CS 154 PROJECT - CHAIN REACTION

TEAM MEMBERS :

1 : Killari Ramprasad : 170050068
2 : Ambati Satvik : 170050101
3 : Lukka Govinda Siva Nagadev : 170050085

INTRODUCTION :

Our project is a DR RACKET implementation of the game **CHAIN REACTION**.
The objective of this strategic game is to take control of the board by eliminating opponents' orbs. Whoever finally takes control of the board, wins the game.
In our project, we have restricted the number of players from 1 to 3.

DESCRIPTION OF THE PROBLEM :

- The gameplay takes place in an board whose size is given by the user.
- The critical mass of a cell is equal to the number of orthogonally adjacent cells.
- Players take turns to place "orbs" of their corresponding colors. A player can only place an orb in an empty cell or a cell which already contains one or more orbs of his colour.
- When a cell is loaded with a number of orbs equal to its critical mass, the stack immediately explodes. As a result of the explosion, to each of the orthogonally adjacent cells, an orb is added and the initial cell loses as many orbs as its critical mass. The explosions might result in overloading of an adjacent cell and the chain reaction of explosion continues until every cell is stable.
- When a red cell explodes and there are green cells around, the green cells are converted to red and the other rules of explosions still follow.
- The winner is the one who eliminates every other player's orbs.

BASIC DESIGN :

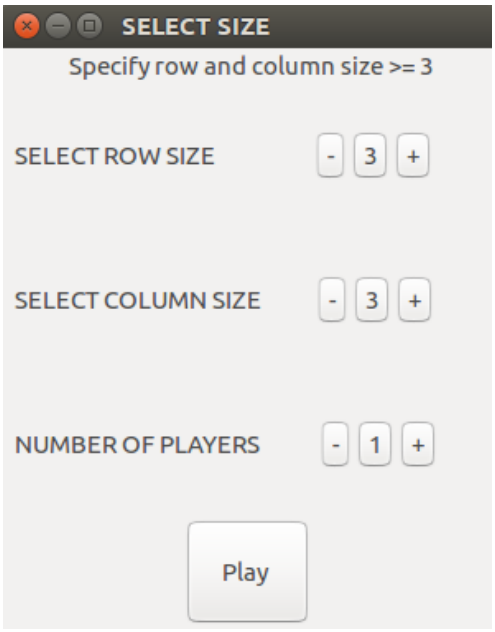
- **GRID :** The grid is created using a 2d-vector of buttons as the elements of the vector. Label of the button indicates the player's color and number of orbs in a particular cell of that particular color. Grid is placed in a frame.
- "**bitmaps.rkt** " **:** This file contains all the images used in the game in .png format.
- **SCORE** : A function used to calculate the score of a board with respect to one player based on some heuristics.
- **BEST-POS :** Given a grid , it returns the best position for the computer to place it's orb maintaining a dominant strategy.
- **UPGRADE** : Returns the grid after the player has placed an orb by completing all the explosions .

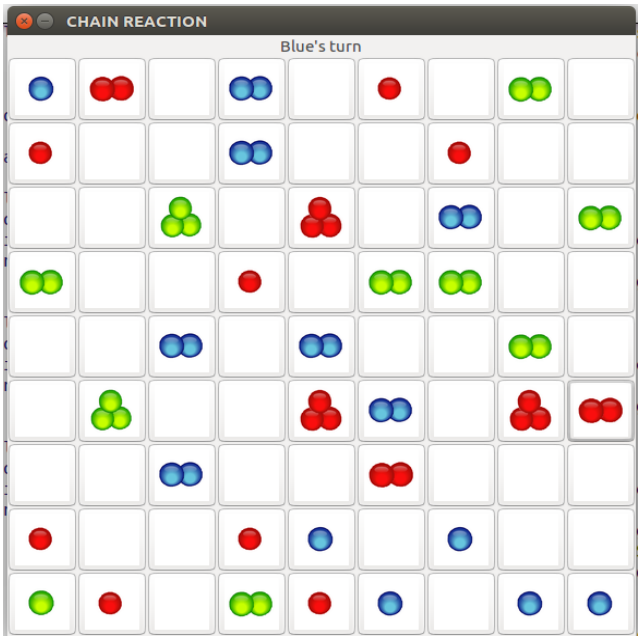
- **MSGSET :** A function used to display the player's turn at the top of the frame , and it changes in cycle after every turn.
- **Graphics** : we have used GUI package to implement graphics and rsound package to implement sound .

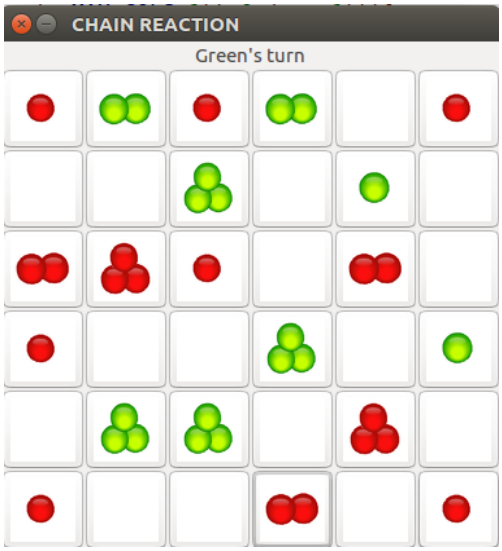
INTERNAL STRUCTURE :

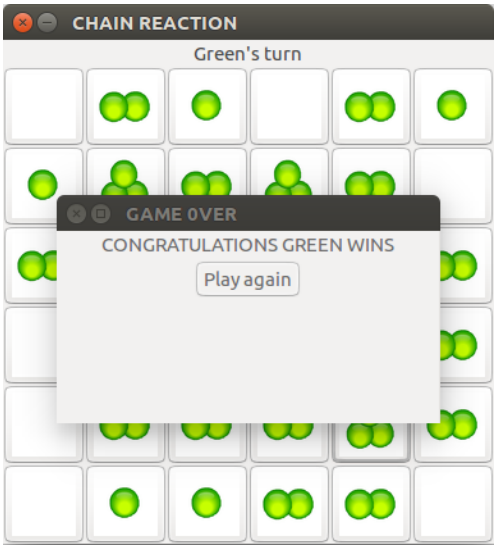
- Many Abstractions have been used to minimize the repetition of code.
- Many State variables and global variables have been defined.
- Many classes provided by inbuilt GUI library have been used such as frame% , button% , dialog% and panel%.

SAMPLE INPUT/OUTPUT:









Game takes mouse event as input and responds by placing an orb of that colour in the cell selected.

OTHER FEATURES :

- **MULTIPLE MODES** : Our game has three modes - single player (with computer) , two player and three player .
- **SELECT SIZE :** The player can select the size of grid of his interest and multiple mode at the beginning of the game .

ALGORITHM OF SINGLE PLAYER :

Heuristic Strategy:

In this section, we develop a way to evaluate the value of a board using heuristics that expert players have gathered through their experience. Like any other complex combinatorial games, the heuristic value is not guaranteed to be an indicative of a dominant strategy but usually it turns out to be so. We shall call a cell critical if the number of orbs in the cell is one less than its critical mass.

1. If the board is a won game, the value is 10000.
2. If the board is a lost game, the value is -10000.
3. For every orb, for every enemy critical cell surrounding the orb, subtract 5 minus the critical mass of that cell from the value.
4. In case that the orb has no critical enemy cells in its adjacent cells at all, add 2 to the value if it is an edge cell or 3 if it is a corner cell.
5. In case that the orb has no critical enemy cells in its adjacent cells at all, add 2 to the value if the cell is critical.
6. For every orb of the player's color, add 1 to the value.
7. For every contiguous blocks of critical cells of the player's color, add twice the number of cells in the block to the score.

8. Computer places it's orb at every possible position and evaluates the score of the board formed after all explosions .

POINTS OF INTEREST :

C**LEVER CODING :**

- Is-game-over? Function checks whether any player has won after every turn .
- Is-eliminated? Function checks whether a player has eliminated after every turn in case of a 3 player mode . and converts the gameplay into a 2 player mode with the help of a global variable curr_players .
- A variable state stops all the explosions after the completion of a game despite of the fact that the function is called.

LIMITATIONS OF OUR PROJECT:

- The single player algorithm works cleverly, but there may come rare instances where computer may lose the game in case of very good players.
- On following the same path of placement of orbs in two different games by the player, the orbs placed by the computer are also in the same position in the two games.
- In one-player, the player is always Red and the computer is Green.
- Graphics aren't used much .