COMP 302

Chewy Lokum Legend - Phase 2

Design Discussion

Open IT

In the first phase of the project, we aimed to design the game as modular as possible so that further changes and extensions would be easily made. In order to build a modular design, we utilised design patterns such as strategy and subcriber/publisher so for each type of swap, match, explosion and generation, we created specialised classes as we call “handlers” in general. Moreover, we listed the names of those handler classes in a XML file within an order to be able to easily adjust the rules to be applied and their priority. Through this design, we do not need to change the existing classes and codes to remove or add new rules to handle new occasions, we just need to add a new class and state the name and the priority in the XML file.

For the second phase of the project, we are asked to bring a new type of level to the game which has a time limit for the player to reach the target score. Additionally, we are asked to introduce a new type of lokum to this type of levels which gives additional time when exploded. In our design discussions, we have decided to obtain this functionality by adding a new class called “TimeLokum” as a subtype of RegularLokum of our design so that each instance of TimeLokum can store the amount of additional time in its own field as well as it can be easily distinguished by new handler, TimeLokumExploder, which is going to be responsible for explosion of TimeLokum. For the new type of level, we extended our ScoreGoal class with another subclass called TimeScoreGoal. This type of goal states the amount of time for the particular level to be completed. As in our design in phase 1, the information about each level, the type, the target score, the number etc., is stored in another XML file. So for TimeScoreLevels, we are going to extend the structure of the XML file a little and adjust the XMLParser accordingly and that is all.

As the second requirement of the second phase, we are asked to bring special swaps. This is another easy task to be done with our design. To fullfil this requirement, we are going to add a few fields and methods to our GameEngine to fetch the allowed number of special swaps for each level and keep the record of them and allow the user to choose non-consecutive lokums and swap them when special swap option is selected. Allowing special swaps is trivial since GameEngine will just ignore the Board class' response to the swappability of two lokums because Board checks if they are consecutive and the swap results in a match.

To conclude, our design for the phase 1 was so well-thought that we will not have to make a change in the structure of the design. We are going to make only some additions and small changes. Furthermore, we are planning to polish the flaws of our implementation for phase 1 by revising the code and make the current version more coherent.