Color Switch

A clone of the original Color Switch game that implements the infinite Classic structure as well as some additional functionalities.

The game requires the user to provide continuous input to the ball to prevent it from falling down to infinity thus ending the game and also to move forward, with the ball jumps, preventing collisions with obstacle parts of color different from that of the ball and, collecting stars and switching colors.

The difficulty of the game is increased with respect to the obstacles and ball jump as the same proceeds and 3 different obstacles have been used by us.

We implemented the game using OOPs concepts and by using JavaFX.

Design Patterns

1)**Facade Design Pattern** used to implement the various cases of choice in the code. Here switch case and if else statements have been used to implement the same.

2)**Strategy Design Pattern** used to initialize the different obstacles for their setup.That is their making, movement and initialisation on the scene.

3)**Factory Design Pattern** has been used to initialise the objects of various obstacles we have used.

4) **Iterator** has been used to traverse through the HashMap of colors that contains information of the colors of our object and ball.

Apart from these, all **OOPs concepts** have been put to use in this project. With special emphasis on **classes** and their functionalities being **coehent** to their classes.

We implemented our **own exception**,NotSufficientStars indicating when stars are lesser than that required to resurrect

We have also implemented **serializable interface** to enable saving the game as per user requirement..

Contribution

Ananya Jain and Khushi Agarwal

Since we belong to different branches and follow different courses,our time distribution was diverse.Hence,we followed the strategy of dividing everything in half and implementing it.

Ananya

GUI: Save game, New Page

1 Obstacle, Ball Linking , coherence and Movements, Logic implementation

Khushi

GUI: Exit Game, Pause Game

2 Obstacle, linking, Collision, Serialisable, Logic Implementation.

Difficulties and Solution

Thought the course of the we faced a number of difficulties,

1)Obstacle positioning and arrival
Solved by hit and trial and timeline

2)Intersection implementation
After a lot a searching we found the intersect handler.

3)Serializable implementation
Understanding the fxml entities that cannot be serialised and linking them with classes to store info.

4)Audio Implementation
Continuous error solving. Changed the VM arguments and looked into which directory to keep the audio in.

BONUS IMPLEMENTATION

1)Night and Day Mode
2)Surprise star value increase
3)Audio for a very pleasant experience.
4)Great emphasis on graphics.

By:

Ananya Jain(2019408)

Khushi Agarwal(2019312)