**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**Jnana Sangama, Belgaum-590018**



**A Computer Graphics Mini Project Report**

**on**

**“CHAIN REACTION v2.0”**

**Submitted in Partial fulfillment of the Requirements for the VI Semester of the Degree of**

**Bachelor of Engineering**

**In**

**Computer Science & Engineering**

**By**

**ADITYA M. KAKDE**

**(1CR16CS008)**

**AKASH A S**

**(1CR16CS001)**

**Under the Guidance of**

**Kartheek G C R**

**Assistant Professor, Dept. of CSE**



# DEPARTMENT OF Computer SCIENCE AND ENGINEERING

**CMR INSTITUTE OF TECHNOLOGY**

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI,

BANGALORE-560037

**CMR INSTITUTE OF TECHNOLOGY**

1. #132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI,
2. BANGALORE-560037

# DEPARTMENT OF Computer SCIENCE AND ENGINEERING

1. ****

**CERTIFICATE**

This is to certify that the Computer Graphics Project work entitled **“Chain Reaction Game v2.0”** has been carried out by **Akash A S (1CR16CS001)** and **Aditya M. Kakde (1CR16CS008)** bonafide students of CMR Institute of Technology in partial fulfillment for the award of **Bachelor of Engineering** in **Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year **2018-2019**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. This Computer Graphics Project Report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

**----------------------- ----------------------**

**Signature of Guide Signature of HOD**

**Kartheek G C R Mrs. Jhansi Rani**

**Assistant Professor Professor, Head**

**Dept. of CSE, CMRIT Dept. of CSE, CMRIT**

External Viva

Name of the examiners                                                                          Signature with date

1.

2.

**ABSTRACT**

OpenGL provides a set of commands to render a three dimensional scene. That means

you provide them in an Open GL-useable form and Open GL will show this data on the screen(render it). It is developed by many companies and it is free to use. You can develop Open GL- applications without licensing.

Open GL is a hardware- and system dependent interface. An Open GL-application will

work on every platform, as long as there is an installed implementation, because it is system independent, there are no functions to create windows etc., but there are helper functions for each platform. A very useful thing is GLUT.

We recreated the popular Android game, Chain Reaction, for the partial fulfillment of credits towards our Computer Graphics Laboratory. The game is however a desktop version. Our primary aim is to implement the entire game using OpenGL libraries and functions, and build the game from scratch. The number of players can, however, vary from 2 to upto 5. At last, it would be a comprehensible achievement, if this project reaches the hands of others who can not only play this game, but also make modifications and take it to greater heights.

**ACKNOWLEDGEMENT**

The satisfaction and euphoria that accompany a successful completion of any task would be incomplete without the mention of people who made it possible, success is the epitome of hard work and perseverance, but steadfast of all is encouraging guidance.

So with gratitude I acknowledge all those whose guidance and encouragement served as beacon of light and crowned our effort with success.

I would like to thank **Dr. Sanjay Jain,** Principal, CMRIT, Bangalore for providing excellent academic environment in the college and his never-ending support for the B.E program.

I would also like to thank **Mrs. Jhansi Rani,** Professor andHOD, Department of Computer Science and Engineering, CMRIT, Bangalore who shared her opinions and experiences through which I received the required information crucial for the project.

I consider it a privilege and honour to express my sincere gratitude to my internal guide **Kartheek G C R,** Asst. Professor, Department of Computer Science & Engineering, CMRIT, Bangalore for their valuable guidance throughout the tenure of this project work.

I would also like to thank all the faculty members who have always been very Co-operative and generous. Conclusively, I also thank all the non-teaching staff and all others who have done immense help directly or indirectly during our project.

**AKASH A S**

**ADITYA M. KAKDE**

**TABLE OF CONTENTS**

**Title Page No**

ABSTRACT i

ACKNOWLEDGEMENT ii

LIST OF FIGURES AND TABLES iv INTRODUCTION 1

SYSTEM REQUIRMENTS 2

IMPLEMENTATION 3

CODE 5

SCREENSHOTS 6

CONCLUSION AND FUTURE SCOPE 17

BIBILOGRAPHY 18

**LIST OF FIGURES AND TABLES**

|  |  |  |
| --- | --- | --- |
| Sl. No. | Description | Page No |
| 1 | 5.1 Opening Screen of game | 14 |
| 2 | 5.2 Play Area (grid) | 15 |
| 3 | 5.3 Orbs Spinning | 16 |
| 4 | 5.4 End game Screen | 16 |
| 5 | 5.5 Multiplayer Orbs | 17 |

**CHAPTER 1**

**INTRODUCTION**

The main Agenda of this project was to recreate the popular Android game, Chain Reaction. We created the project from scratch using OpenGL libraries and implemented the graphics required by the game. The game’s algorithm was however implemented using C.

Chain Reaction v.2.0, is our version of the game for PCs. The game consists of subtle yet interesting graphics. The game being multi-player (offline), can be played by 2 to 5 players simultaneously, each player uniquely identified.

The main idea of the game is however very simple, it is to take control of the board by eliminating your opponents orbs. Players takes 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 cells for the player. A player may only place their own orbs in a blank cell or a cell that contains orbs of their colour. As soon as a player loses all their orbs they are out of the game.

**CHAPTER 2**

**SYSTEM REQUIREMENTS**

1. Hardware Requirements

**Processors:** Intel i3,i5,i7

**Processor Speed:** 3.00GHZ

**RAM:** 4GB

**Storage:** 500GB

**Monitor:** 15inches

**Keyboard:** Standard 102 keys

**Mouse:** Standard 3 buttons

1. Software Requirements

 Operating System can be either Windows/Linux. The project is compatible with either of them.

 OpenGL libraries and GLUT installed

 C compiler

**CHAPTER 3**

**IMPLEMENTATION**

We used Visual Studio for as our IDE and used OpenGL to implement the graphics needed for the game. This enabled us to make the streamlined yet subtle graphics for is appealing yet simplistic in form. The entire algorithm and implementation is written in C.

Lastly, for interaction with the game, the user will however have to use a mouse.

**CHAPTER 4**

**CODE**

4.1 BACKGROUND MATRIX:

void array\_to\_screen(int x, int y)

{

arr\_s[0] = -1;

arr\_s[1] = -1;

arr\_s[2] = -1;

arr\_s[3] = -1;

if (x >= 0 && x < rows && y >= 0 && y < cols)

{

arr\_s[0] = (y \* 5);

arr\_s[1] = (y \* 5) + 5;

arr\_s[2] = -(x \* 5);

arr\_s[3] = -(x \* 5) - 5;

}

}

void screen\_to\_array(int x, int y)

{

screen\_a[0] = -1;

screen\_a[1] = -1;

int x\_val = ((int)(x / x\_width));

int y\_val = ((int)(y / y\_width));

if (x\_val > 0 && x\_val <= cols && y\_val > 0 && y\_val <= rows) //check if it lies within boundary

{

x\_val = (x\_val - 1) % cols; // sets x and y in terms of array indexes

y\_val = (y\_val - 1) % rows;

screen\_a[1] = x\_val;

screen\_a[0] = y\_val;

}

}

void generateEmptyPlane()

{

for (int i = 0;i < b\_h;i++)

{

for (int j = 0;j < b\_l;j++)

{

board[i][j][0] = 0;

board[i][j][1] = -1; //this is id

}

}

}

void displayGrid()

{

for (int i = 0;i < b\_h;i++)

{

for (int j = 0;j < b\_l;j++)

{

printf("%d\t", board[i][j][0]);

}

printf("\n");

}

}

4.2 ORBS:

void computeCircle(float radius) //called once

{

float radiance, x, y;

for (int angle = 0;angle < 362;angle++)

{

radiance = (pi\*angle) / (float)180;

x = (radius)\*cosf(radiance);

y = (radius)\*sinf(radiance);

circle[angle][0] = x;

circle[angle][1] = y;

}

}

void computeCircle2(float radius) //called once

{

float radiance, x, y;

for (int angle = 0;angle < 362;angle++)

{

radiance = (pi\*angle) / (float)180;

x = (radius)\*cosf(radiance);

y = (radius)\*sinf(radiance);

circle\_move[angle][0] = x;

circle\_move[angle][1] = y;

}

}

void translateCircleMove(float h, float k)

{

float x, y;

for (int angle = 0;angle < 362;angle++)

{

x = circle\_move[angle][0];

y = circle\_move[angle][1];

glBegin(GL\_LINES);

glVertex2f(h, k);

glVertex2f(h + x, k + y);

glEnd();

}

}

void translateCircle(float h, float k,int c)

{

float x, y;

for (int angle = 0;angle < 362;angle+=20)

{

x = circle[angle][0];

y = circle[angle][1];

if (c == 2)

{

glBegin(GL\_LINES);

glVertex2f(h + ellipse\_1[i\_x][0], k + ellipse\_1[i\_x][1]);

glVertex2f(h + ellipse\_1[i\_x][0] + x, k + ellipse\_1[i\_x][1] + y);

glEnd();

glBegin(GL\_LINES);

glVertex2f(h, k);

glVertex2f(h + x, k + y);

glEnd();

}

if (c == 3)

{

glBegin(GL\_LINES);

glVertex2f(h + ellipse\_2[i\_y][0], k + ellipse\_2[i\_y][1]);

glVertex2f(h + ellipse\_2[i\_y][0] + x, k + ellipse\_2[i\_y][1] + y);

glEnd();

glBegin(GL\_LINES);

glVertex2f(h + ellipse\_1[i\_x][0], k + ellipse\_1[i\_x][1]);

glVertex2f(h + ellipse\_1[i\_x][0] + x, k + ellipse\_1[i\_x][1] + y);

glEnd();

glBegin(GL\_LINES);

glVertex2f(h, k);

glVertex2f(h + x, k + y);

glEnd();

}

if (c == 1)

{

glBegin(GL\_LINES);

glVertex2f(h, k);

glVertex2f(h + x, k + y);

glEnd();

}

}

}

4.3 MOUSE FUNCTION:

void mouse(int button, int state, int x, int y)

{

if (no\_of\_players\_selected && !can\_click\_mouse)

{

screen\_to\_array(x, y);

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_UP)

{

if (screen\_a[0] == 5 && (screen\_a[1] == 3 || screen\_a[1] == 4))

{

start\_game = true;

can\_click\_mouse = true;

glutDestroyMenu(glutGetMenu());

glutPostRedisplay();

return;

}

}

}

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_UP && can\_click\_mouse)

{

screen\_to\_array(x, y);

if (screen\_a[0] != -1 && screen\_a[1] != -1) //if the click is valid

{

do\_rotation = true;

detectRule(screen\_a[0], screen\_a[1],turn);

}

}

if (button == GLUT\_RIGHT\_BUTTON && state == GLUT\_UP)

{

printf("Recurisive id\tFrom\t\tTo\t\tID\tExcitation\n");

//animate();

//startAnimation = true;

for (int i = 0;i <= r\_rows;i++)

{

for (int j = 0;j < 4;j++)

printf("%d\t\t%d\t%d\t%d\t%d\t%d\t%d\n", i, recursive\_array\_2[i][j][0][0], recursive\_array\_2[i][j][0][1], recursive\_array\_2[i][j][1][0], recursive\_array\_2[i][j][1][1], recursive\_array\_2[i][j][2][1], recursive\_array\_2[i][j][2][0]);

}

printf("\n");

for (int i = 0;i < no\_of\_players;i++)

{

if(player\_allowed\_to\_play[i])

printf("player %d is %s\t",i, "True");

}

printf("\n");

}

}

4.4 ANIMATION OF ELLIPSE:

void animate()

{

for (int j = 0;j < 4;j++)

{

if (!(recursive\_array[cell\_counter][j][0][0] == recursive\_array[cell\_counter][j][1][0] && recursive\_array[cell\_counter][j][0][1] == recursive\_array[cell\_counter][j][1][1]))

{

//create a circle and move it

if (recursive\_array[cell\_counter][j][0][0] == recursive\_array[cell\_counter][j][1][0])

{

int sign = recursive\_array[cell\_counter][j][0][1] > recursive\_array[cell\_counter][j][1][1] ? -1 : 1;

if (recursive\_array[cell\_counter][j][0][1]!= recursive\_array[cell\_counter][j][1][1])

{

translateCircleMove(recursive\_array[cell\_counter][j][0][0],recursive\_array[cell\_counter][j][0][1] + 1\*sign); //draw a basic circle

movementBuffer[j][0][0] = recursive\_array[cell\_counter][j][0][0];

movementBuffer[j][0][1] = recursive\_array[cell\_counter][j][0][1] + 1 \* sign;

recursive\_array[cell\_counter][j][0][1] += (1 \* sign);

}

}

else

{

int sign = recursive\_array[cell\_counter][j][0][0] > recursive\_array[cell\_counter][j][1][0] ? -1 : 1;

if (recursive\_array[cell\_counter][j][0][0]!= recursive\_array[cell\_counter][j][1][0]) {

translateCircleMove(recursive\_array[cell\_counter][j][0][0] + 1 \* sign, recursive\_array[cell\_counter][j][0][1]); //draw a basic circle

movementBuffer[j][0][0] = recursive\_array[cell\_counter][j][0][0] + 1 \* sign;

movementBuffer[j][0][1] = recursive\_array[cell\_counter][j][0][1];

recursive\_array[cell\_counter][j][0][0]+= (1 \* sign);

}

}

}

}

Drawing = true;

}

4.5 DISPLAY FUNCTION:

void display(void) {

glClearColor(0, 0, 0, 1);

glClear(GL\_COLOR\_BUFFER\_BIT);

if (endgame)

{

endGame();

drawBitmapText("PLAYER % WINS", c\_x-8, c\_y + 10, 0);

glFlush();

}

else

{

if (!start\_game)

{

glColor3f(0.1, 0.1, 0.1);

glBegin(GL\_POLYGON);

glVertex2f(c\_x - (l\_margin \* 2 / (float)2), c\_y + (t\_margin / (float)2)); //top left

glVertex2f(c\_x + (l\_margin \* 2 / (float)2), c\_y + (t\_margin / (float)2)); //top right

glVertex2f(c\_x + (l\_margin \* 2 / (float)2), c\_y - (t\_margin / (float)2)); //bottom right

glVertex2f(c\_x - (l\_margin \* 2 / (float)2), c\_y - (t\_margin / (float)2)); //top left

glEnd();

glColor3f(1, 1, 1);

drawBitmapText("Play Game", c\_x - 4, c\_y - 0.5, 0);

glColor3f(1, 1, 1);

drawBitmapText(intro, -2, c\_y - 10, 0);

glFlush();

}

if (start\_game && no\_of\_players\_selected)

{

if (!player\_win)

{

glColor3f(color[turn][0], color[turn][1], color[turn][2]);

for (int i = 0;i < cols + 1;i++)

{

glBegin(GL\_LINES);

glVertex2i(i \* 5, 0);

glVertex2i(i \* 5, -rows \* 5);

glEnd();

}

for (int i = 0;i < rows + 1;i++)

{

glBegin(GL\_LINES);

glVertex2i(0, -i \* 5);

glVertex2i(cols \* 5, -i \* 5);

glEnd();

}

glPushMatrix();

glTranslatef(translate\_factor, translate\_factor, 0);

glBegin(GL\_LINES);

for (int i = 0;i < cols + 1;i++)

{

glVertex2i(i \* 5, 0);

glVertex2i(i \* 5, -rows \* 5);

}

for (int i = 0;i < rows + 1;i++)

{

glVertex2i(0, -i \* 5);

glVertex2i(cols \* 5, -i \* 5);

}

glEnd();

glPopMatrix();

for (int j = 0;j < rows + 1;j++)

{

for (int i = 0;i < cols + 1;i++)

{

glBegin(GL\_LINES);

glVertex2f(i \* 5, j\*(-5));

glVertex2f(i \* 5 + (translate\_factor), (j\*-5) + (translate\_factor));

glEnd();

}

}

frame\_draw();

glColor3f(color[turn][0], color[turn][1], color[turn][2]);

if (Drawing)

{

for (int i = 0;i < 4;i++)

{

if (movementBuffer[i][0][0] != 0 && movementBuffer[i][0][1] != 0)

{

translateCircleMove(movementBuffer[i][0][0], movementBuffer[i][0][1]);

}

}

}

glFlush();

}

else

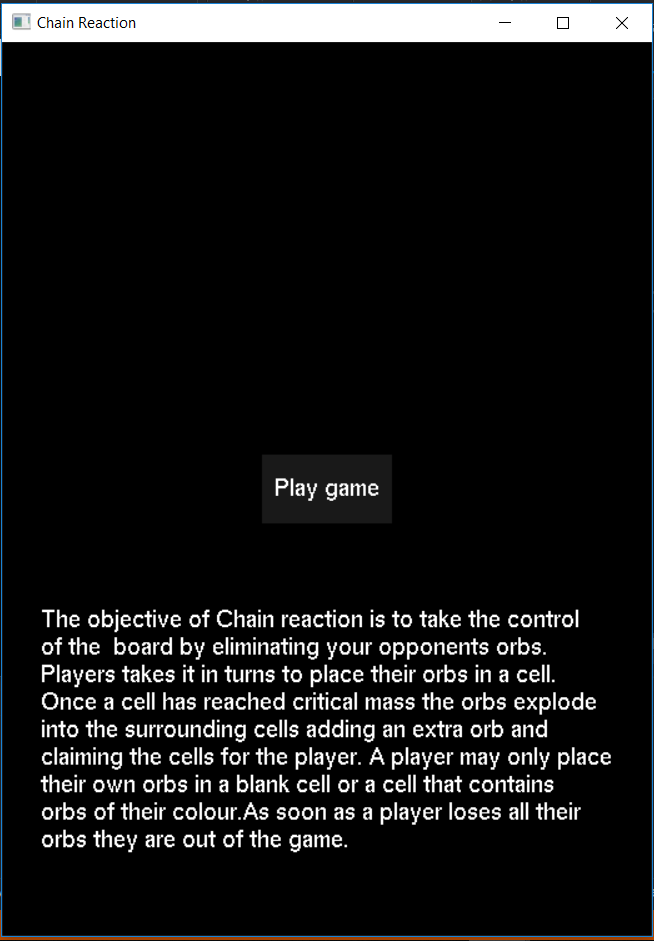
{

printf("Player wins\n");

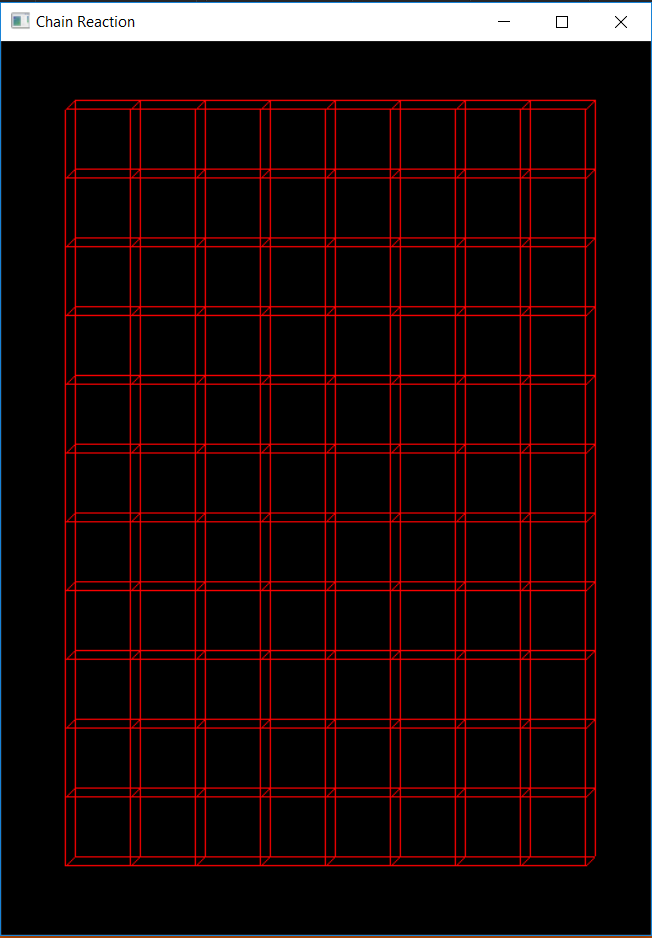
**CHAPTER 5**

**SCREENSHOTS**

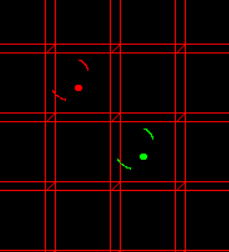
PICTURE 5.1: OPENING SCREEN



PICTURE 5.2: PLAY AREA (GRID)



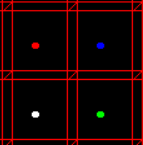
PICTURE 5.3: ORBS ROTATING



PICTURE 5.4 : END GAME SCREEN



PICTURE 5.5 : MULTIPLAYER ORBS



**CHAPTER 6**

**CONCLUSION AND FUTURE SCOPE**

This is to conclude that the project we undertook has been worked upon with sincere effort and has been completed successfully. The requirements and goals of the project have been achieved. The desktop application has been thoroughly tested and can now players can play in the real world. By this project I hope to bring satisfaction to the users and meet their expectations.

**BIBILOGRAPHY**

[www.youtube.com](http://www.youtube.com)

[www.stackoverflow.com](http://www.stackoverflow.com)

[www.geekforgeeks.com](http://www.geekforgeeks.com)

Computer Graphics textbook