

PROJECT REPORT

TOPIC- CHAIN REACTIONS

Date - 23rd November 2014

GROUP 16. SLOT-6

Team members-

1. Ankush Mukherjee (team leader)- 14D100019
2. Prajesh Jangale - 14D100009
3. Akash Chikhalikar - 14D100004
4. Aditya Kalra - 14D100014

This project gives the details of the different parts of the code; work done by the team members and how to play the game.

About the game-

Chain reaction is a strategy game. Our project will include the 2 player version of it and also the one player version. It starts with a grid of 2D array with both players having different colours. When the player clicks on one of the boxes in the grid the number of orbs increase by one. When the number of orbs exceeds the critical number it can hold, it blasts and captures the opponents orbs in all horizontal and vertical directions with one step each. If this blast causes another box to exceed its critical number, we get a series of blasts. The objective of the game is to finish the number of orbs of the opponent.

Codeblocks integrated with simplecpp was downloaded from <http://www.cse.iitb.ac.in/~cs101/project.html>

Libraries included in our code :

```
#include <simplecpp>
#include <cstdio>
#include <cmath>
using namespace std;
```

A class named 'game' is created that includes all the class functions, constructor and destructor.

Coded by Ankush.

```
class game
{
public:
    int prev[10][6];
    int **grid ;
```

Constructor used to initialize prev[][] and grid

```
game()
{
    for(int i=0; i<10; i++)
        for(int j=0; j<6; j++)
            prev[i][j]=0;
    grid =new int*[10];
```

```
for(int i=0; i<10; i++)
{
    grid[i]=new int[6];
}
}
```

Member functions of class -

```
void display();
void display(int grid[10][6]);
int checkWin(int **grid,int player_no);
int isBorder(int i,int j);
int isCorner(int i,int j);
void add(int i,int j,int **grid,int player_no);
```

Destructor to destroy the object created

```
~game()
{
if(grid!=NULL)
    delete []grid;
}
};
```

Display() function : to display all the orbs according to the given conditions from the main function. According to the given case we will have to print orb in the given square. The orb of player 1 will have green and that of player 2 will be

blue. While playing with the computer, the player gets blue and the computer gets green .

The graphics part of this function was done by Prajesh and Akash.

```
void game::display()  
{  
}
```

The function display() displays the 2d array(pointer to pointer or int **)gridThe function display(int [][])displays 2D array. The function display(The concept of function overloading is used here.)

```
void game::display(int grid[10][6]  
{  
  
}
```

The function checks if one of the players has already won the game or not.

checkWin function was made by Akash.

```
int game::checkWin(int **grid,int player_no)  
{  
}
```

isBorder checks whether the box (i+1,j+1) is on the border or not.

This function was made by Prajesh.

```
int game::isBorder(int i,int j)
{
}
```

isCorner checks whether the box (i+1,j+1) is on the corner or not

This function was made by Prajesh.

```
int game::isCorner(int i,int j)
{
}
```

add () adds an orb of the player to the square. If the number of orbs in the square is exceeding the critical number of that square then it explodes, i.e the orbs of the player expand to the neighbouring squares. If this explosion triggers the explosion of the neighbouring square then this function does the same for that square to. The process continues recursively.

This function was made by Ankush.

```
void game::add(int i,int j,int **grid,int player_no)
{
}
```

validateInput checks whether the input is valid or not.

This function was made by Akash.

```
int validateInput(int **grid,int i ,int j, int player_no)
{
}
```

main function:

```
int main()
{
    bool newgame=true;
```

initCanvas makes a new window where we're going to play the game!

```
    initCanvas("CHAIN REACTIONS",500,500);
    do
    {
        if(newgame==false)
        {
            Rectangle blank(250,250,490,490);
            blank.setColor(COLOR("white"));
```

```
    blank.setFill();
    blank.imprint();
}
```

The next loop prints a grid of 10 by 6 for the main playfield.

```
for(int i=1; i<7; i++)
{
    for(int j=1; j<11; j++)
    {
        Rectangle r(90+40*i,50+40*j,40,40);
        r.imprint();

    }
}
```

The following statements show the colours of the players on the window.

```
Text tt(150,40,"ONE PLAYER");
tt.imprint();
Rectangle r_option1(150,40,100,20);
r_option1.setColor(COLOR("red"));
r_option1.imprint();
```

```
Text Tt(290,40,"TWO PLAYER");
```

```
Tt.imprint();
Rectangle r_option2(290,40,100,20);
r_option2.setColor(COLOR("red"));
r_option2.imprint();
float X,Y;
int choice;
while(true)
{
    int inp=getClick();
    X=inp/65536;
    Y=inp%65536;
```

The following statement checks if the player has selected one player or two player game. It also assigns the colours to the player and/or computer according to the selection done by the player.

```
if((X<172.5)&&(X>127.5)&&(Y<50)&&(Y>30))
{
    choice=1;
    Text t1(100,480,"PLAYER1-COMPUTER");
    t1.imprint();
    Circle c1(147,480,5);
    c1.setColor(COLOR("green"));
    c1.setFill();
```

```
c1.imprint();
Text t2(353,480,"PLAYER");
t2.imprint();
Circle c2(390,480,5);
c2.setColor(COLOR("blue"));
c2.setFill();
c2.imprint();

break;
}
else if((X>240)&&(X<340)&&(Y<50)&&(Y>30))
{
    choice=0;
    Text t1(110,480,"PLAYER-1");
    t1.imprint();
    Circle c1(149,480,5);
    c1.setColor(COLOR("green"));
    c1.setFill();
    c1.imprint();
    Text t2(353,480,"PLAYER-2");
    t2.imprint();
    Circle c2(395,480,5);
```

```
        c2.setColor(COLOR("blue"));
        c2.setFill();
        c2.imprint();

        break;
    }
    else
        continue;

}

r_option1.setColor(COLOR("white"));
r_option1.setFill();
r_option1.imprint();

r_option2.setColor(COLOR("white"));
r_option2.setFill();
r_option2.imprint();
```

The following statement creates the undo button for the two players.

```
Text t1(150,40,"UNDO");           //Creates a box for
function "UNDO"
```

```
Rectangle r_undo(150,40,45,20);  
r_undo.setColor(COLOR("red"));
```

```
t1.imprint();
```

```
r_undo.imprint();
```

```
if(choice==1)
```

```
{
```

```
    r_undo.setColor(COLOR("white"));
```

```
    r_undo.setFill();
```

```
    r_undo.imprint();
```

```
}
```

```
wait(1);
```

The following statements initialize all the running variables to zero.

```
int i=0,j=0,k=0,l=0;
    bool undo=true;
    game ob;
    int no=1;
```

The following if statement checks if the system has crashed or not and declares it as a memory allocation failure .

```
if(ob.grid==NULL)
{
    cout<<"Memory allocation failure";
    return -1;
}
```

```
for(int i=0; i<10; i++)
{
    if(ob.grid[i]==NULL)
    {
        cout<<"Memory allocation failure";
        return -1;
    }
}
```

```
    for(int j=0; j<6; j++)
        ob.grid[i][j]=0;

}

int count=0;
```

Function of the following statement is to open the history file and stores all the previous steps and their configuration.

```
FILE *history = fopen("historydata","wb+");

int t[10][6];
for(int i=0; i<10; i++)
    for(int j=0; j<6; j++)
    {
        t[i][j]=0;
    }
int size=sizeof(t);
fwrite(t,size,1,history);
do
{
    ++count;
```

```
no=1-no;
float x,y;
while(true)
{
    if(choice==0 || no==1)
    {
        int v = getClick();
        x=v/65536,y=v%65536;

    }
}
```

Here, the computer is choosing randomly to create an orb in that place.

```
else
{
    x=rand()%241+110.1;
    y=rand()%401+70.1;

}
```

The following part of the code is to perform the undo step if the player has chosen to do so.

```

if((x<172.5)&&(x>127.5)&&(y<50)&&(y>30)&&(count!=1)&&(u
ndo==true)&&(choice==0))
    {
        for(int i=0; i<10; i++)
            for(int j=0; j<6; j++)
                ob.grid[i][j]=t[i][j];
        no=1-no;
        ob.display();           //display current
configuration of the game.
        count--;
        undo=false;
    }
else if((x<110)|| (x>350)|| (y<70)|| (y>470))
    {
        continue;
    }

```

The floor function is used to round off the coordinate clicked by the player to an integer below.

```

else
    {
        k=floor((x-110)/40);
        l=floor((y-70)/40);
    }

```

```

if(validateInput(ob.grid,l+1,k+1,no+1)==0)

{

    continue;

}

else
{
    if((choice==1)&&(count%2==1))
    {
        Rectangle r(90+40*(k+1),50+40*(l+1),40,40);
        r.setColor(COLOR("green"));
        r.imprint();
        wait(0.5);

        Rectangle
r1(90+40*(k+1),50+40*(l+1),40,40);
        r1.imprint();
    }
    for(int i=0; i<10; i++)
        for(int j=0; j<6; j++)
            t[i][j]=ob.grid[i][j];
}

```

```
fwrite(t,size,1,history);
```

The use of files to store steps of the game was coded by Ankush.

```
        break;
    }
}
}
```

```
i=l;
```

```
j=k;
```

```
ob.add(i,j,ob.grid,no+1);
```

```
ob.display();//print current configuration of array
```

```
if(count>2)
```

```
    if(ob.checkWin(ob.grid,no+1))
```

```
    {
```

```
        break;;
```

```
    }
```

```
fwrite(t,size,1,history);  
undo=true;  
  
}  
while(count<=1 || (!ob.checkWin(ob.grid,no+1)));  
  
if(no+1==1)  
{  
    Text t3(250,10,"WINNER - PLAYER 1");  
  
    t3.imprint();  
}  
else  
{  
    Text t4(250,10,"WINNER - PLAYER 2");  
    t4.imprint();  
}
```

To show replay option and get the click from the player to select a replay option.

```
Rectangle r_REPLAY(350,40,55,20);  
Text t2(350,40,"REPLAY");
```

```
r_REPLAY.setColor(COLOR("red"));
r_REPLAY.imprint();
t2.imprint();
int v=getClick();
float x=v/65536;
float y=v%65536;

if(x<372.5&&x>327.5&&y<50&&y>30)
{
    for(int i=1; i<7; i++)
    {
        for(int j=1; j<11; j++)
        {
            Rectangle r1(90+40*(i),50+40*(j),35,35);
            r1.setColor(COLOR("white"));
            r1.setFill();
            r1.imprint();
        }
    }
    rewind(history);
```

```
for(int i=0; i<10; i++)
    for(int j=0; j<6; j++)
        ob.prev[i][j]=0;
while(!feof(history))
{
    for(int i=0; i<10; i++)
        for(int j=0; j<6; j++)
            t[i][j]=0;
    fread(t,size,1,history);
    ob.display(t);
    wait(0.5);

}
wait(1);
ob.display();
}
wait(1);

for(int i=1; i<7; i++)
{
    for(int j=1; j<11; j++)
    {
```

```
    Rectangle r(90+40*i,50+40*j,40,40);
    r.setColor(COLOR("white"));
    r.setFill();
    r.imprint();
}
}
Rectangle r_undo1(150,40,45,20);
r_undo1.setColor(COLOR("white"));
r_undo1.setFill();
r_undo1.imprint();
Rectangle r_REPLAY1(350,40,55,20);
r_REPLAY1.setColor(COLOR("white"));
r_REPLAY1.setFill();
r_REPLAY1.imprint();
```

To print the choice for the player to start the game again by giving a choice of yes or no.

```
Rectangle next_GAME(250,200,300,20);
next_GAME.setColor(COLOR("red"));
next_GAME.imprint();
Text t5(250,200,"Would you like to play again?");
t5.imprint();
```

```
Rectangle yes(200,230,30,20);
Rectangle nO(300,230,20,20);
yes.imprint();
nO.imprint();
Text t_no(300,230,"NO");
Text t_yes(200,230,"YES");
t_no.imprint();
t_yes.imprint();

do
{
    int u=getClick();
    x=u/65536;
    y=u%65536;
    if((x<215)&&(x>185)&&(y<240)&&(y>220))
    {
        newgame=false;
        break;
    }
    else if((x<310)&&(x>290)&&(y<240)&&(y>220))
    {
        return 1;
    }
}
```

```
    }  
    else  
    {  
        continue;  
    }  
}  
while(true);  
}  
while(true)  
}
```

Final compilation and debugging of the parts submitted by different members was done by Ankush.

Acknowledgements-

The whole project was a great learning experience. It helped us a lot to implement our own thought and express it in form of a code. All the concepts were made clear through this project work.

We also learned the importance of team work and how a team coordinates.

The best part was the writing of the diary which helped us to plan the work and complete it well within time.

The help and support given by our T.A **Sourabh Ghurye** boosted the progress of our project.

Finally grand accolades to Prof. Deepak .B. Phatak and Prof. Supratik Chakraborty for organizing such enthusiastic and a wonderful project work!!!