Technical Specification Design (TSD)

Dashboard

SUMMARY

The player needs a control panel to access some additional features of the game, reach the main menu (while playing), and also view key data regarding the game status and the current level.

GOALS

The player will be able to use the dashboard to:

* Activate extra driving features (lights, blinkers, wipers, air conditioning, and music player).
* Pause the level and open the main menu.
* See their vehicle's current velocity.
* See the countdown for the current level.
* See the instructions for the current level.
* See their current game score.
* See real-time updates about detected traffic violations and the resulted score reductions.

Future Goals (if there will be enough time):

* Get "phone calls" as an extra challenge, if the player answers they fail.

SOLUTION

The Dashboard feature could combine several different elements to address each of these goals:

|  |  |  |
| --- | --- | --- |
| *Element* | *Purpose* | *Control* |
| Rear View Mirror | Show how much time is left for this level's mission (Countdown). | Non-clickable, display only. Will update automatically with every change. |
| Mobile | Show the current level. |
| Show the player's current Score. |
| Show the latest traffic violation the player performed in this level. |
| Show instructions for this level. |
| \*Show messages and calls. |
| Speedometer | Show the player's velocity. |
| Blinkers | Signal before making a turn with the player's vehicle. | Click using the left mouse button. |
| Lights, Wipers, AC | Earn bonus points based on the scene's background. |
| Music | Switch between available soundtracks. |
| Key Fob | Pause the level.  Open the main menu of the game. |

CODE DESIGN

|  |  |
| --- | --- |
| **DashboardButton**  **Class** | |
| +init(self, image\_off, image\_on, scale, x\_pos, y\_pos) | Initialize the DashboardButton object |
| +button\_pressed(self) | Check if button is pressed then update image and execute this button's function |
| +update\_image(self) | Switch the button's image (ON/OFF) |
| +execute() | Execute this button's function |

**♥** execute() – should we use Command pattern?

|  |  |
| --- | --- |
| **DashboardManager**  **Class** | |
| +init() | Initialize the DashboardButton object |
| +subscribe(event\_type, listener) | Attach a new listener to the dashboard buttons' state |
| +unsubscribe(event\_type, listener) | Detach an existing listener from the dashboard buttons' state |
| +notify(event\_type, data) | Update listeners about changes in buttons' state |

♥ I designed it by Observer pattern but not entirely sure about this, there is also Mediator we can use.

Other functions:

* Main\_menu()
* Draw\_text() – (countdown, level, score, violations, instructions, velocity)
* Change\_music()

TEST PLAN

* Manual tests:
  + Make sure each element is placed where it should be
  + Make sure each button's boundaries are recognized
  + Make sure each button is clickable and the image is updated accordingly between ON/OFF
  + Make sure each button does what it's supposed to
  + Make sure each view-only element shows the right information
  + Make sure each view-only element shows the information the right way
* See if we can make the code better – more efficient and more lightweight.

WORK

|  |  |
| --- | --- |
| Task | Hours |
| Find and create assets | 8 |
| Place all the elements on the screen | 0.5 |
| Define buttons boundaries | 2 |
| Make each button clickable | 3 |
| Decide on design patterns for DashboardManager and view-only elements |  |
| Make each button functional |  |
| Make each view-only element functional |  |