as a way to test the concept. For instance, it has been used to modify the actions and movement of enemy spaceships in a modified version space invaders game.
(But mostly it is being used to playtest different 2D game and find interesting strategies to go through the levels.)

It is more difficult to code this, compared to than neural one, as here you need to code in functions to consider different generations, cross parts from successful functions (exchange certain modules), mutate different functions (modify certain modules), and also write functions for initial population (all possible functions that lead to the desired solution/outcomes)

Conclusion:

As time passes and our understanding about AI improves (i.e. more accurate algorithm sets come out), we gain more opportunities to improve the Enemy/NPC AI implemented in our games. Hence, if we consider logically, the difficulty and realism should end up drastically increasing over time.

But that is not really the case in reality. That's because there is also another factor that affects what level of AI is implemented in game: "the desire of the gamers"

The gamers or players usually play games to remove their minds from the monotonous and stressful lives that they have. So when they play a game, they do not want to be "monotonous and predictable", nor do they want it to be "hard and stressful". Instead they seek for a middle ground where it is both challenging and fun.

So, the best way to promote the implementation of actual, high level AIs in Enemies and NPCs in games is to not only increase our understanding about such AIs, but to also encourage the players to want it as well.

REFERENCES:

https://en.wikipedia.org/wiki/Artificial_intelligence_in_video_games

https://videogameintelligence.com/classifying-game-ai/

https://thenextweb.com/news/a-beginners-guide-to-ai-the-difference-between-video-game-ai-and-real-ai

https://www.britannica.com/technology/artificial-intelligence

https://en.wikipedia.org/wiki/Deterministic_algorithm

https://gamedev.stackexchange.com/questions/51693/difference-between-decision-trees-behavior-trees-for-game-ai

https://en.wikipedia.org/wiki/Decision_tree

https://gamedev.stackexchange.com/questions/51693/difference-between-decision-trees-behavior-trees-for-game-ai

https://en.wikipedia.org/wiki/Nondeterministic_algorithm

https://www.investopedia.com/terms/n/neuralnetwork.asp

42 Custom Code Video

Custom Code Video

Outcome	Weight
Functional Decomposition	♦♦♦♦♦

As it is link to custom code video, i am giving it same ranking as custom code

Outcome	Weight
Structured Coding Principles	♦♦♦♦♦

As it is link to custom code video, i am giving it same ranking as custom code

Outcome	Weight
Programming	♦♦♦♦♦

As it is link to custom code video, i am giving it same ranking as custom code

Outcome	Weight
Apply Reading Techniques	♦♦♦♦♦

As it is link to custom code video, i am giving it same ranking as custom code

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Custom Code Video

Submitted By:
S M Ragib Rezwan
103172423
2021/06/01 17:05

Tutor:
NAJAM NAZAR

June 1, 2021

https://www.youtube.com/watch?v=GiYJS7RaK4Q

43 Custom Project Video

A link to your custom project video.

Outcome	Weight
Functional Decomposition	♦♦♦◊◊

gave it the same rating as custom project as it is link to that video

Outcome	Weight
Structured Coding Principles	♦♦◊◊◊

gave it the same rating as custom project as it is link to that video

Outcome	Weight
Programming	♦♦♦♦◊

gave it the same rating as custom project as it is link to that video

Outcome	Weight
Apply Reading Techniques	♦♦♦♦♦

gave it the same rating as custom project as it is link to that video

Swinburne University of Technology

2021 Sem 1 Introduction to Programming

Doubtfire Submission

Custom Project Video

Submitted By:
S M Ragib Rezwan
103172423
2021/06/02 13:11

Tutor:
NAJAM NAZAR

June 2, 2021

https://www.youtube.com/watch?v=QAzsxbK_PEc