SUBMITTED BY: ANMOL ARORA(130050027) and PRANJAL KHARE(130050028)

**********PROJECT REPORT**********

**********GAME::CHAIN REACTION **********

**********OBJECTIVE**********

This is a strategy game for 2 to 8 players.

The objective of Chain Reaction is to take control of the board by eliminating your opponents' orbs.

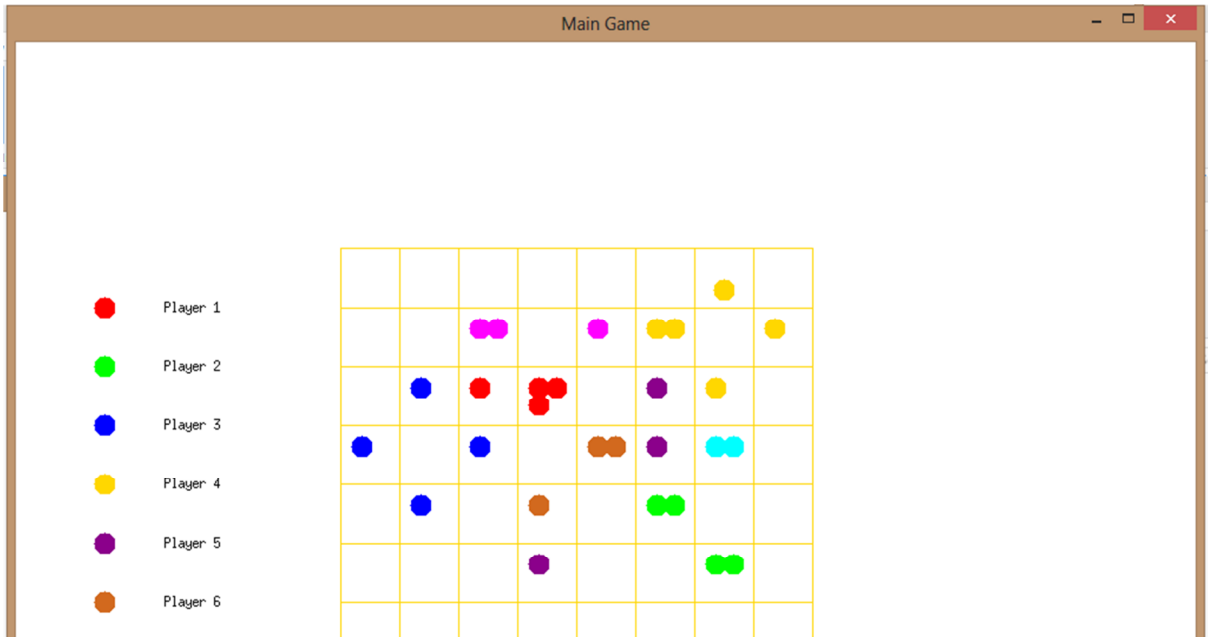
Players take it in turns to place their orbs in a cell.

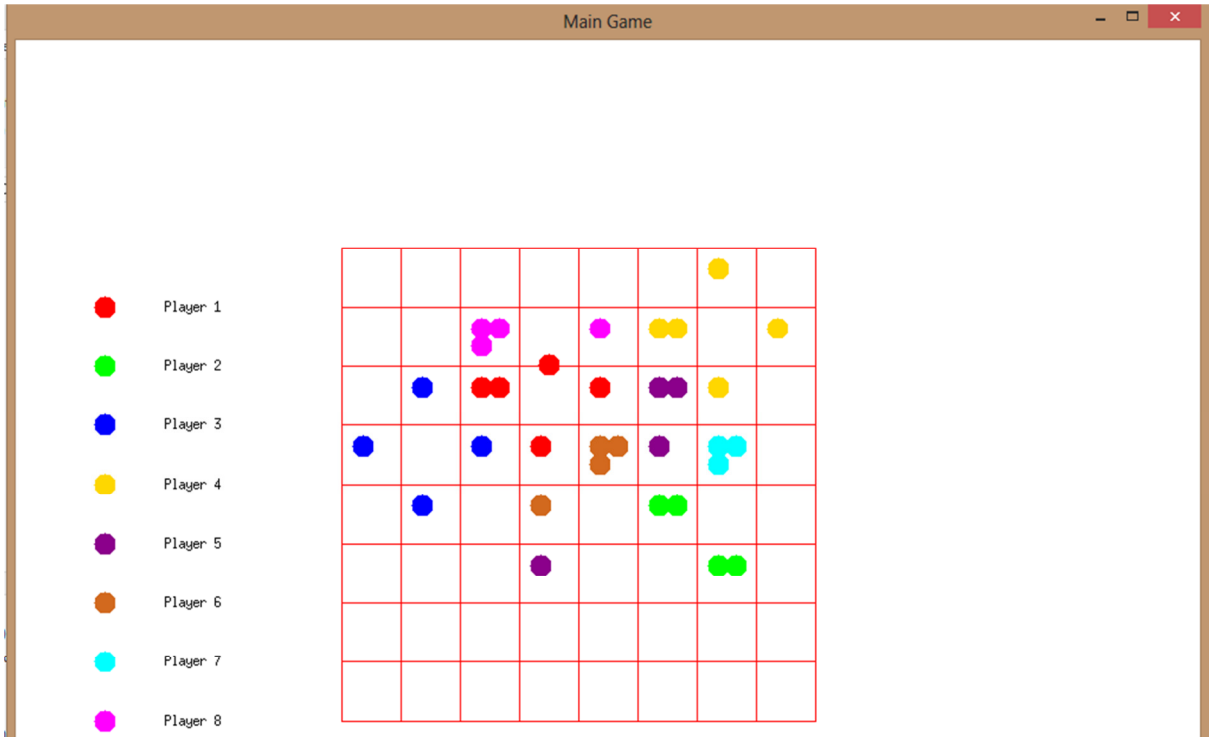
Once a cell has reached critical mass, the orbs explode into the surrounding cells adding an extra orb and claiming the cell for the player.

A player may only place their orbs in a blank cell or a cell that contains orbs of their own colour. As soon as a player loses all their orbs, they are out of the game.

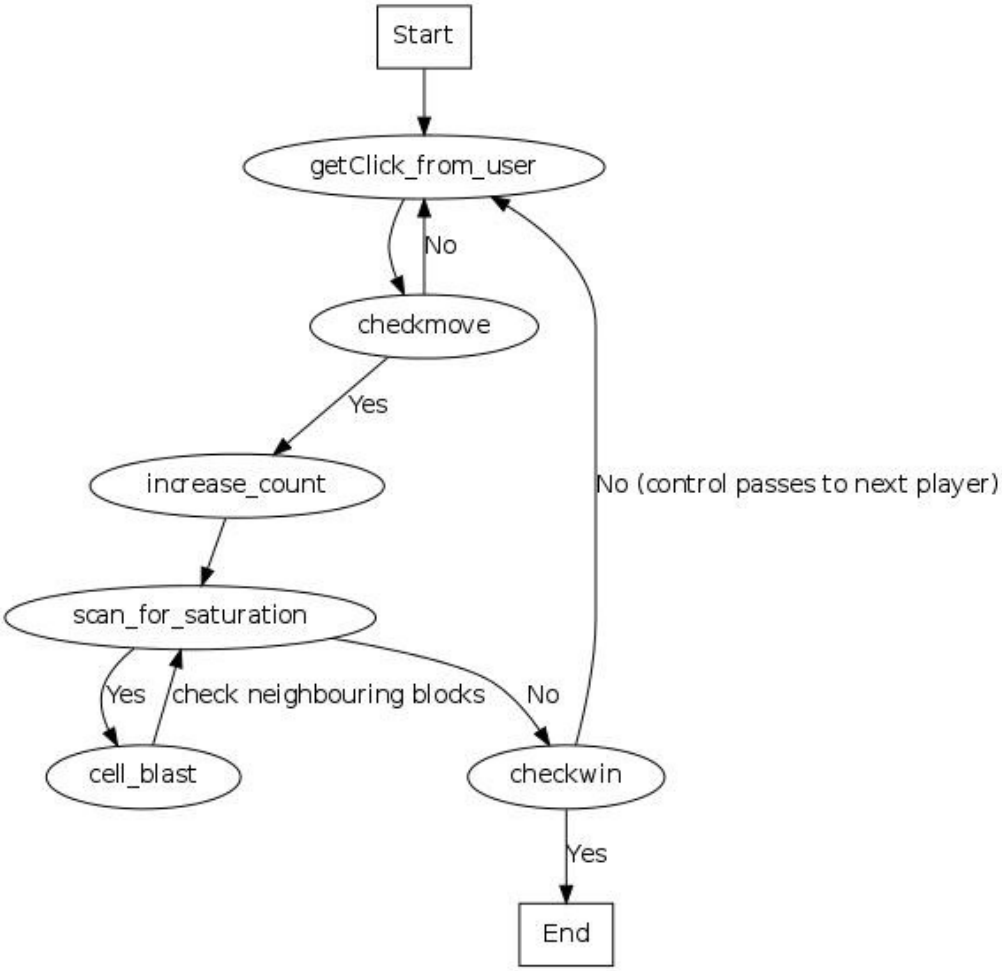
**********INPUT /OUTPUT**********

Since the project is game-based and has **graphical** user interface based on initCanvas, there are no specific test cases. Here are some gameshots:


Main Game
Player 1
Player 2
Player 3
Player 4
Player 5
Player 6


Main Game
Player 1
Player 2
Player 3
Player 4
Player 5
Player 6
Player 7
Player 8

**********ALGORITHM**********



Algorithm for Chain Reaction (Created using dot)

**********MAJOR CLASS**********

1). Block: This class consists of 9 circles and 1 rectangle. The complete grid displayed during the run time is created by an array of Blocks. In short, each square of the grid is a Block. The 9 circles in this class are of two categories: i). The first eight circles are symbolic of the orbs which the block holds ii). The ninth circle which is used for the animation purposes (of moving the orbs during blast).

**********MAJOR FUNCTIONS**********

******MEMBER FUNCTIONS*******

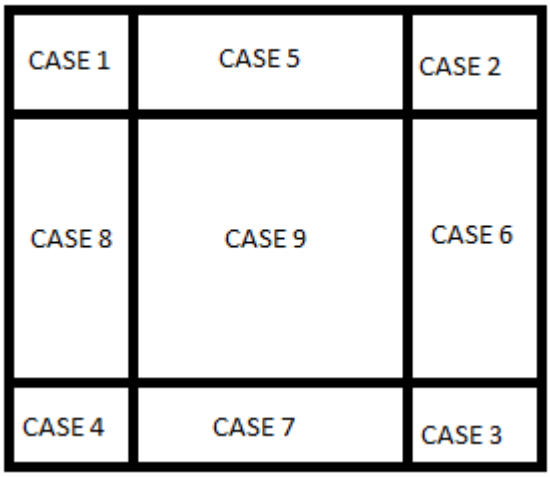
1).orient: This function is responsible for creating the grid at the starting of the game. It resets the squares to their respective positions, describes the initial attributes of the circles and hides them.

2).exit: This function is responsible for hiding the grid and resetting its attributes when the player chooses to play again or quit.

3).up/down/left/right: These four functions are responsible for animating the ball moving to the respective direction when it is called.

*****NON MEMBER FUNCTIONS*****

NOTE: The cells in the grid have been classified into various cases for the working of current algorithm. The cases are as follows:



1). blast: This is a very important function, which performs the following functions sequentially:

i). Sets the count of blasted cell to the required and hides its circles.

ii).Sets the player attribute of neighbouring cells to the player attribute of blasted cell and increase their counts by one each.

iii). Calls the respective up/down/left/right function for showing the path of moving orb.

iv). updates the status of the blasted cell.

v). sets the player attribute of blasted cell to null so that it is available for further clicks by other players.

vi). Calls the scan function to scan the neighbouring cells to check if they are saturated.

2).scan: This function is responsible for deciding whether a particular cell is fit to blast or not. It takes the case, row and column number of the cell to be tested and calls blast function if the cell has qualified to blast.

3).Case: This function takes the coordinate provided by getClick() function and processes it to give the row number, column number and case of the cell inside which the user clicked.

4).checkwin: This function checks if the game has been won. To do this, it goes through every cell and checks its player attributes. It declares the game as won if all the cells have player attributes equal to either null (=0, meaning no player occupies it) or a single player.

5).checkmove: This function checks if the current player's move is legal. The move is marked legal if the player attribute of the clicked cell is either null or equal to the current player.

6).player_present: This function ensures that a player who has lost all its orbs once is not able to return in the game subsequently. This function checks if any cell in the grid has player attribute equal to the current player. If the condition is true then the current player is still in the game and the game continues in normal way, otherwise the current player's value is increased by one (meaning the control is passed to the next player in line). The same function is called again to check if the next player (current_player +1) is in the game or not. Hence, this is a recursive function.

7).show: This function accepts the row and column number of a cell and shows/updates its current status (such as the player occupying it(reflected by the colour of the orb) and the number of orbs in the cell).

8).colour_red/colour_green/colour_blue:These functions take the player code as parameters and return the r/g/b value corresponding to that player.

**********HOW TO RUN THE CODE**********

Compile the file final_project.cpp with s++ compiler.

Command lines: $s++ final_project.cpp -o project

$./project

Then proceed through the program interface.

**********SPECIAL FEATURES**********

1). Provisions for 2-8 players at a time.

2). Each player(1 to 8) has been assigned a particular colour in the program itself, through the use of colour_red/green/blue functions.The colour of the players can be very easily changed by changing their respective rgb values in the functions itself(change required at only one place).

3). Using animation to track the movement of the orbs during blast.

4). Use of buttons at appropriate places which allow the user to freely navigate through the program.

5). Appropriate function added to ensure that a player once eliminated doesn't get to play further (Required condition for a game played by more than 2 players).

6). No loss of balls for any case.

7). Concise and user-friendly interface.