
SYSTEM REQUIREMENTS SPECIFICATIONS

PURPOSE:

The goal of this project is to construct a user-friendly program to enable users to play the game **“CHAIN REACTION”**. It’s a multiplayer game which allows group strength to vary from 2 players to 6 players. Game play proceeds with players strategically placing their orbs so as to control the board. The objective of the game is to win by eliminating orbs of every other player. Elimination occurs by exploding orbs near enemy positions after a certain critical number of orbs have been placed in a particular square. It is inspired by the underlying principle of the chain reactions in chemistry which take place due to instability of heavy nuclei.

BLACK BOXES

As the principles of OOPs stress upon the division of work into different units, we have created several functions allotting specific task to each of them. The following are the functions used in our code and their description:

1. **main()** :

This is the function which will be called when the program is executed. It's basically like a central node from where all the functions are called and where they return to. It has different return values for different cases depending on the behavior of the function it calls. The different return values helps in pinpointing the function due to which the error is occurring.

2. **HowToPlay()** :

This function on being called displays a paragraph on how to go about playing the game. Mouse click is implemented in a region to allow the user to return to the main menu. It is of Boolean type and returns false values incase of any error like failed image loading etc.

3. **NumberOfPlayer()** :

This function basically takes input from the user on the number of players that are going to be playing the game. It displays a clickable box with numbers printed on them. An array of structures was used to implement the different regions of mouse click corresponding to each particular number. It is of type int and returns the number of players.

4. **GamePlay()** :

This function is the backbone of the game. This function runs the game. Throughout its call it changes the various variable values and calls to various functions to enable the game to run. First of all it prints the grid on the screen, thereby creating a click region upon it. It then in accordance with the mouse clicks changes the grid variables and changes turns etc. It also calls for functions to check if the game is over or a particular player has been

knocked out in every turn of the game. It is of Boolean type and returns false if any error occurs.

5. **MainScreen()** :

This function basically sets up a screen and prints the welcome screen on it . It has well-defined clickable regions which lead to different parts in the game like the Play Mode, instructions page or end the game (Exit).

6. **TurnPrinter(int T)** :

This function prints an image corresponding to the person whose turn it is. Ex- If its player 3's turn whose color code is green, then it will display an image displaying number 3 with a green background thereby indicating that player 3 has to play.

7. **CheckWin(int g[][6][2])** :

The victory condition is achieved when only orbs of one color are present on the board. This function will therefore check this condition before every move of all the players as the game is prone to sudden wins and reversals due to the nature of game play. It simply checks the [][][1] two dimensional grid of the main grid checking if there is only a single non-zero number present which would thereby indicate a winning situation. It is of integer type and returns -1 if there is no winner and returns with the ID of corresponding player if a particular player has won.

8. **LocationType(int xx , int yy)** :

This function takes the co-ordinates of the move via mouse click and determines whether the square is a corner(returns 0) , an edge (returns 1) or an inland square (returns 2).

9. **Blast(int x , int y, int Gr[][6][2] , int ii) :**

The function combs the grid for any square where the number of orbs have crossed the critical value. Firstly it calls a function to get to know what kind of square the move has been played in. Thereby it checks the number of balls in the square and checks if it has crossed the critical mass. If it hasn't crossed the critical mass, the functions simply calls a function to print the balls corresponding to the no. of balls. If has crossed the critical mass , the function edits the variable associated with the adjacent squares and then recursively calls the functions on their co-ordinates to check for a successive blast condition.

10. **init() :**

This function simply initialize all the sub-systems of the SDL libraries and prepares them to be used. It is of boolean type and returns false in case of any error.

11. **SDL_Surface *load_image(std::string filename) :**

This function takes a filename as a parameter and loads a pointer pointing to this image on a pointer pointing to a SDL_Surface type variable.

e.g.

SDL_Surface* A = load_img("FILENAME");

12. **apply_surface(int x, int y, SDL_Surface* source, SDL_Surface* destination, SDL_Rect* clip = NULL) :**

This function blits (i.e. puts the image on the screen to display) . Its take the offset(x,y) which is the point of the upper left corner of the image. Source is the pointer to the SDL_Surface which has the image loaded on it. The destination is the pointer to the SDL_Surface which is the

screen.

13. checkPlayerExist(int l, int g[][6][2]) :

This functions takes the ID of a player as the parameter and checks the [][][1] of the grid. If there is no match means the player doesn't exist and the function returns false else it returns true. This function helps in the purpose of eliminating a player during multiplayer mode when all of his orbs have been conquered by other players.

14. BlitBalls(int id, int quant, int xx, int yy, int G[][6][2]):

This crucial function prints the balls on the screen at the offset xx,yy of the particular player(id). It consists of switch-case statements which pinpoint the colour and quantity of the balls to be printed.

15. GameOver(int ID) :

If a player has won , the Gameplay functions will invoke a call to this function which will change the screens and print the corresponding victory message. It will also provide a button to return to the main menu.

SOFTWARE REQUIREMENTS

Applications required :

SOFTWARE NAME	VERSION	DOWNLOAD LINK
Code::Blocks	13.12	http://www.codeblocks.org/downloads/26
GCC Compiler	4.8.1, 32 bit.	http://www.codeblocks.org/downloads/26
GDB Debugger	-	http://www.codeblocks.org/downloads/26

Libraries required :

Name of Library	Where to get it ?
IOstream	Preinstalled in Code::Blocks
Stdio	Preinstalled in Code::Blocks
Cstring	Preinstalled in Code::Blocks
SDL & its extension libraries.(SDL_img.h, SDL_ttf.h)	http://lazyfoo.net/tutorials/SDL/01_hello SDL/index.php
	-