Chain Reaction

Meet Arora- 2016055
Gagandeep Singh- 2016037

Implementation

- Design Patterns:
  - Singleton - For Grid Class
  - Decorator - During Serialization (ObjectOutputStream)
- Almost all the scenes in the game are made and implemented through scene-builder, except the one the actual game is played on. That is made programmatically and is therefore dynamic.
- Using inheritance to create custom classes in order to get the required functionality more easily.
- Use of Serialization for saving the state of the game and resuming thereafter.
- Using stack to implement the function of “undo”. We have created it such that a player gets in total only 3 chances to redo their previous move in the entire game. It is also possible to undo moves one-after-the-other without the appropriate player playing their chance.
- Making use of randomisation and probability to build a simple opponent AI.

Issues Faced

- ❖ **Changing scenes when making them with scene builder**: We first thought of making a new window and close the previous one each time we needed to change scenes. Then we thought of changing scenes. At the end, we figured out that we need to change the panes as we were only defining panes in scene builder, thus keeping the window constant.
- ❖ **Exiting the error window with keyboard**: We did not find any function which could close a window with a key event. So we added a keyevent to the scene of the window and commended to close the window when a particular key would have been pressed.
- ❖ **Undo**: We faced a lot of problems with the undo functionality due to subtle logical errors. At first, some balls would remain even after undo. Then, at some point, the player would remain the same but the grid would restore to the previous state. The most major obstacle was getting the correct previous state along with passing the control to the correct player in a more-than-2-player game when some of the players had already lost.
- ❖ **Saving and Resuming**: Serialization was a big problem because javafx components are not serializable so we had to figure out a efficient solution. So, we saved the properties of these components using serializable data types and objects, and rebuilt them from scratch on resume.
- ❖ **Animations**: Again, another big obstacle. First issue faced was that the control would pass on to the next player while the animations were playing, thus the player could play their chance and alter the state of the game. We corrected that using animation counts and the setOnFinished function. Next major challenge(since there was many) in animations was that they would continue on for a long time even though it could be seen that a player had won. To tackle this, we again used an animation count and set a threshold upon clearing which the animations would end and the game would get over.

Contribution

- **Gagandeep Singh:**
  Everything starting from the initialization of the game interface, all of it’s buttons, their working(both with mouse and key events), selecting colours for each player, selecting the number of players, selecting the grid size, managing how to quit the game properly and some of its styling. Linking all the scenes together, for example, sending the number of players, the grid size, and the colour for each player, after validation, to start the actual game.
- **Meet Arora:**
  Everything related to the working of the actual game i.e. from the visual appearance of the game, the functionality, the basic AI, the animations in the game, the fade animations of the game scene, to the handling of all the buttons and events during the game, and more features of the game like undo, saving, and resuming.

Extra features

- **Keyboard Mode**
  The entire application is fully configured to be used intuitively just with the keyboard.
- **Sounds**
  The game supports sounds for events such as bursting of the balls, selecting an invalid cell, and when a player wins. It can be turned on and off easily just with a click of a button.
- A **Simple AI** to play against
  Are your friends tired of losing and won't play with you anymore?! We have the perfect solution, a single-player mode where you can play against the computer and improve your strategy. The AI is based on randomisation and makes use of probability to make its moves.