

## CS ALL Projects – Project Two: Virtual Robot Bargain Hunt (VRBH) Individual Project Portfolio (IPP)

<b>Student Name: Asis Rai</b>	<b>SID: 6528683</b>
-------------------------------	---------------------

PWP – the link to your up-to-date Project Work Portfolio (PWP) is <https://raia10coventryacuk.wordpress.com/>

Link to my GitHub is <https://github.com/GroupD5Cov/D5VirtualRobotProject>

### Project Work Portfolio (PWP)

#### Week 1 - 18-22 January

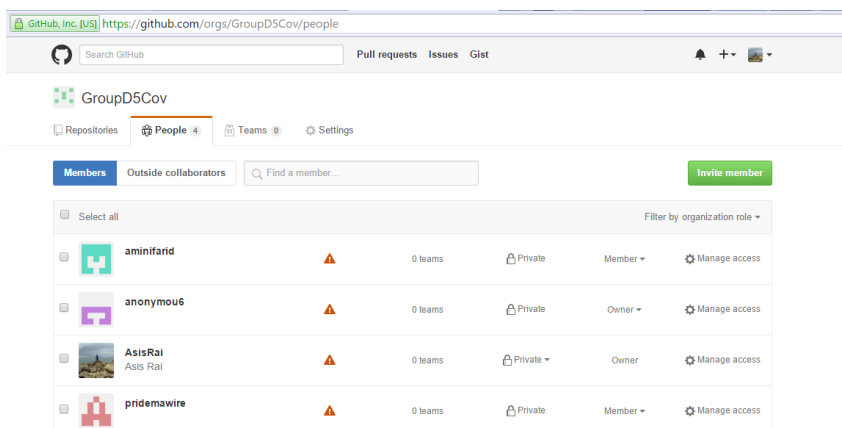
This was the first week and I think the least productive week for all of us. Our project tutor introduced the project two to us: Virtual Robot Bargain Hunt (VRBH).

#### Group work requirements: (contributed to 75% of project marks)

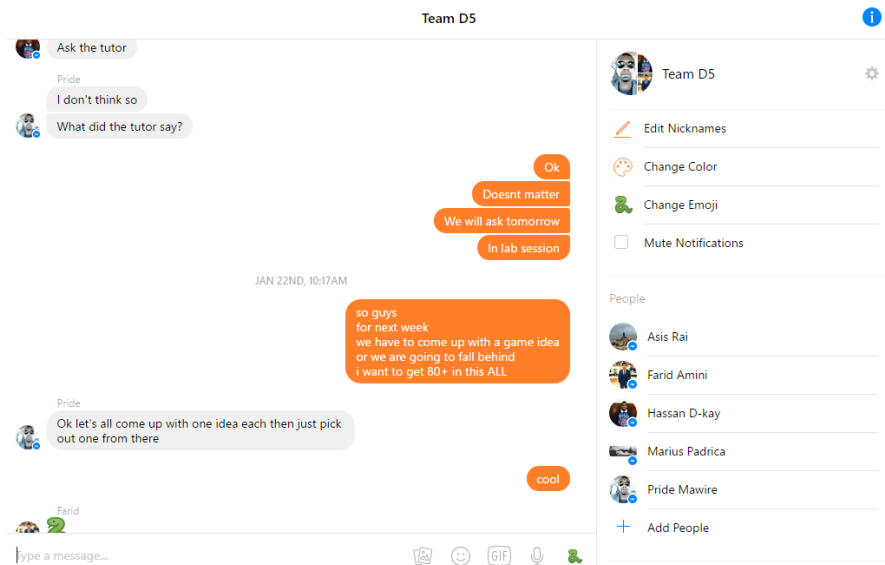
Your team is required to develop a program with a graphical user interface which enables virtual robots to search for bargains. The basic elements of the program are required as follows: (contributed to 70% of project marks)

- To set up different locations with items, each of which should have a name, type (e.g., clothing, jewellery), price and quantity
  - To enable a list of items to be specified by the user for a robot to search for bargains
  - To enable a location specified by the user for a robot to start the searching for a list of requested items
  - To display the bargain items (i.e. cheapest) requested by the user, along with their locations, on the screen
  - To allow a robot to gather as many bargain items requested by the user as possible within the given time specified by the user
  - To display a robot's movement through different locations when gathering the bargain items
  - To allow a robot to sort the bargain items gathered in an ascending or descending order specified by the user according to certain criteria, such as name or price, determined by the user.
  - To display the sorted items on the screen
- Individual work with extra features: (contributed to 25% of project marks) Each team member may produce extra features, in addition to the above required for the group work.

After we were told to go and meet our respective team members. I found out that I was allocated to Group D5 and my group members were – Pride Mawire, Hassan Hersi and Marius Padrica. After meeting with the group, we discussed for an hour of the lesson brainstorming ideas and sharing our experiences from our last project. We then decided to create a GitHub group so that we can all manage the program developed through GitHub and also work on the codes collaboratively.



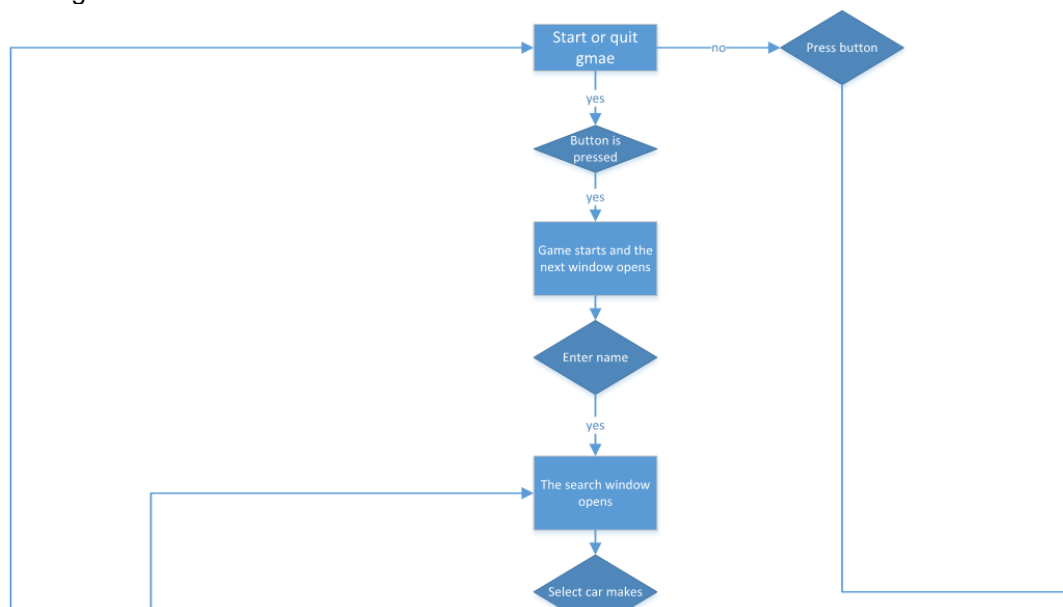
After everyone was connected in the GitHub Group we decided to connect on Facebook as well, so that we could all communicate efficiently.

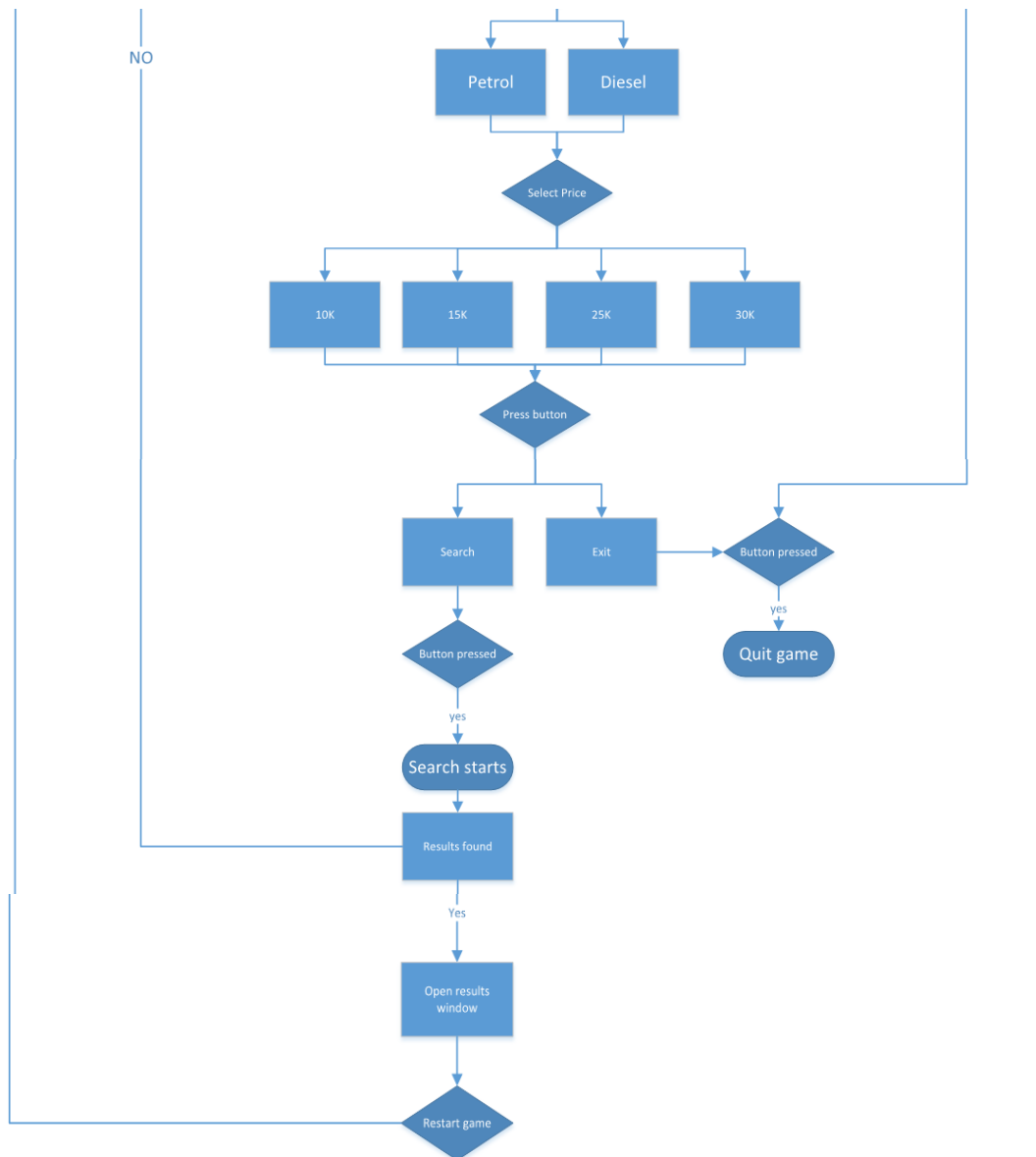


All of us were not sure what we were doing in the first week because this project was new to some group members and they stated that they needed time to understand the whole project tasks and requirements before they could contribute anything to the group and therefore we agreed that in week one all of us would learn the Python GUI (Graphical User Interface) basic features provided in the Tkinter package, so that we would come up with 4-5 ideas and then we could pick out the best possible one which contribute the highest marks. We decided to work on the programming tasks in the GUI tutorials individually however if anyone needed help then we would communicate on Facebook and help each other.

### Week 2 (25-29 January)

In week 2 we all met up on Monday 25<sup>th</sup> and decided to hear out the ideas that we had each came up with. All our ideas were good but it was all vague and not very clear to all of us so we decided to find the similarities in our ideas and pick the best similarity and then come up with a flowchart which will give us a guide line to work to create the prototype graphical interface design for the VRBH program. After we wrote down all of our idea into a paper and picked the best one, we made a flowchart to understand as a group of how the program might work but we decided the flowchart is only to be used as a guideline/prototype and it was not the final idea or flowchart. After we created the flowchart, we as a group sort-of understood what we had to prepare for the coming weeks.





Later in the week when we all met again, we decided to create some design to be implemented into the prototype graphical user interface design. It was decided as a group that Farid would create the design on Photoshop and I would also be involved in the process, I will solely work on the code that would make it work on Python because we thought out of everyone I was the most capable, Pride and Marius will continue work to create the storyboard.

After Farid and I finished creating the design, Pride and Marius worked on the storyboard with all of us helping out explaining and discussing the processes in the storyboard.

## Storyboard

1. This will be the start page as you click on the game. When this window opens you will be able to start the game or exit the game if you wish to. The window will also display the name of the game.



2. The second window will display a bar that will ask the user to enter their name. The user's name will be saved to show the results for the user when they carry out the search on the next window.



3. This window will display the search engine on the left and an image of the world map on the right. This window will be used to carry out the search. The user will be able to search for new or used cars. The user can search for four makes of cars: BMW, Vauxhall, Range Rover and Lexus. Also different fuel types: diesel and petrol. And the price range. This window will also contain two buttons which are to search or exit.



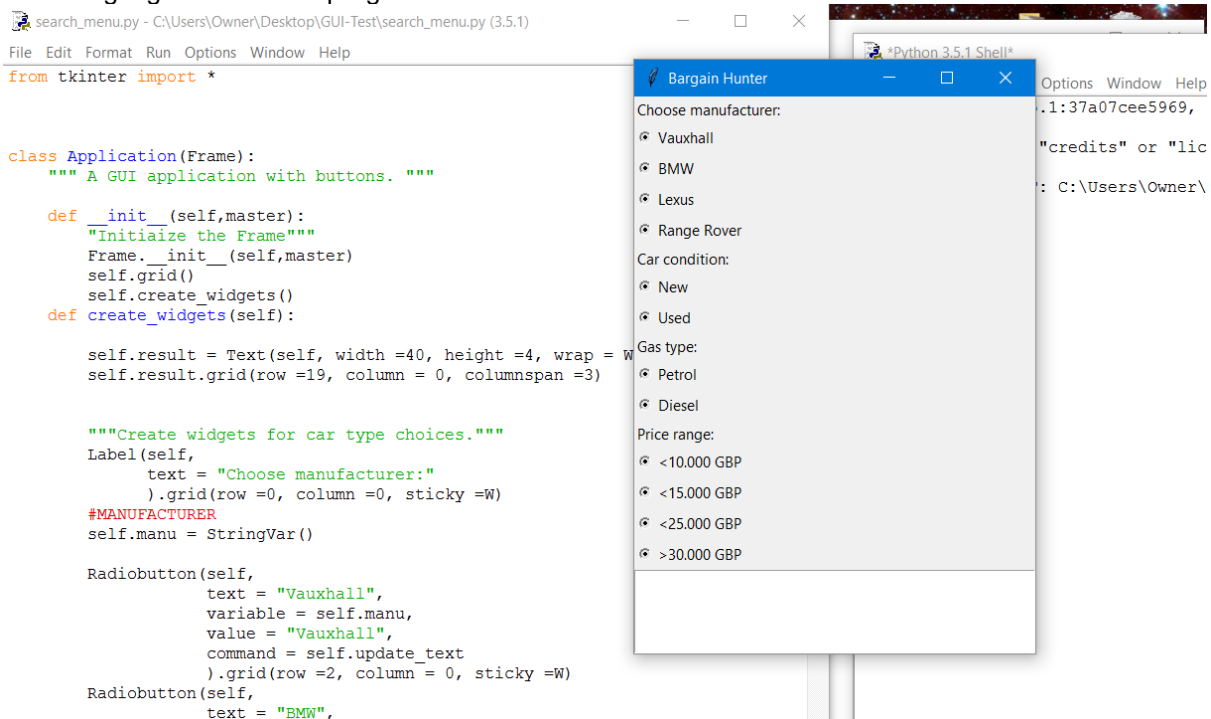
4. The last window will display an image of the world map which will show the robot moving around on the world image to find the location of the item and display the results.



After this I started working on the code where all the images will be implanted into the GUI and in the right places and this would be our prototype graphical user interface design for the VRBH program. We decided to call our VRBH program 'Bargain Inspector'. In this week we also started to prepare our IPP which will be used to asses us after week 8.

### Week 3 (1-5 February)

Week 3 I started to work on the Start menu of our program and I planned to finish the start menu by the end of the week so that we could start working on the GUIs associated tasks to get better understand of Python. We as a group started researching and understanding the basics of searching algorithms so that we can start adding searching algorithms to our programs and decide which will be the most appropriate and relevant searching algorithm for our program.



Later in the week we started to document the program flow in UML activity diagrams which is needed for the IPP file later in the weeks. We also started to prepare for our report of our research and conclusion on searching algorithms. We decided to meet up again later in the week or next week as a team to discuss the situation of the team and to address who is most up-to-date with the GUIs tasks and whoever is the most up-to-date can start working on the searching algorithms and the others can catch-up whilst doing other activities such as updating flowchart, UML activity diagrams and making more pictures in Photoshop for the program etc.

#### Week 4 (8-12 February)

This week we as a group carried on doing GUIs associated tasks and this week was not very productive as most of the group members were missing because they were ill. Therefore, whoever came in for the project session started learning basic sorting algorithms and furthermore carried out a research on a sorting algorithm of our choice. We decided that this week, we were all going to be focused in researching sorting algorithms and learning about it as much as we can so that we understand how to implement them in our program and the missing group members can catch-up when they come in again while we start to implement the sorting algorithms on the code as we would have gone through the algorithms already.

Later in the week we hoped that we would have every single group member in the session, however they had not recovered in the week and therefore again we could decide which sorting algorithm we wanted to implement in the code and therefore we started doing our report of our research on sorting algorithm which would be included in the IPP file.

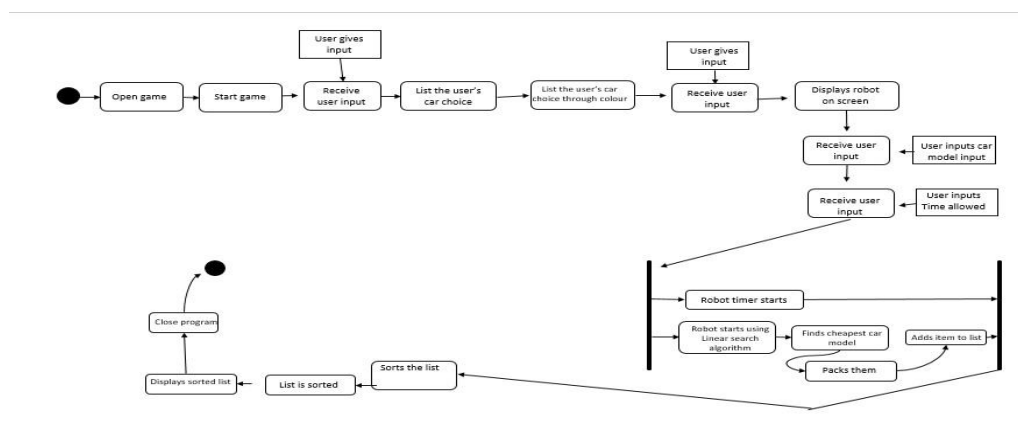
#### Weeks 5 (15-19 February)

This week luckily the whole group was in the session and we managed to tell the missing group members about what they missed in the last session. We all came together and I showed what I had done so far. They were happy with what I had done so far which was create the start screen with radio buttons, however we were falling behind and we decided to tweak our idea so that It would be faster and easy for us to finish our program. This meant flowcharts and storyboard had to be changed as well. We decided to change the idea because some group members were becoming unreliable and if we had stayed with our current idea that meant we each had to be assigned with a specific task for the program to work such as creating a database, searching algorithms, sorting algorithms and we decided to tweak our idea where instead of a database we would use arrays in the program which will make it easier and faster for us to program.

Because I had done a start-up screen before for our initial idea, I was tasked to create a new start-up screen for the program using pygame. Our game idea of choosing car models changed to choosing colours where each car models would be assigned to a specific colour and that colour would be assigned to arrays and then with the help of random numbers the robot would give the car models each random colour (car model) price, name. Through this sorting and searching algorithms would be connected. However in-order for the program to do this the user has to input values into the right from the start screen and then user has to input 4 values, the colour they want(colour assigned to a car model, choose where they want to search from(the map stay the same from previous storyboard and will be displayed on the screen and the user choose when they want to start from using co-ordinates), how many models of the chosen car models the user hope to find and how long the user will allow the robot to search for the values the user just inputted. This meant that we needed a new flow-chart but because this week we had to start our UML diagrams as well and therefore we decided to update our new idea into the UML diagram. I then started to work on the UML diagram and after I will also start working on the start screen and a new storyboard.

#### Week 6 (22-26 February)

This week we were supposed to start testing however our code was not done and therefore we did not do testing this week. By this week I finished the new UML diagram and the new storyboard and started to work on the new start-up screen using pygame.



## New Storyboard



1. First screen would stay the same as before. This will be the first screen when the program starts and two buttons would be added to it, 'Start' button and 'Quit' button. Start button will start the game and take the user to the next screen and Quit game button will close the code and exit the program. Furthermore, instructions will be added to the screen for the user considering one of the principle from 106CR, Designing for usability.



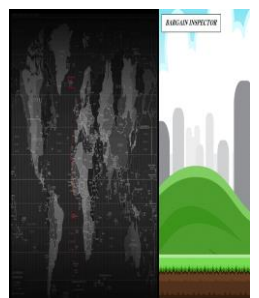
2. This will be the second screen the user will see after the 'Start' button is pressed. On the right hand side of the screen the user will see a text box and a question. The question will say 'Enter colour Red for BMW, colour Blue for Vauxhall, colour Green for Land Rover and colour pink for Lexus. The user will enter the input the colour type in the text box. If the user has entered anything other than the colours allowed, then they will see error message giving the user instruction again to what the user can input and the user may try again.



3. This will be the third screen the user will see and only allowed if the user has inputted the correct colour type from previous screen. In this screen the user will see be able to see the map clearly on the screen and in the right hand side of the screen the user will be able to input the co-ordinates to which side of the map the user wants the robot to search from for the car models. There will be a clear instruction in the right hand side of the screen telling the user what to do in this screen. The screen will move on to the next screen after the user has inputted the co-ordinates in this screen.



4. This will be the forth screen and the user will only be able to move to this screen after the user has inputted the co-ordinates in the previous screen, instructing the robot where to start searching from. This screen will follow the same procedure as the last one, however in this screen the user will be asked to input how many car models the user wants the user to search for and the user has to input the number of car models accordingly. E.g. 1



5. This will be the fifth screen and in this screen the user will be able to input the number of minutes(time) the user would like the robot to search for the car models. After the user has inputting the input the time, all the input has been recorded by the robot and now the robot will start searching for the car models by using the values provided by the user and display the robot's movement in left side of the map. After it has finished searching, the robot will have displayed the results found in the right side of the map with the results and the time it took for the robot to find the results, the results will be displayed in ascending order.



## Week 7 (29 February – 4 March)

This week I was very busy with reiving for my advantage module CISCO which has a lot of content and I was going to be busy for the next coming week as well because that is when I have my CISCO exam. I finished making the user-screen for our program using pygame last weekend and I gave the finished code to the group to work and finish off the program.

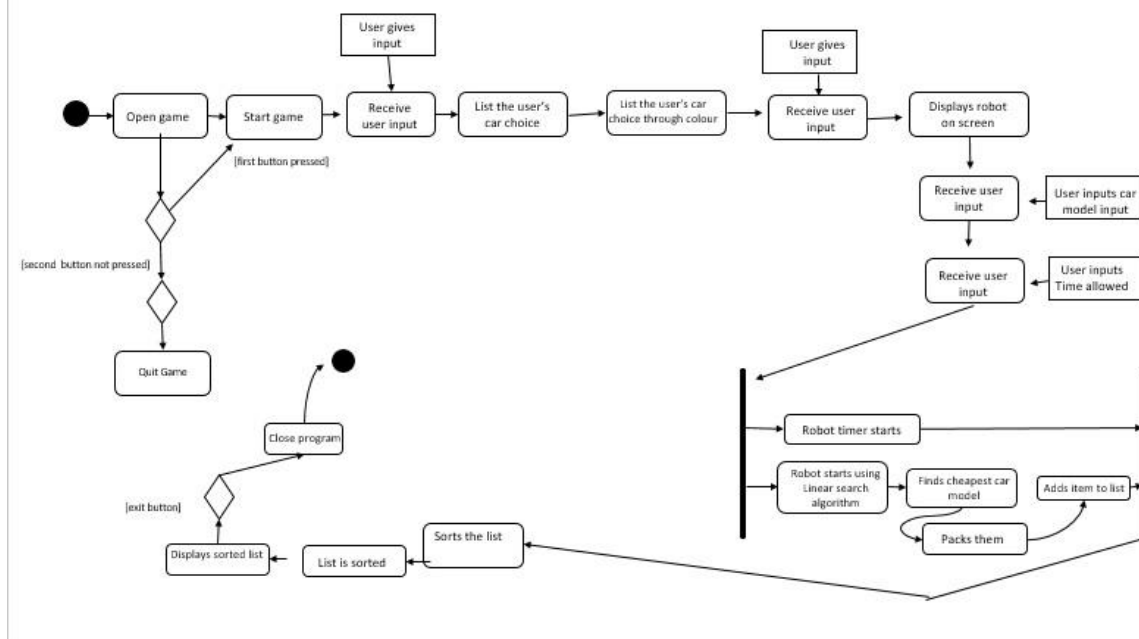


In this week We also had to for the presentation for next week which would require us to do a 10-minute group presentation and discuss with the class:

- What you have achieved as a group
- What problems you have encountered and how you resolved them
- What you have learnt from working as a group (in terms of skills and knowledge)

## Week 8 (7 – 11 March)

This week we delivered our presentation to the class and the tutors, however we did not have a demo to show to the class because we still had not finished the code fully however was  $\frac{3}{4}$  done and just needed a little bit of time. We managed to explain to the class and the tutors what our program is does and the outcome with the help of updated UML activity diagrams and storyboard.





Later in the week we finished the program and was ready for testing. I tested the program and made improvements to the program individually as well as with my group.

Test NO.	Description	Expected Result	Actual Result	feedback	Improvements
1	Start-up Screen	The code is run, and then code should bring a GUI with start-screen and instructions displayed for the user. Buttons to start and quit the game should also appear.	When the code was run the code was compiled and returned with a 'USABLE' GUI with the instructions to guide the user, also the buttons appeared and can be used by following the instructions.	N/A	N/A
2	Play button	Clicking the button should take it to the main screen where now the user will follow the on-screen texts to interact with the game	Button pressed and taken to the main screen and able to enter texts into the texts box	Texts font and colour seems hard to see as it is too small, need to increase the font size or change the font style and font colour.	Texts font changed from aerial to times and size changed from 10-14 and now able to see the text displayed on the screen clearly.
3	User input for car model with colours, where red = BMW, Green = Vauxhall, Blue = Landover and Pink = Lexus	Textbox accepts all four inputs and upper case or lowercase will input does not matter. If Inputs other than allowed entered should generate type error.	Textbox accepted all four inputs and upper case or lowercase will input did not matter. BMW inputted and type error came with a message error.	Change the error message to something appropriate for the first-time users.	Error changed to please enter the colour type matching car model type.
4	User input for co-ordinates	User input the co-ordinates of map to search from. The map displayed on the screen and looking that user can enter co-ordinates to search from a specific country.	Entered 'E' for East and started from east side of the map.	N/A	N/A

5	User input for how many	Users should be able to search for	Users should be able to search for	Feedback received were vague and not	Feedback received were vague and not
	number of car models to find	items for more than 14 car models.	items for more than 14 car models.	enough time to do improvements.	enough time to do improvements.
6	User input for how long the user should wait for the search	No more than 60 minutes should be accepted as it is too large for user and for the code.	180 minutes were accepted as well as the short minutes such as a minute.	Reduce the time allowed to 60 minutes max.	Program made to accept no more than 40 minutes as a group we thought it was more than enough time.
7	Robot's movement visibility	Robot's movement must be displayed on the screen.	Robot's movement were clearly displayed on the screen with white smoke left by the robot behind it.	N/A	N/A

□

After testing everything was done and then we started working on our IPP.

A graphical user interface design prototype (i.e. a storyboards with screen outputs) for the VRBH program with your contribution to the GUI design as an individual clearly identified on the prototype.

Design of the graphical user interface design prototype - a storyboards with screen outputs



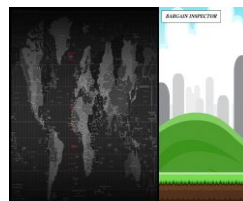
- First screen would stay the same as before. This will be the first screen when the program starts and two buttons would be added to it, 'Start' button and 'Quit' button. Start button will start the game and take the user to the next screen and Quit game button will close the code and exit the program. Furthermore, instructions will be added to the screen for the user considering one of the principle from 106CR, Designing for usability.



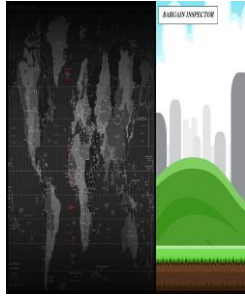
- This will be the second screen the user will see after the 'Start' button is pressed. On the right hand side of the screen the user will see a text box and a question. The question will say 'Enter colour Red for BMW, colour Blue for Vauxhall, colour Green for Land Rover and colour pink for Lexus. The user will enter the input the colour type in the text box. If the user has entered anything other than the colours allowed, then they will see error message giving the user instruction again to what the user can input and the user may try again.



- This will be the third screen the user will see and only allowed if the user has inputted the correct colour type from previous screen. In this screen the user will see be able to see the map clearly on the screen and in the right hand side of the screen the user will be able to input the co-ordinates to which side of the map the user wants the robot to search from for the car models. There will be a clear instruction in the right hand side of the screen telling the user what to do in this screen. The screen will move on to the next screen after the user has inputted the co-ordinates in this screen.



- This will be the forth screen and the user will only be able to move to this screen after the user has inputted the co-ordinates in the previous screen, instructing the robot where to start searching from. This screen will follow the same procedure as the last one, however in this screen the user will be asked to input how many car models the user wants the user to search for and the user has to input the number of car models accordingly. E.g. 1



- This will be the fifth screen and in this screen the user will be able to input the number of minutes(time) the user would like the robot to search for the car models. After the user has inputting the input the time, all the input has been recorded by the robot and now the robot will start searching for the car models by using the values provided by the user and display the robot's movement in left side of the map. After it has finished searching, the robot will have displayed the results found in the right side of the map with the results and the time it took for the robot to find the results, the results will be displayed in ascending order.

## **A report of your research on search algorithms supported with in-text citations/references. (A maximum of 500 words)**

To start off with the research on search algorithms I used the help of 'Python Basics Lab 6' in Computer Science Module. The lab sheet consisted of two search algorithms which are Linear Search and Binary search.

Linear Search is used to find an item in a list which do not have to be in order. [1] This form of search algorithm is one of the most basic search algorithm. When Linear Search is used the program will start at the beginning of the list and then the program will continue searching the list until the search has reached the end of the list of the searched item is found. Where this may be the most basic search algorithm and easy to implement however it could also be very time consuming because if the searched items are not in the list then the search start from beginning and end at the end of the list regardless to generate a result, therefore Linear search algorithm is basic but time consuming.

Binary Search consists of two ways where a programmer can implement Binary search. They are Depth-First search and Breadth-First search. These are algorithms to find connecting nodes in a graph.

Depth-First Search algorithm works by starting at the root also called the starting node, and then following of the branches connected to the root, keep following the branch till you find the desired node or reached the last node of the selected branch. If the desired node is not found, then return to the root and go to the other branch and repeat the last process. [2]

Breadth-First search works by starting at the root node (starting node) and then moving to the second level node and scan from the furthest left node of the second level to the farthest right node of the second level, if the desired node is found then stop, otherwise move on to the third level node and repeat the process.

Researching these algorithms were fascinating and I wanted to extend my knowledge further by researching more search algorithms. While searching on the internet I found out there are many other searching algorithms but the most relevant to the program I was creating was Naïve string search.

[3] Naïve string search can be interpreted graphically sliding a pattern between two strings. The naïve string search simply tests all the possible pattern between texts/string. This will be particularly useful if my program were to search for a match between a given text/string and a text/string in the program. It is a simple and efficient search algorithm.

For our program we used decided to implement Binary search as we thought it would work well with the new tweaked idea of ours and if we managed to do it we thought would help perhaps to achieve a higher mark and good understanding within us.

<sup>1</sup> University, C., 2016. *Python Basics Lab 6*. [Online]

Available at: [https://csmoodle.coventry.ac.uk/pluginfile.php/663997/mod\\_resource/content/2/Python%20Basics%20Lab%206%20-%20Search%20Algorithms%20v2.pdf](https://csmoodle.coventry.ac.uk/pluginfile.php/663997/mod_resource/content/2/Python%20Basics%20Lab%206%20-%20Search%20Algorithms%20v2.pdf)

<sup>2</sup> Cprogramming.com, 2016. *Cprogramming.com*. [Online]

Available at: [http://www.cprogramming.com/discussionarticles/sorting\\_and\\_searching.html](http://www.cprogramming.com/discussionarticles/sorting_and_searching.html)

[Accessed 09 March 2016].

<sup>3</sup> <http://www.personal.kent.edu/>, 2016. <http://www.personal.kent.edu/>. [Online]

Available at: <http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/StringMatch/naiveStringMatch.htm>

[Accessed 07 March 2016].

```
def binarySearch(self, alist, item):
    """This will search a list for the search item and returns all information about that item"""
    first = 0
    last = len(alist)-1
    found = False

    while first<=last and not found:
        midpoint = (first + last)//2
        if alist[midpoint][0] == item:
            return(alist[midpoint])
        else:
            if item < alist[midpoint][0]:
                last = midpoint-1
            else:
                first = midpoint+1

    return found
```

## **A report of your research on sorting algorithms supported with in-text citations/references. (A maximum of 500 words)**

A sorting algorithm is an algorithm that places the elements of a list in a certain order. There are numerous sorting algorithms out there and I have found few of the predominate ones in practice in my research. [4]

Bubble sort sorting algorithm is one of the most basic and simplest to understand. Bubble sort works by bubbling up smallest or largest elements, depending on the sorting order, then it will move to the other second largest element and bubble the elements, it will keep doing this until it reaches the end of the list, then the processes is repeated again until it has sorted out all the elements. Bubble sort is one of the inefficient sorting algorithms and is good if the programmer is working with a small list however it is not recommended for use with big lists as it will consume too much time bubbling elements in the entire list and waste time and productivity.

Insertion sort is another sorting algorithm that I found. I found out that insertion sort is inefficient in large lists like bubble, however easy to implement and more efficient than bubble sort sorting algorithm. Insertion sort takes elements from an unsorted list and inserts them in a new sorted list which is empty in the beginning. Insertion sort is more efficient than bubble sort because the unsorted and sorted lists stay the same and therefore programmers can use the same list to link and represent the sorted and unsorted sections.

Selection sort sorting algorithm is similar to bubble sort however differs by only making one exchange for each pass through the list. Selection sort looks for the largest value as it passes the value and after completing the pass it places the largest value in the correct location where it is placed as sorted. Just like bubble sort, it keeps repeating the process of finding the first largest element and placing it in its proper location and moving into the second largest element and placing the element into its proper place. For this process to continue requires n-1 for passes and sort n items.

Merge sort is more efficient sorting algorithm that subdivides the list into two sub-lists, sorts them and then backup into their respective lists. Merge sort uses a recursive call, when it is made it will divide each lists into sub lists and they will eventually become individual elements and constructed into final sorted lists.

Other sorting algorithms I researched were Heap sort, Quick sort and Shell sort. Heap sort is known for its insertion and merge sorting. Only a constant number of array value are stored outside the input array. This sorting algorithm is not appropriate for the program I am creating. [5]

[6] Quick sort algorithm is similar to Merge sort algorithm, both of them use a recursive call, executed by using divide and conquer strategy, two pointers are used at the end of each element list and then both are compared

<sup>4</sup> University, C., 2016. *Python Basics Lab 7*. [Online]  
Available at: [https://cumoodle.coventry.ac.uk/pluginfile.php/748776/mod\\_resource/content/6/Python%20Basics%20Lab%207%20-%20Sorting%20Algorithms.pdf](https://cumoodle.coventry.ac.uk/pluginfile.php/748776/mod_resource/content/6/Python%20Basics%20Lab%207%20-%20Sorting%20Algorithms.pdf)  
[Accessed 06 March 2016].

<sup>5</sup> <http://staff.ustc.edu.cn/>, 2016. *CHAPTER 7: HEAPSORT*. [Online]  
Available at: <http://staff.ustc.edu.cn/~csl/graduate/algorithms/book6/chap07.htm>  
[Accessed 06 mar 2016].

<sup>6</sup> RSI, 2016. *The Quick Sort*. [Online]  
Available at: <https://interactivepython.org/runestone/static/pythonds/SortSearch/TheQuickSort.html>  
[Accessed 08 March 2016].

with each other, then they are finally compared with each other and picked out the best depending on the sorting order.

Shell sort algorithm is an exchange sort algorithm that works by comparing values at intervals and reducing the size of the interval after each pass through the list. Shell sort is similar to Bubble sort, however faster because the values can move out of place to be put into closer to their destination depending on the sort criteria.

As a group we decided to use bubble sort and quick sort because we knew the lists generated by arrays will be small and these algorithms will be efficient in sorting the lists.


```
def bubbleSort(self, colourL):
    """ This will sort the list in order of price generated by random, cheapest first"""
    for passnum in range(len(colourL)-1,0,-1):
        for i in range(passnum):
            if colourL[i][2]>colourL[i+1][2]:
                temp = colourL[i]
                colourL[i] = colourL[i+1]
                colourL[i+1] = temp

def quick_sort(self, items):
    """ This will sort a list in order searched by binary search"""
    if len(items) > 1:
        pivot_index = len(items) // 2
        smaller_items = []
        larger_items = []

        for i, val in enumerate(items):
            if i != pivot_index:
                if val < items[pivot_index]:
                    smaller_items.append(val)
                else:
                    larger_items.append(val)
```



## Program implementation – GitHub commented code including algorithms (e.g. searching and sorting algorithms) and external libraries/APIs with screen shots of key functionality testing

 AsisRai Added files via upload 99f3e6d an hour ago

1 contributor

516 lines (445 sloc) | 20.2 KB

Raw Blame History

```
1 #importing in functions that we need for the program to work
2 import pygame as pg
3 import random
4 from textbox import TextBox
5
6 pg.init()
7
8 #setting up colour RGB values
9 white = (255,255,255)
10 red = (255,0,0)
11 blue = (0,0,255)
12 green = (0,255,0)
13 black = (0,0,0)
14 pink = (255,20,147)
15 bright_red = (200,0,0)
16 bright_green = (0,200,0)
17
18 bg = pg.image.load('startscreen.png')
19
20 clock = pg.time.Clock()
21
22 font = pg.font.Font(None, 25)
23 frame_count = 0
24 frame_rate = 60
25 start_time = 180
26
27 menuDisplay = pg.display.set_mode((1200,600))
28 gameDisplay = pg.display.set_mode((1200, 600))
29 display_width = 800
30 display_height = 600
31
32 gameExit = False
33
34 KEY_REPEAT_SETTING = (200,70)#textbox to appear in the same position after action
35
36 """setting instruction colour and font"""
37 def instruction(i,Space,List):
38     intrs = List
39     font = pg.font.SysFont("Times", 25)
40     message = intrs[i]
41     rend = font.render(message, True, pg.Color("red"))
42     return (rend, rend.get_rect(topleft=(900,35+Space)))
43
44 """setting font and colour of the name displayed at the start"""
45 def text_objects(text, font):
46     textSurface = font.render(text, True, black)
47     return textSurface, textSurface.get_rect()
48
49 """quit game button function"""
50 def quit_game():
51     pg.quit()
52     quit()
53
54 """start game button function"""
55 def start_game():
56     app = MainProgram()
57     app.main_loop()
58
59 #Right hand side title screen colour
60 def game_intro():
61     x=0
62     y=0
63     i = 0
```

```

64     intro = True
65     gameDisplay.fill(black)
66     while intro:
67         for event in pg.event.get():
68             if event.type == pg.QUIT: #quit function added
69                 pg.quit()
70                 quit()
71
72 #Now creating the start-up screen
73
74     gameDisplay.blit(bg,(x,y)) #displaying in starmenu bg being pygame, x and y being the position
75     largeText = pg.font.Font('freesansbold.ttf',80)#font and font size
76     TextSurf, TextRect = text_objects("Bargain Inspector", largeText)#Program name
77     TextRect.center = ((display_width/2),(display_height/2)) #text alignment
78     gameDisplay.blit(TextSurf, TextRect)
79
80     button("Start",80,450,120,85,white,bright_green,start_game) #Button which starts the program, position,size, colour and linked to t
81     button("Quit Game",605,450,160,85,white,bright_red,quit_game) # Button which closes the program, position,size, colour and linked t
82
83     intrs = ["INSTRUCTIONS:", "Enter colour for car type", "Enter co-ordinates", "Enter value of car models", "Enter time for robot",]#I
84     space = int(150) #position of the instruction on the screen
85
86     while i != 5: #5 total strings
87         prompt = instruction(i,space,intrs)#i=number of instructions, space = position, intrs = intructions
88         gameDisplay.blit(prompt)
89         space = space + 40 #how close to each other the instructions
90         pg.display.update() #for clock speed functuon
91         i = i+1
92
93
94     pg.display.update()
95     #limit the clock speed to 15 FPS (to prevent overflow)
96     clock.tick(15)

```

```

97
98 #buttons function defined, event driven action (for the Start game and Quit button)
99 def button(msg,x,y,w,h,ic,ac,action=None):
100     mouse = pg.mouse.get_pos()
101     click = pg.mouse.get_pressed()
102     print(click)
103     if x+w > mouse[0] > x and y+h > mouse[1] > y: #when mouse button clicked outcome (1,0,0)
104         pg.draw.rect(gameDisplay, ac,(x,y,w,h))
105         if click[0] == 1 and action != None: #if mouse position (0,0,0 = no action, otherwise event driven action)
106             action()
107     else:
108         pg.draw.rect(gameDisplay, ic,(x,y,w,h))
109     smallText = pg.font.SysFont("Times",20)
110     textSurf, textRect = text_objects(msg, smallText)
111     textRect.center = ( (x+(w/2)), (y+(h/2)) )
112     gameDisplay.blit(textSurf, textRect)
113
114 #Initialising the main program with class attribute
115 class MainProgram(object):
116     def __init__(self):
117         """The initialisation function of key components of the main program"""
118         pg.init()
119         pg.display.set_caption("D5's Bargain Inspector")
120         bg = pg.image.load('mainScreen.png')
121         gameDisplay.blit(bg,(0,0))
122         self.red = []
123         self.blue = []
124         self.green = []
125         self.pink = []
126         self.colourA = []
127         self.colour = ""
128         self.num_items = 0
129         self.finishedList = []

```

```

130     self.time = 0
131     self.frame_count = 0
132     self.frame_rate = 60
133     self.start_time = 180
134     self.screen = menuDisplay
135     self.clock = pg.time.Clock()
136     self.robot_loc = []
137     self.fps = 60.0
138     self.done = False
139     self.input = TextBox((900,200,200,40),command=self.get_input,          #setting the size and position of the text box
140                          clear_on_enter=True,inactive_on_enter=False)
141     self.user_input = ""
142     self.color = white
143     self.prompt = self.make_prompt('Enter Red:BMW, Blue:Vauxhall, Green:Land Rover, Pink:Lexus')
144     pg.key.set_repeat(*KEY_REPEAT_SETTING) #textbox to appear in the same position after action
145
146     def make_prompt(self,message):
147         """ Function to create the labels, called everytime a new input is entered """
148         pg.draw.rect(menuDisplay , white,(820,165,400,30)) #1 is left right position, 2 is up down, 3 is width, 4 is height
149         font = pg.font.SysFont("Times", 14)
150         message = Message
151         rend = font.render(message, True, pg.Color("black"))
152         return (rend, rend.get_rect(topleft=(820,165)))#position of the text in the screen
153
154     def event_loop(self):
155         """ A continuous FOR loop which allows an exit for our main program"""
156         for event in pg.event.get():
157             if event.type == pg.QUIT:
158                 self.done = True
159                 self.input.get_event(event)
160
161     def random_types(self):
162         """Randomly generates colours into the screen and randomly gives them a price and a name eg. red-bmw,price """
163         names = ["BMW", "Vauxhall", "Land Rover", "Lexus"]
164
165         names = ["BMW", "Vauxhall", "Land Rover", "Lexus"]
166         for i in range(50):
167             item = random.randint(1,3)
168             radx = random.randint(0,790)
169             rady = random.randint(0,590)
170             radp = random.randint(1,20)
171             radnum = random.randint(1,4) - 1
172             radn = names[radnum]
173             coords = [radx,rady,radp,radn]
174
175             if item == 1:
176                 pg.draw.rect(menuDisplay , red,(radx,rady,10,10))
177                 self.red.append(coords)
178             elif item == 2:
179                 pg.draw.rect(menuDisplay , blue,(radx,rady,10,10))
180                 self.blue.append(coords)
181             elif item == 3:
182                 pg.draw.rect(menuDisplay, green,(radx,rady,10,10))
183                 self.green.append(coords)
184             elif item == 4:
185                 pg.draw.rect(menuDisplay, pink,(radx,rady,10,10))
186                 self.pink.append(coords)
187             i = i +1
188
189     def get_input(self,id,input):
190         """ allows the user to search for cars by enterin a specific colour """
191         try:
192             input = input.lower()
193             self.user_input = input
194             self.colour = input
195             if self.user_input == "red" or self.user_input == "blue" or self.user_input == "green" or self.user_input == "pink":
196                 self.prompt = self.make_prompt('Where do you want to start : e.g. NW')
197                 self.input = TextBox((900,200,200,40),command=self.robot_start, # textbox position
198                                     clear_on_enter=True,inactive_on_enter=False)

```

```

197         if input == "red":
198             for coord in self.red:
199                 x = coord[0]
200                 y = coord[1]
201                 pg.draw.rect(menuDisplay, red,(x,y,15,15))
202                 self.colourA = self.red
203
204         elif input == "blue":
205             for coord in self.blue:
206                 x = coord[0]
207                 y = coord[1]
208                 pg.draw.rect(menuDisplay, blue,(x,y,15,15))
209                 self.colourA = self.blue
210
211         elif input == "green":
212             for coord in self.green:
213                 x = coord[0]
214                 y = coord[1]
215                 pg.draw.rect(menuDisplay, green,(x,y,15,15))
216                 self.colourA = self.green
217
218         elif input == "pink":
219             for coord in self.pink:
220                 x = coord[0]
221                 y = coord[1]
222                 pg.draw.rect(menuDisplay, pink,(x,y,15,15))
223                 self.colourA = self.pink
224
225         else:
226             self.prompt = self.make_prompt('Please enter the colour type given')
227             self.screen = menuDisplay
228
229     except ValueError:

```

```

230         print("ERROR")
231
232     def robot_start(self,id,input):
233         """ Allows the user to choose the starting position of the robot"""
234         input = input.upper()
235         self.robot_loc = input
236         if input == "N":
237             pg.draw.rect(menuDisplay, red,(400,0,20,30))
238             self.robot_loc = [400,0]
239         elif input == "E":
240             pg.draw.rect(menuDisplay, blue,(750,300,20,30))
241             self.robot_loc = [750,300]
242         elif input == "S":
243             pg.draw.rect(menuDisplay, pink,(400,550,20,30))
244             self.robot_loc = [400,550]
245         elif input == "W":
246             pg.draw.rect(menuDisplay, green,(10,300,20,30))
247             self.robot_loc = [10,300]
248         elif input == "NW":
249             pg.draw.rect(menuDisplay, bright_green,(10,10,20,30))
250             self.robot_loc = [10,10]
251         elif input == "NE":
252             pg.draw.rect(menuDisplay, bright_red,(750,10,20,30))
253             self.robot_loc = [750,10]
254         elif input == "SW":
255             pg.draw.rect(menuDisplay, red,(10,550,20,30))
256             self.robot_loc = [10,550]
257         elif input == "SE":
258             pg.draw.rect(menuDisplay, pink,(750,550,20,30))
259             self.robot_loc = [750,550]
260         else:
261             self.prompt = self.make_prompt('Please enter a valid co-ordinate for the robot to search')
262         if input == "N" or input == "E" or input == "S" or input == "W" or input == "NW" or input == "NE" or input == "SW" or input == "SE":
263             self.prompt = self.make_prompt('Please enter the number of car types you will like to find?')

```

```

264         self.input = TextBox((900,200,200,40),command=self.number_of_items, #textbox position
265                               clear_on_enter=True,inactive_on_enter=False)
266
267     def number_of_items(self,id,input):
268         """ This will allow the user to enter the number of chosen car models they want to find"""
269
270         if input.isdigit() and (int(input) <= len(self.colourA)):
271             self.num_items = int(input)
272             self.prompt = self.make_prompt('Enter the minutes you want the robot to search for?')
273             self.input = TextBox((900,200,200,40),command=self.input_time, #textbox position
274                                   clear_on_enter=True,inactive_on_enter=False)
275
276         else:
277             self.prompt = self.make_prompt('Please enter how many chosen car models to find?')
278
279     def input_time(self,id,input):
280         """ Allows the user to enter the time for the robot to search for car types"""
281
282         if input.isdigit() and int(input) <= 15:
283             self.time = input
284             self.start_time = int(self.time) * 60
285
286         else:
287             self.prompt = self.make_prompt('Please enter a valid time, e.g 1 for 1 minute')
288
289     def collide(self,c1, p1, p2, p3,xORy):
290         """ Tests to see if the next pixals are not white"""
291         locations = [p1,p2,p3]
292         self.Collide = False
293         i = 0
294         if xORy == "X":
295             while i != 3:
296                 colour = menuDisplay.get_at((c1,locations[i])) # gets the colour of the pixal at the coordinates
297
298                 if (c1 >= self.nextX and c1 <= (self.nextX + 15)) and (p1 >= self.nextY and p1 <= (self.nextY + 15)):
299                     i=i+1
300                     continue
301                 elif (colour[0] != 255 or colour[1] != 255 or colour[2] != 255):
302                     self.Collide = True
303                     break
304                 else:
305                     i=i+1
306                     continue
307             elif xORy == "Y":
308                 while i != 3:
309                     colour = menuDisplay.get_at((locations[i],c1))
310                     if (c1 >= self.nextY and c1 <= (self.nextY + 15)) and (p1 >= self.nextX and p1 <= (self.nextX + 1)):
311                         i=i+1
312                         continue
313                     elif (colour[0] != 255 or colour[1] != 255 or colour[2] != 255):
314                         self.Collide = True
315                         break
316                     else:
317                         i=i+1
318                         continue
319
320     def bubbleSort(self,colourL):
321         """ Used to sort the list in order of price, cheapest first"""
322         for passnum in range(len(colourL)-1,0,-1):
323             for i in range(passnum):
324                 if colourL[i][2]>colourL[i+1][2]:
325                     temp = colourL[i]
326                     colourL[i] = colourL[i+1]
327                     colourL[i+1] = temp
328
329     def binarySearch(self, alist, item):
330         """Used to search a list for the search item and returns all infomation about that item"""

```

```

330     first = 0
331     last = len(alist)-1
332     found = False
333
334     while first<=last and not found:
335         midpoint = (first + last)//2
336         if alist[midpoint][0] == item:
337             return(alist[midpoint])
338         else:
339             if item < alist[midpoint][0]:
340                 last = midpoint-1
341             else:
342                 first = midpoint+1
343
344     return found
345
346 def quick_sort(self,items):
347     """ Used to sort a list in order by x coords for binary search"""
348     if len(items) > 1:
349         pivot_index = len(items) // 2
350         smaller_items = []
351         larger_items = []
352
353         for i, val in enumerate(items):
354             if i != pivot_index:
355                 if val < items[pivot_index]:
356                     smaller_items.append(val)
357                 else:
358                     larger_items.append(val)
359
360
361         self.quick_sort(smaller_items)
362         self.quick_sort(larger_items)
363         items[:] = smaller_items + [items[pivot_index]] + larger_items
364
365 def robot_move(self):
366     """Makes the robot move visually and makes a countdown timer that countdowns from the users input"""
367     i = 0
368     if self.colour == "red":
369         self.bubbleSort(self.red)
370         locations = self.red
371     elif self.colour == "blue":
372         self.bubbleSort(self.blue)
373         locations = self.blue
374     elif self.colour == "green":
375         self.bubbleSort(self.green)
376         locations = self.green
377     elif self.colour == "pink":
378         self.bubbleSort(self.pink)
379         locations = self.pink
380     #pg.draw.rect(menuDisplay, white,(self.robot_loc[0],self.robot_loc[1],20,30))
381     print(locations)
382     while i != self.num_items : #Makes the robot move visually
383         self.event_loop()
384         nextX = locations[i][0]
385         nextY = locations[i][1]
386
387         if self.robot_loc[0] == nextX and self.robot_loc[1] == nextY:
388             pg.draw.rect(menuDisplay, black,(nextX,nextY ,15,15))
389             self.finishedList.append(locations[i][0])
390             i = i + 1
391
392         elif self.robot_loc[0] < nextX:
393             pg.draw.rect(menuDisplay, white,(self.robot_loc[0],self.robot_loc[1],20,30))
394             self.robot_loc[0] = self.robot_loc[0] + 1
395             pg.draw.rect(menuDisplay, pink,(self.robot_loc[0],self.robot_loc[1],20,30))
396             self.input.draw(self.screen)

```



```

397
398     elif self.robot_loc[1] < nextY:
399         pg.draw.rect(menuDisplay, white, (self.robot_loc[0], self.robot_loc[1], 20, 30))
400         self.robot_loc[1] = self.robot_loc[1] + 1
401         pg.draw.rect(menuDisplay, pink, (self.robot_loc[0], self.robot_loc[1], 20, 30))
402         self.input.draw(self.screen)
403
404     elif self.robot_loc[0] > nextX:
405         pg.draw.rect(menuDisplay, white, (self.robot_loc[0], self.robot_loc[1], 20, 30))
406         self.robot_loc[0] = self.robot_loc[0] - 1
407         pg.draw.rect(menuDisplay, pink, (self.robot_loc[0], self.robot_loc[1], 20, 30))
408         self.input.draw(self.screen)
409
410     elif self.robot_loc[1] > nextY:
411         pg.draw.rect(menuDisplay, white, (self.robot_loc[0], self.robot_loc[1], 20, 30))
412         self.robot_loc[1] = self.robot_loc[1] - 1
413         pg.draw.rect(menuDisplay, pink, (self.robot_loc[0], self.robot_loc[1], 20, 30))
414         self.input.draw(self.screen)
415
416     self.event_loop()
417     # Starts the timer countdown
418     pg.draw.rect(menuDisplay, green, (810, 540, 400, 60))
419     total_seconds = self.frame_count // self.frame_rate
420     total_seconds = self.start_time - (self.frame_count // self.frame_rate)
421     if total_seconds < 0:
422         total_seconds = 0
423     minutes = total_seconds // 60
424     seconds = total_seconds % 60
425     output_string = "Time left: {0:02}:{1:02}".format(minutes, seconds)
426     text = font.render(output_string, True, black)
427     menuDisplay.blit(text, [810, 540])
428     if output_string == "Time left: 00:00":
429         self.done = True
430
431     self.frame_count += 1
432     clock.tick(frame_rate)
433     pg.display.flip()
434
435     self.input.draw(self.screen)
436     self.screen.blit(*self.prompt)
437
438     pg.display.update()
439
440     if self.colour == "red":
441         self.quick_sort(self.red)
442     elif self.colour == "blue":
443         self.quick_sort(self.blue)
444     elif self.colour == "green":
445         self.quick_sort(self.green)
446     elif self.colour == "pink":
447         self.quick_sort(self.pink)
448
449     self.clock.tick(self.fps)
450     if self.time != 0:
451         self.done = True
452
453     def output_lists(self, i, Space):
454         """Displays the list of cheapest items picked up"""
455         if self.colour == "red":
456             output = self.binarySearch(self.red, self.finishedList[i])
457         elif self.colour == "blue":
458             output = self.binarySearch(self.blue, self.finishedList[i])

```

```

458         elif self.colour == "green":
459             output = self.binarySearch(self.green, self.finishedList[i])
460         elif self.colour == "pink":
461             output = self.binarySearch(self.pink, self.finishedList[i])
462
463         font = pg.font.SysFont("Times", 20)
464         message = str(output[3]) + " | " + str(output[2])
465         rend = font.render(message, True, pg.Color("black"))
466         return (rend, rend.get_rect(topleft=(820,35+Space)))
467
468     def main_loop(self):
469         """ Makes the program loops and call certain function only if an event has been met"""
470         i = 0
471
472
473
474         """adds sound to the code"""
475         pg.mixer.music.load('programsound.wav')
476         pg.mixer.music.play(-1)
477
478         space = 0
479         self.random_types()
480         while not self.done:
481             self.event_loop()
482             self.input.update()
483             self.input.draw(self.screen)
484             self.screen.blit(*self.prompt)
485
486             pg.display.update()
487             self.clock.tick(self.fps)
488             if self.time != 0:
489                 self.done = True
490             self.done = False

```


  

```

491         if self.time != 0:
492             self.robot_move()
493             pg.draw.rect(menuDisplay , green,(810,0,450,540))
494             pg.display.update()
495             while i != self.num_items:
496                 self.prompt = self.output_lists(i,space)
497                 self.screen.blit(*self.prompt)
498                 space = space + 20
499                 pg.display.update()
500                 i = i+1
501                 self.done = False
502                 #self.main_program()
503             while not self.done:
504                 self.event_loop()
505
506         #Sets up the start-up screen
507         menuDisplay.fill(white)
508         pg.draw.rect(menuDisplay , black,(800,0,10,600))
509         #Calls mainprogram function to start the game
510         game_intro()
511
512
513         pg.display.update()
514         pg.quit()
515         quit()

```

## One additional feature implementation on GitHub (individual work)

 AsisRai individual work 9ef26d9 10 seconds ago

1 contributor

516 lines (445 sloc) | 20.2 KB

Raw Blame History

```
1 #importing in functions that we need for the program to work
2 import pygame as pg
3 import random
4 from textbox import TextBox
5
6 pg.init()
7
8 #setting up colour RGB values
9 white = (255,255,255)
10 red = (255,0,0)
11 blue = (0,0,255)
12 green = (0,255,0)
13 black = (0,0,0)
14 pink = (255,20,147)
15 bright_red = (200,0,0)
16 bright_green = (0,200,0)
17
18 bg = pg.image.load('startscreen.png')
19
20 clock = pg.time.Clock()
21
22 font = pg.font.Font(None, 25)
23 frame_count = 0
24 frame_rate = 60
25 start_time = 180
26
27 menuDisplay = pg.display.set_mode((1200,600))
28 gameDisplay = pg.display.set_mode((1200, 600))
29 display_width = 800
30
31
32
33
34 display_height = 600
35
36
37 gameExit = False
38
39
40 KEY_REPEAT_SETTING = (200,70)#textbox to appear in the same position after action
41
42
43 """setting instruction colour and font"""
44 def instruction(i,Space,List):
45     intrs = List
46     font = pg.font.SysFont("Times", 25)
47     message = intrs[i]
48     rend = font.render(message, True, pg.Color("red"))
49     return (rend, rend.get_rect(topleft=(900,35+Space)))
50
51
52 """setting font and colour of the name displayed at the start"""
53 def text_objects(text, font):
54     textSurface = font.render(text, True, black)
55     return textSurface, textSurface.get_rect()
56
57
58 """quit game button function"""
59 def quit_game():
60     pg.quit()
61     quit()
62
63
64 """start game button function"""
65 def start_game():
66     app = MainProgram()
67     app.main_loop()
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584

```

```

63     i = 0
64     intro = True
65     gameDisplay.fill(black)
66     while intro:
67         for event in pg.event.get():
68             if event.type == pg.QUIT: #quit function added
69                 pg.quit()
70                 quit()
71
72     #Now creating the start-up screen
73
74     gameDisplay.blit(bg,(x,y)) #displaying in starmenu bg being pygame, x and y being the position
75     largeText = pg.font.Font('freesansbold.ttf',80)#font and font size
76     TextSurf, TextRect = text_objects("Bargain Inspector", largeText)#Program name
77     TextRect.center = ((display_width/2),(display_height/2)) #text alignment
78     gameDisplay.blit(TextSurf, TextRect)
79
80     button("Start",80,450,120,85,white,bright_green,start_game) #Button which starts the program, position,size, colour and linked to t
81     button("Quit Game",605,450,160,85,white,bright_red,quit_game) # Button which closes the program, position,size, colour and linked t
82
83     intrs = ["INSTRUCTIONS:", "Enter colour for car type", "Enter co-ordinates", "Enter value of car models", "Enter time for robot",]#I
84     space = int(150) #position of the instruction on the screen
85
86     while i != 5: #5 total strings
87         prompt = instruction(i,space,intrs)#i=number of instructions, space = position, intrs = intructions
88         gameDisplay.blit(*prompt)
89         space = space + 40 #how close to each other the instructions
90         pg.display.update() #for clock speed functuon
91         i = i+1
92
93
94     pg.display.update()
95     #limit the clock speed to 15 FPS (to prevent overflow)
96     clock.tick(15)
97
98     #buttons function defined, event driven action (for the Start game and Quit button)
99     def button(msg,x,y,w,h,ic,ac,action=None):
100         mouse = pg.mouse.get_pos()
101         click = pg.mouse.get_pressed()
102         print(click)
103         if x+w > mouse[0] > x and y+h > mouse[1] > y: #when mouse button clicked outcome (1,0,0)
104             pg.draw.rect(gameDisplay, ac,(x,y,w,h))
105             if click[0] == 1 and action != None: #if mouse position (0,0,0 = no action, otherwise event driven action)
106                 action()
107         else:
108             pg.draw.rect(gameDisplay, ic,(x,y,w,h))
109         smallText = pg.font.SysFont("Times",20)
110         textSurf, textRect = text_objects(msg, smallText)
111         textRect.center = ( (x+(w/2)), (y+(h/2)) )
112         gameDisplay.blit(textSurf, textRect)
113

```



Lightshot

Your screenshot is copy

**PWP – the link to your up-to-date Project Work Portfolio (PWP).**

<https://raia10coventryacuk.wordpress.com/>

## Program testing and results

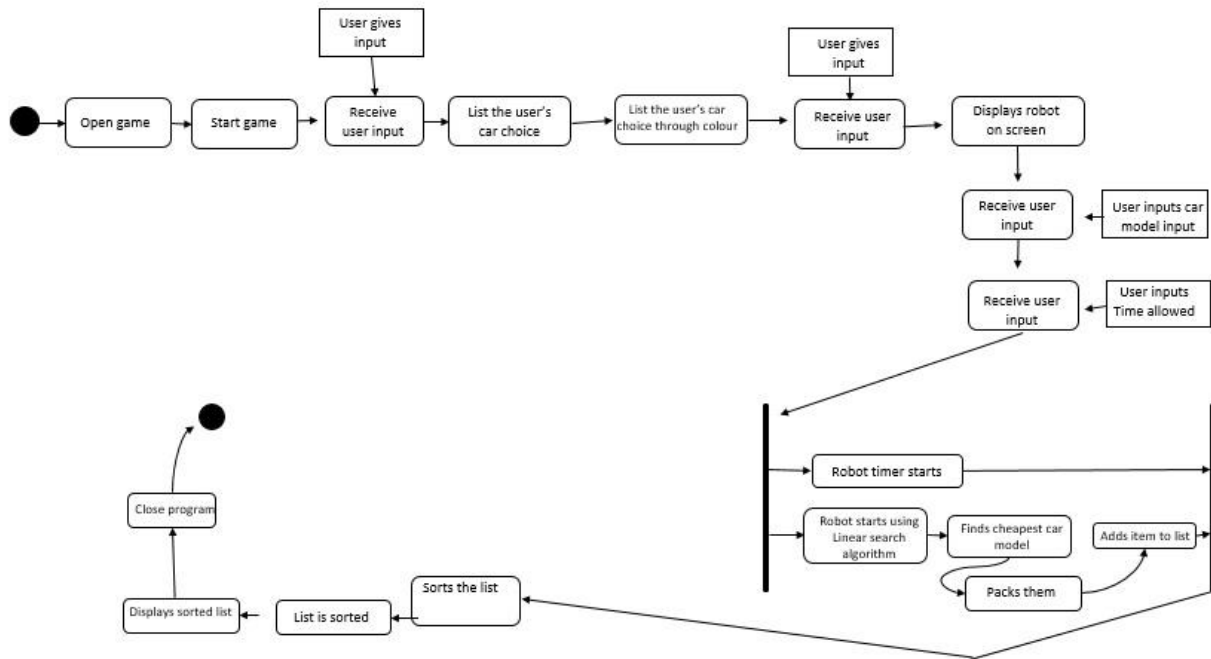
Test NO.	Description	Expected Result	Actual Result	feedback	Improvements
1	Start-up Screen	The code is run, and then code should bring a GUI with start-screen and instructions displayed for the user. Buttons to start and quit the game should also appear.	When the code was run the code was compiled and returned with a 'USABLE' GUI with the instructions to guide the user, also the buttons appeared and can be used by following the instructions.	N/A	N/A
2	Play button	Clicking the button should take it to the main screen where now the user will follow the on-screen texts to interact with the game	Button pressed and taken to the main screen and able to enter texts into the texts box	Texts font and colour seems hard to see as it is too small, need to increase the font size or change the font style and font colour.	Texts font changed from aerial to times and size changed from 10-14 and now able to see the text displayed on the screen clearly.
3	User input for car model with colours, where red = BMW, Green = Vauxhall, Blue = Landover and Pink = Lexus	Textbox accepts all four inputs and upper case or lowercase will input does not matter. If inputs other than allowed entered should generate type error.	Textbox accepted all four inputs and upper case or lowercase will input did not matter. BMW inputted and type error came with a message error.	Change the error message to something appropriate for the first-time users.	Error changed to please enter the colour type matching car model type.
4	User input for co-ordinates	User input the co-ordinates of map to search from. The map displayed on the screen and looking that user can enter co-ordinates to search from a specific country.	Entered 'E' for East and started from east side of the map.	N/A	N/A

5	User input for how many	Users should be able to search for	Users should be able to search for	Feedback received were vague and not	Feedback received were vague and not
	number of car models to find	items for more than 14 car models.	items for more than 14 car models.	enough time to do improvements.	enough time to do improvements.
6	User input for how long the user should wait for the search	No more than 60 minutes should be accepted as it is too large for user and for the code.	180 minutes were accepted as well as the short minutes such as a minute.	Reduce the time allowed to 60 minutes max.	Program made to accept no more than 40 minutes as a group we thought it was more than enough time.
7	Robot's movement visibility	Robot's movement must be displayed on the screen.	Robot's movement were clearly displayed on the screen with white smoke left by the robot behind it.	N/A	N/A



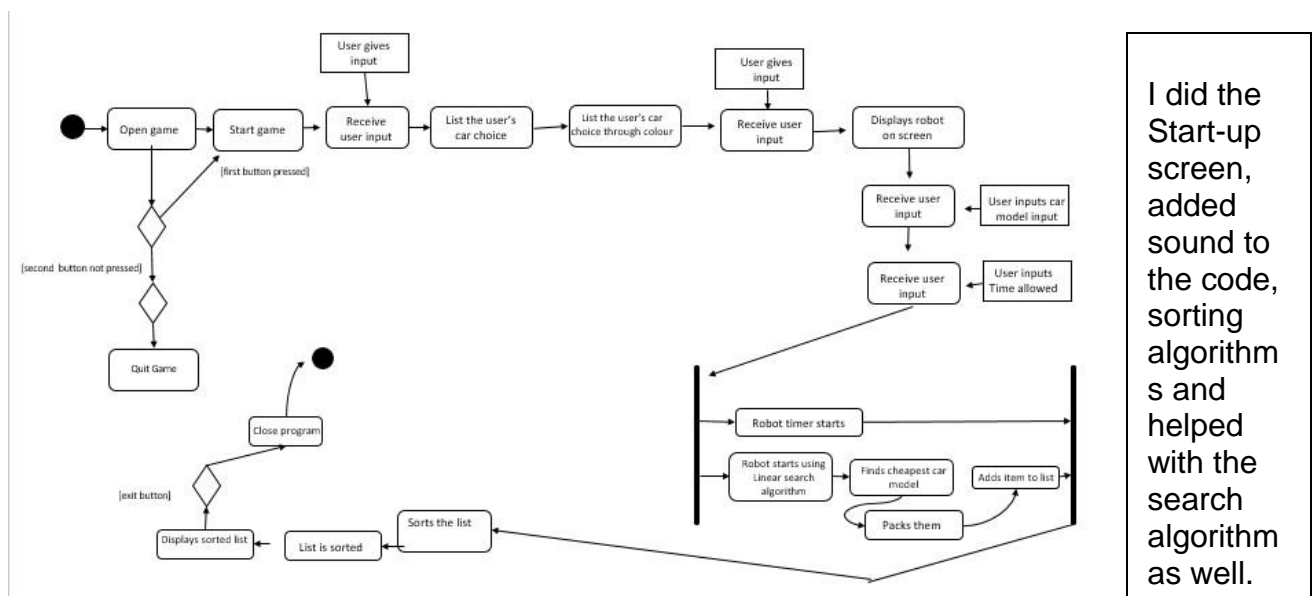


**UML activity diagrams for the program flow with your contribution to the design as an individual clearly identified on the diagrams**



I did the Start-up screen, added sound to the code, sorting algorithms and helped with the search algorithm as well.

**Updated UML activity diagrams for the program flow with your contribution to the design as an individual clearly identified on the diagrams**



I did the Start-up screen, added sound to the code, sorting algorithms and helped with the search algorithm as well.

### A report of what design concepts and principles that have been considered and applied to the prototype design (a maximum of 500 words)

I have learned the design concepts and principles from 106CR and I have successfully implemented in the VRBH program. I found these design concepts and principles very important creating the program as I found out that without the right design concepts and principles, the program could turn out to be very unorthodox for the end user using the program and then this could prove to be very problematic.

From 106CR I have learned 12 important design concepts and principles by Beynon, Turner & Turner in 'Designing Interactive Systems, which I think is ideal to take into consideration when making a program for the end user because these design concepts and principles provide important guideline for good human interaction system design.

One of the principle I considered and implemented was visibility. This principle is concerned with how the data is displayed on the screen to the end user. This principle explains that the best way to data is in graphs or diagram and should be accessible to everyone. I have used this principle by making the text appear in small square and cube boxes which is coloured and should make the user feel very familiar with the program and should make the user feel ready and easy to interpret manner.

Second principle I thought must be considered and implemented was consistency. This principle made sure I kept things the same throughout such as using the same font, same text boxes, same shape buttons into same alignment. This will give the end users indications to which one is button and which one is a text box in the program and helps them to anticipate the next screen or next action.

Another principle I implemented was affordance, this meant that I had to design things to make it easy for the end user to know what each things displayed on the screen was for, for this when creating the storyboard and on the real program, I thought about using radio buttons to make the user to select only one option and using text boxes which made user know that they had to type values or texts into the boxes perhaps by using their past experiences.

Finally, another principle I learned from 106CR was navigation. Using this principle meant that I had to provide support for the end user to move around the system and indicate where else the end user can go to the system, for this I created a navigation in each screen such as in screen 1 there will be buttons with texts inside them to indicate what those buttons do when pressed and give indication to the end user to go further then system by clicking the 'Start' button or exit the program by clicking 'Exit' button. Similarly in screen 3, the end user is able to select only the options that are available and I made sure the option were fit to the end user and what the end user wanted by using other principles such as familiarity, in screen 3 the end user selects the values that the end user want to proceeded with, searched and sorted, to proceed with all of this the user has to proceed by clicking the 'Proceed' button which is colour coded with colour 'Green' which is a universal colour used for 'Go' or 'Safe'.

These were only few of many principles I learned in 106CR and implemented into the program, however they are all equally nevertheless. I believe these design concepts and principles have ensured that I have been guided me in the right steps and has helped me to evaluate and critique prototype design ideas which has made the program 'USABLE'.

## **Program usability testing – a report (a maximum of two pages) on the evaluation of the program using one of the usability methods (such as heuristic analysis, usability tests and cognitive walkthroughs)**

I chose to do testing because a program or a programmer can only get better by testing and practicing. Testing can reveal many things that a programmer may have missed in the code or errors that are hard to capture such as logic errors and the chance of reducing errors can only be possible by doing testing. Testing can be done to reveal more things other than a program functionality, such as to find out if the users can complete the program which will help reveal the difficulty of the program and therefore can be helpful to programmer to dictate whether the code should be more or less difficult.

I have chosen to do usability testing on participants using the program we created as a group. Usability testing reveals the problems real users experience with your product or web site under the actual conditions of use. We ask participants to carry out realistic tasks and note where they experience problems. [7] We think we have completed all the testing necessary on the code and implemented all the functionality. The test is being done because this test will allow me to get accurate data from real life scenarios by simply observing how a user or a group of users will use our program, how difficult they will find it and if the code generates any errors even though it has been tested beforehand.

During the usability tests I think I observed were effectiveness, efficiency, satisfaction of the user and the learnability because I believe that these are key factors when designing a program usability. I planned to carry out my usability testing was by having people carry out tasks on the program which were considered important by me and then I observed them and saw how they did it. I told them to think out loud during the so that if they encounter an error then I would fix it on the spot and tell them to re-do the test again to see if the problem has been fixed.

I chose to pick 5 people at the ECG building, Coventry University to do the test. I laid out some important tasks in they can perform on the code and watched them do it. I had to make sure I was not making them test to link, load or code testing, I could not ask friends/family, online surveys or focus groups to do the test. I was hopeful by the end of the code I would know the effectiveness, efficiency, satisfaction and learnability of the program.

First task I laid out for my participants was that I loaded up the start-up screen and told them to start the main interface, which was to start the game by clicking the start button.

Result – This went great as everyone was able to start the program without me telling them what button to press or where to press which told me that my start-up screen was effective, efficient and needed no help which meant that no improvements were needed on the start-up main screen.

Second task I laid out was to follow the on-screen instructions and make the robot to find the cheapest car model of their choice from BMW, Vauxhall, Land Rover or Lexus.

Result – This also went great for 2 participants; however, the rest took more time than the others. The participants that finished the quickest were able to follow the on-screen instructions to find the cheapest car model of their choice which suggested to me that they have some computer knowledge and have a very good understanding of how the instructions were set. They pointed out that one of the things they found amazing was how they were able to see visually where in the map, which continent it was searching for and the county within the continent which helped me understand that they were able to give me a really good feedback and not a vague one. However, for the rest 3 participants it did not go well even though they were able to the main task that I laid out. I noticed it was because some of them were really slow at inputting the values needed and some took their time to read out the instruction fully and then read it again and again. This told me that they were not expert users and had a very little knowledge with computers. They told me that instructions were little bit hard to understand and that's why they kept reading it repeatedly. They also mistook the minutes allowed for hours. This told me that I had to change or word the instructions in a way that perhaps all users can understand and therefore I changed the instructions with giving examples next to it and told the 3 users to do the test again. In the second time I noticed that it took them a minute less to complete the task and told them to describe their experience. They told me that it was really good visually, however could be more fun if it had some kind of music in the background. I added a music to the background and told them to do the test again and this time they finished even early because they had done the test 3 times which relates to one of the principle 'familiarity' and they said the music fit the background really well and made it more interesting. With

---

<sup>7</sup> Userfocus.com, 2016. Userfocus.com. [Online] Available at: [Userfocus.com](http://Userfocus.com) [Accessed 06 march 2016].

the feedback received and changes I have made I can conclude that the program is flow of the program and the program itself is effective, efficient, satisfaction and learnability.

The final task I set them was that after the result is displayed, exit the program.

This went amazingly well for all 5 participants and relates to the metal mode as they were able to find the X button on the right side window which was colour coded to red and they would have been familiar with it.

Overall, I believe the program we created was a success and was able to meet the requirements of visual design and interface design. The program has proved to be very effective as all the participants were able to actually do specific tasks that I set with efficiency. I took their feedback of making the program more fun by adding a music to the background and added the music in the spot and the second prototype proved to be provide satisfaction. I took their advice of learnability, making the instructions clearer and changed it in the spot and second prototype provided to be more successful that the first one as the second program was changed with more clear instructions and tested again, with these changes I can conclude that I can use the second porotype as our final program as it meets all the requirements which are Effectiveness, Efficiency, Satisfaction and Learnability.

## **Bibliography**

- Cprogramming.com, 2016. *Cprogramming.com*. [Online]  
Available at: [http://www.cprogramming.com/discussionarticles/sorting\\_and\\_searching.html](http://www.cprogramming.com/discussionarticles/sorting_and_searching.html)  
[Accessed 09 March 2016].
- <http://staff.ustc.edu.cn/>, 2016. *CHAPTER 7: HEAPSORT*. [Online]  
Available at: <http://staff.ustc.edu.cn/~csli/graduate/algorithms/book6/chap07.htm>  
[Accessed 06 mar 2016].
- <http://www.personal.kent.edu/>, 2016. <http://www.personal.kent.edu/>. [Online]  
Available at:  
<http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/StringMatch/naiveStringMatch.htm>  
[Accessed 07 March 2016].
- RSI, 2016 . *The Quick Sort*. [Online]  
Available at:  
<https://interactivepython.org/runestone/static/pythonds/SortSearch/TheQuickSort.html>  
[Accessed 08 March 2016].
- University, C., 2016. *Python Basics Lab 6*. [Online]  
Available at:  
[https://cumoodle.coventry.ac.uk/pluginfile.php/663997/mod\\_resource/content/2/Python%20Basics%20Lab%206%20-%20Search%20Algorithms%20v2.pdf](https://cumoodle.coventry.ac.uk/pluginfile.php/663997/mod_resource/content/2/Python%20Basics%20Lab%206%20-%20Search%20Algorithms%20v2.pdf)
- University, C., 2016. *Python Basics Lab 7*. [Online]  
Available at:  
[https://cumoodle.coventry.ac.uk/pluginfile.php/748776/mod\\_resource/content/6/Python%20Basics%20Lab%207%20-%20Sorting%20Algorithms.pdf](https://cumoodle.coventry.ac.uk/pluginfile.php/748776/mod_resource/content/6/Python%20Basics%20Lab%207%20-%20Sorting%20Algorithms.pdf)  
[Accessed 06 March 2016].
- Userfocus.com, 2016. *Userfocus.com*. [Online]  
Available at: [Userfocus.com](http://Userfocus.com)  
[Accessed 06 march 2016].