Adventure World Supermarket GUI

Report

Candidate 253219

Overview

The following document reports on the making of the Adventure World Supermarket Graphical User Interface (GUI). It is divided into sections: 1, Changes to the game; 2, Designing the GUI.

Changes to the Game

The overarching goal of the game remains the same, however, in response to the feedback received regarding the lack of complexity of the game on the first part of the assessment, some new features were were implemented.

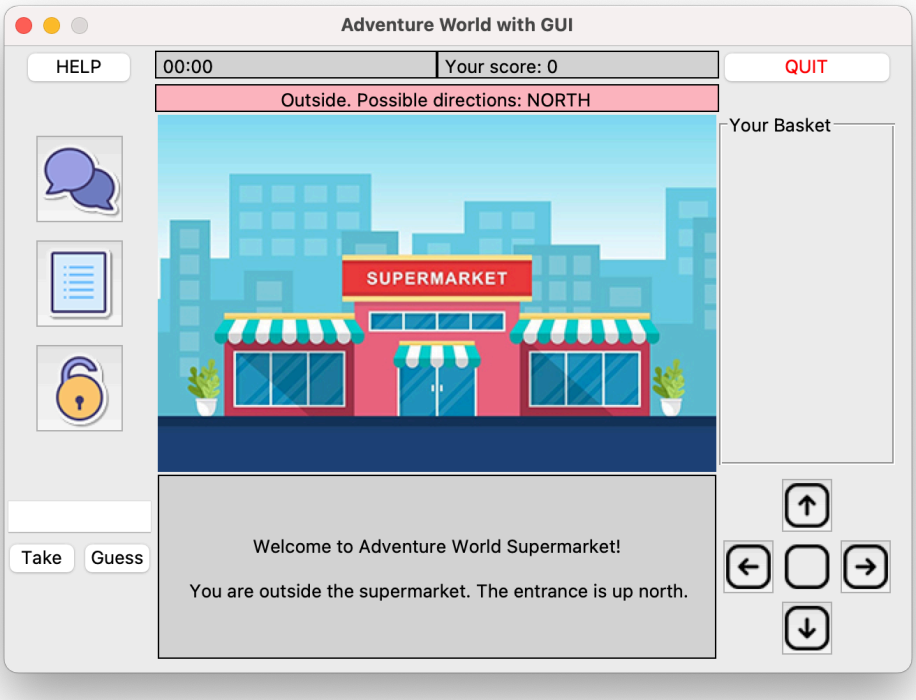
Implementing a score and timer

A points scoring system was implemented, as well as a timer. Each item collected is worth 2 points. If a player attempts to take an item from the wrong aisle, 2 points is deducted from their score. If the bonus item has been guessed, points are doubled at checkout. Fast players who complete the game under 3 minutes further double their points at checkout, whereas slower players who take over 8 minutes to complete the game are penalised by having half of their points take off at checkout.

Implementing an locked room

An extra room, accessible with a key, was also added. They player can collect a key in aisle 3, can then find a lock in aisle 6 which opens the secret aisle. The aisle contains snacks for the player: the player can only choose one snack. Each of these snacks are worth various amounts of points which the player doesn't know. When the player takes a snack, points are added to the total score.

Designing the GUI



General design concept and game adjustments

Adventure World Supermarket with GUI was designed on Mac OS, and therefore requires to be run on this operating system for a faithful representation of its visual components. The first step taken in the GUI development stage was the migration of all `TextUI()` functions into a the new `App()` class. This new class was based on the provided starter code, although many changes were necessary to adapt it to the evolved game, and replaced the need for the previous text interface class.

The GUI for AdventureWorld was constructed using the Tkinter `grid()` model. Visually, it is divided into three main sections: two narrow side columns and a large central one. The left-side section holds the help, speak, list, unlock, take and guess commands. The central section doesn't have any buttons and automatically reconfigures itself as the game is played. It holds the live timer, the score, a visual representation of the various supermarket location, and a text box. The right-side section holds the quit button, a basket area which updates as the game is played, and four direction buttons.

A hybrid text entry/button command was necessary as the game relies on the user's guesses of where the items are, and the riddle's answer too. Therefore, the 'take' and 'guess' commands require the text entry box. As the GUI got more refined, the need for certain commands began to seem redundant. The ability to have various pieces of information clearly displayed in the window removed the need for certain commands that were essential in the text-based version of the game. Hence the look, list, score, time, compare, and basket command buttons were removed.

Furthermore, much of the error prevention required in the previous version of the game became unnecessary. Many of the GUI buttons now provide the second word of the speak and go commands, thus eliminating the risk of erroneous commands being inputted by the user. This is shown in `doSpeak()`, previously reliant on the name of the NPC to work, which now automatically recognises which NPC to talk to depending on the user's location, eliminating the need for user input; and in the `doGoCommand()`, which was given four directional buttons representing north, south, east and west in the GUI.

A simple global logging function was also implemented:

```
def gameLog(message):
    f = open(f"log/log.txt", "a")
    f.write(message + '\n')
    f.close()
```

The .txt file showing a the movements and actions take by the user can be found in the log folder.

Challenges

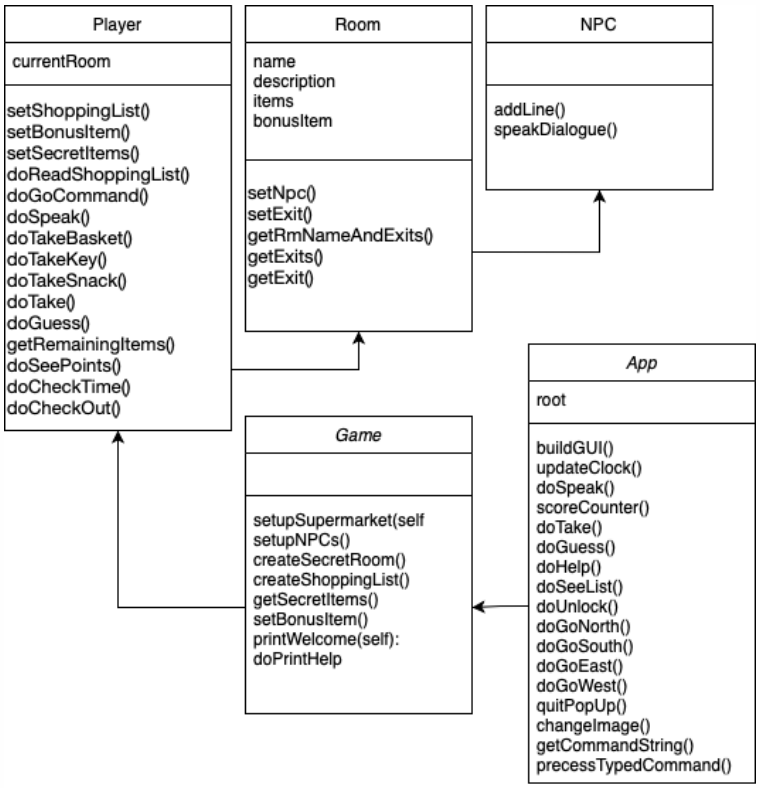
Many iterations of the GUI were made before the final layout was decided upon. There were some initial struggles in placing items in the desired location on the screen, but after some trial and error a suitable layout was created.

A feature which was particularly time-consuming to implement was the live running timer. After unsuccessfully trying a variety of while loops and if statements, the `.after()` function was discovered and made this feature possible:

```
    def updateClock(self):
        self.timerArea.configure(text=" " + self.game.player.doCheckTime())
        if not self.game.player.checkoutExecuted:
            self.timerArea.after(1000, self.updateClock)
```

Perhaps one of the main challenges, however, was the code cleanup required after the GUI implementation. Many functions became unnecessary and could be condensed, and a thorough comb-through of the code had to be done to ensure that there were no redundant lines left over from the text-based version of the game. A visual interface changes the feel of the game and its code structure, and thus the transition from text- to interface-based required many adjustments to maintain code quality.

Class Diagram



Testing

Another aspect of feedback which was improved upon was the lack of automated unit testing on the game.

Game testing

Automated unit tests were implemented for the majority of the functions in `Player()`, the biggest class of the game. The `TestPlayer()` class file can be found in along with the game files.

GUI testing

Manual tests were done to check that all aspects of the GUI were functional, shown below.

Test Number	Description	Expected Result	Actual Result	Pass/Fail
1	Press north button where allowed	Proceed north	Proceeded north	Pass
2	Press north button where	GUI notifies user 'You can't	GUI notifies user 'You can't	Pass

	not allowed	go there'	go there'	
3	Press south button where allowed	Proceed south	Proceeded south	Pass
4	Press south button where not allowed	GUI notifies user 'You can't go there'	GUI notifies user 'You can't go there'	Pass
5	Press east button where allowed	Proceed east	Proceeded east	Pass
6	Press east button where not allowed	GUI notifies user 'You can't go there'	GUI notifies user 'You can't go there'	Pass
7	Press west button where allowed	Proceed west	Proceeded south	Pass
8	Press west button where not allowed	GUI notifies user 'You can't go there'	GUI notifies user 'You can't go there'	Pass
9	Press help button	Displays helpful text	Displayed helpful text	Pass
10	Press speak button in room with no one to speak to	Notifies user no one to talk to here	Notified user no one to talk to here	Pass
11	Press speak button in room with NPC	Prints NPC dialogue	Printed NPC dialogue	Pass
12	Press unlock button before having taken key	Notifies user you can't unlock without key	Notified user you can't unlock without key	Pass
13	Press unlock button after having taken key but in wrong room	Notifies user no doors to unlock here	Notified user no doors to unlock here	Pass
14	Press unlock button in correct location	Unlocks room	Unlocked room	Pass
15	Press take button without inputting text	Notifies user it needs input	Notified user it needs input	Pass
16	Press take button with unrecognised text	Notifies user not sure what they mean	Notified user not sure what they mean	Pass
17	Press guess button without inputting text	Notifies user it needs input	Notified user it needs input	Pass
18	Press guess button with unrecognised text	Notifies user the guess is wrong	Notified user the guess is wrong	Pass
19	Press guess button in the wrong room	Notifies user they can only guess in aisle 2	Notified user they can only guess in aisle 2	Pass
20	Guess correctly	Adds points, adds item to	Added points, added item to	Pass

		basket, notifies user their guess was right	basket, notified user their guess was right	
21	Press list button without having spoken to Lisa	Notifies user they don't have a shopping list yet	Notified user they don't have a shopping list yet	Pass
22	Press list button after having spoken to Lisa	Displays shopping list	Displayed shopping list	Pass
23	Attempted checkout without list or basket	Notifies user they need a basket and a list to checkout	Notified user they need a basket and a list to checkout	Pass
24	Attempted checkout without all items	Alerts user that they still need to collect items before checking out	Alerted user that they still need to collect items before checking out	Pass
25	Checkout with all conditions met	Updates points, stops timer, notifies user of game completion	Updated points, stopped timer, notified user of game completion	Pass
26	Attempted checking out twice	Notifies user they can only checkout once	Notified user they can only checkout once	Pass