Brick Breaker Game

Mini Project Report

Submitted in partial fulfillment of the requirements

For the degree of

Bachelor of Engineering (Computer Engineering)

by:

Atharva S. Birje. [C-26]	Student ID –TU3F2122158
Shivam B. Daki. [C-30]	Student ID -TU3F2122163
Harsh A. Minde. [C-31]	Student ID -TU3F2122164

Under the Guidance of

PROF. KIRTI SURYAWANSHI.

Department of Computer Engineering

TERNA ENGINEERING COLLEGE

Nerul (W), Navi Mumbai 400706

(University of Mumbai)

(2022-2023)

Internal Approval Sheet

TERNA ENGINEERING COLLEGE, NERUL

Department of Computer Engineering

Academic Year 2022-2023

CERTIFICATE

This is to certify that project entitled "Brick Breaker Game" is a bonafide work of:

Atharva S. Birje. [C-26]	Student ID –TU3F2122158
Shivam B. Daki. [C-30]	Student ID -TU3F2122163
Harsh A. Minde. [C-31]	Student ID -TU3F2122164

Submitted to the University of Mumbai in partial fulfilment of the requirement for the award of the Bachelor of Engineering (Computer Engineering).

Guide **Head of Department** **Principa**

Table of Contents

Chapter 1

Introduction

1.1 Motivation :

Games are a fundamental way that humans interact and learn. They provide so many advantages for people of all interests and abilities.

The main benefits of using computer games as learning tools, is related with problem solving, 21stcentury skills, integration of learning and assessment, collaboratives and interactivity, addressing cognitive as well as affective learning issues, and motivation for learning. Gaming is gaining a different level of attention not only from the youngsters but also from different age people.

It is creating a virtual world where we can virtually live our life. Gaming has gained importance in Desktop application as well as in Android Application.

1.2 Need Of The Problem :

As a computer science student, we’ve to learn game development because this field requires some skills that already we’ve and at the same time we can enhance them.

For a ‘game design’ it can mean “emotional engineering” or “largely communication” whilst for the other “everything that goes into a game is more or less game design”. On a more formal matter, many designers have gone extra miles to explicate their conceptions on game design.

Game development can considerably improve our imagination skills, designing skills and the ability to produce new things. For a game to attract attention, it should also be unique. The more the game is realistic, the more appealing and charming it i

Chapter 2

Software And Hardware Requirements

1. OS -Windows 11.
2. Processor –Any Pentium Version.
3. Hard Disk- 32 GB (64 GB recommended).
4. Memory - 4GB RAM for 64 bit.
5. C/C++ Compiler.
6. Cod::Blocks IDE
7. Open GL Libraries
8. Some Graphics Files.

Chapter 3

Flowchart

- Process of the program in detail:

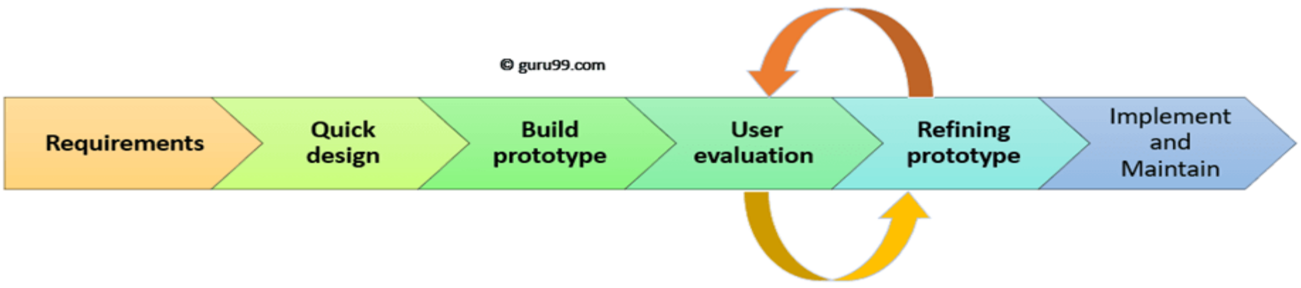


Fig: 3.1.1

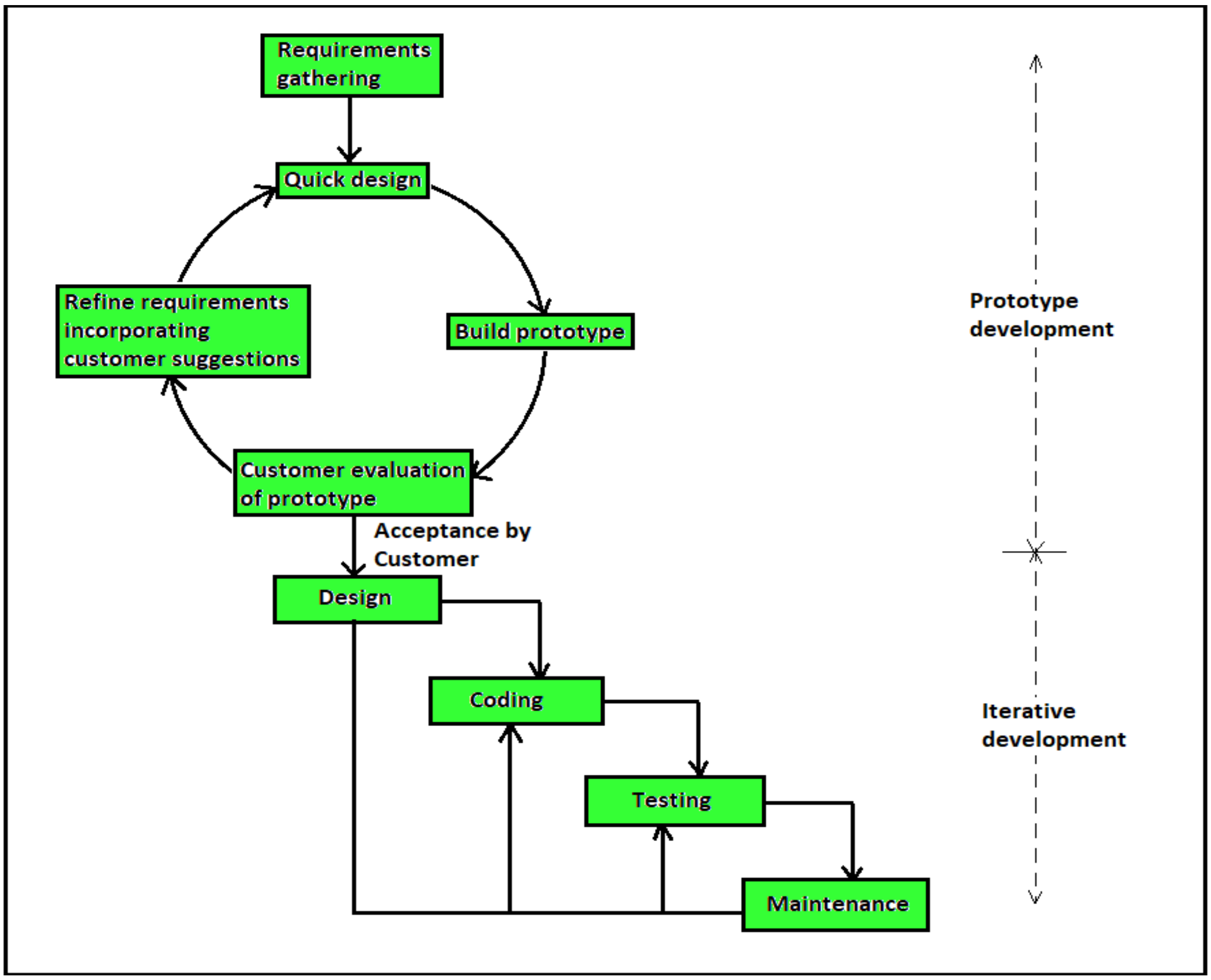


Fig: 3.1.2

Chapter 4

Functions Used

Function	Description
draw_paddle():	A function which is used to draw paddle using the inbuilt polygon function and defining the appropriate vertices.
brick(GLfloat x, GLfloat y, GLfloat z):	A function which is used as the basis for the draw_bricks() function. This is used to draw a single brick.
draw_bricks():	A function which is used to call multiple instances of the brick() and set up the environment for the game.
draw_ball(GLflo at x, GLfloat y):	This function is used to draw the ball according to its current position.
addMenu():	A function which adds the menu interface and layout to the display area when triggered when won or loss.
text():	A function which displays the instructions on how to interact with the game.
display(void):	This is the main function that calls all other functions to present the elements on screen in each frame.
reshape(int w, int h):	A function which adjusts the layout of all elements onto the display area through projection.
keyboard(unsig ned char key, int x, int y):	A function which scans input given. This is done by specifying the role of the keys in the switch case and processing input.
hit():	This is one of the main functions of our game which comes into play when the ball hits the brick. This function defines the trajectory of the ball after hitting. We used the concept by applying the if else conditions and accordingly changing the direction of the ball.
idle():	This is also one of the main functions of our game which handles the motion of the ball along with the rebounding from various surfaces. Here one condition is for increasing the rate after some successive collisions and another condition is to make the changes for the different positions of the ball after rebounding.

All the Computer Graphics concepts that are implemented are listed below:

● **Translation**: Concept of translation is used in our game while moving the paddle as well as while the ball is in motion rebounding from the different surfaces.

● **Scaling**: Concept of scaling can be seen in our game when the difficulty changes, the scaling of the paddle differs. When the level is easy we scale the paddle smaller and when the level is hard we scale the paddle longer

● **Drawing Polygons**: Different shapes were drawn using the inbuilt polygon function and defining the appropriate vertices which matches the shape of the polygon.

● **Game Concept**: Brick Breaker is a famous retro game where we have to destroy all the bricks and the ball should not fall. If the ball falls the game will restart. The main player of the game is paddle, we have to save the ball from going below by the paddle and at the same time try to hit as many bricks as possible.

● **Keyboard Inputs**: Now for playing the game the user needs some kind of input to give. This can be done very easily by using a glut function 'keyboard' and then by specifying the role of the keys in the switch case. In each switch case you need to define what that key will do.

● **Hitting the Brick**: This is the concept which comes into play when the ball hits the brick. This function defines the trajectory of the ball after hitting. We used the concept by applying the if else conditions and accordingly changing the direction of the ball.

● **Trajectory of the Ball**: The concept used for this particular feature of our game which handles the motion of the ball along with the rebounding from various surfaces. Here one condition is for increasing the rate after some successive collisions and another condition is to make the changes for the different directions of the ball after rebounding.

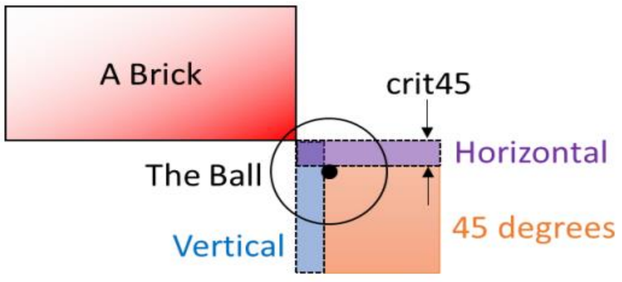


Fig: 4.1

Chapter 5

Output Primitives And Transformations

• Output Primitives:

Output Primitives are simple geometric functions that are used to generate various Computer Graphics required by the user. Some most basic Output Primitives include pixels (points) and straight lines. For this project, we have used the following Output Primitives:

OUTPUT PRIMITIVES	APPLICATIONS
Line	For Walls
Rectangle	For Bricks
Square	For Ball

• Filled Area Primitive:

Filled area primitives are used to give color to a specific region of the graphics scene.
For this project, the flood-fill approach was used to give color to various components of game. Then four connected approaches or eight connected approaches areused to fill with specified color.
The flood fill algorithm has many characters similar to boundary fill. But this flood fill method ismore suitable for filling multiple color boundaries of brick. When boundary is of all bricks is to be filled with one color we use this algorithm i.e. Blue.

• 3D Transformations:

Transformation means changing some graphics into something else by applying rules. We can have various types of transformations such as translation, scaling up or down, rotation, shearing, etc. When a transformation takes place on a 3D plane, it is called 3D transformation.
For this project, we have used concepts of :
Translation : To move the paddle on ground level.
Scaling : To move a ball up or down

Chapter 6

Program

Input Code:

```
#include <stdlib.h>
#include <GL/glut.h>
#include<stdio.h>
#include<string.h>
#include <math.h>


int ball_color = 3, level = 0, size = 1;;
GLfloat twoModel[] = { GL_TRUE };
int game_level[] = { 7,5,2 };
float rate = game_level[level];

GLfloat paddle_size[] = { 2,4,6 };
GLfloat brick_color_array[][3] = { {0,0.25,1} };
GLfloat paddle_color_array[][3] = { {1,1,0} };
GLfloat ball_color_array[][3] = { {1,0,1} };
GLfloat text_color_array[][4] = { {1,0,0} };

//The grid parameters for the bricks
int rows = 4;
int columns = 10;

// Structure to store the coordinates of each brick
struct brick_coords {

        GLfloat x;
        GLfloat y;
};

//Array to store the bricks
brick_coords brick_array[50][50];
GLfloat px, bx = 0, by = -12.8, speed = 0, dirx = 0, diry = 0, start = 0;
```

```
// Function to draw the paddle
void draw_paddle()
{
        glColor3fv(paddle_color_array[0]);
        glBegin(GL_POLYGON);
        glVertex3f(-paddle_size[size] + px, 0 - 15, 0);
        glVertex3f(paddle_size[size] + px, 0 - 15, 0);
        glVertex3f(paddle_size[size] + px, 1 - 15, 0);
        glVertex3f(-paddle_size[size] + px, 1 - 15, 0);
        glEnd();
}


//Function to draw a single brick
void brick(GLfloat x, GLfloat y, GLfloat z)
{
        glColor3fv(brick_color_array[0]);
        glBegin(GL_QUADS);
        glVertex3f(x, y, z);
        glVertex3f(x + 3, y, z);
        glVertex3f(x + 3, y + 1, z);
        glVertex3f(x, y + 1, z);
        glEnd();
}


// Function to draw the grid of bricks
void draw_bricks()
{
        int i, j;
        if (start == 0)
        {
                for (i = 1; i <= rows; i++)
                {
                        for (j = 1; j <= columns; j++)
                        {

                                brick_array[i][j].x = (GLfloat)(j * 4 * 0.84);
                                brick_array[i][j].y = (GLfloat)(i * 2 * 0.6);
                        }
                }
        }
        glPushMatrix();
        glTranslatef(-19.5, 5, 0);

        for (i = 1; i <= rows; i += 1)
        {
                for (j = 1; j <= columns; j += 1)
                {

                        if (brick_array[i][j].x == 0 || brick_array[i][j].y == 0)
                        {
                                continue;
                        }
                        brick(brick_array[i][j].x, brick_array[i][j].y, 0);
                }
        }
        glPopMatrix();
}

//Function to draw the spherical ball
void draw_ball(GLfloat x, GLfloat y, GLfloat radius)
{
        int i;
        int triangleAmount = 20; //# of triangles used to draw circle

        GLfloat twicePi = 2.0f * 3.14;

        glColor3fv(ball_color_array[0]);
        glBegin(GL_TRIANGLE_FAN);
        glVertex2f(x, y); // center of circle
        for (i = 0; i <= triangleAmount; i++) {
                glVertex2f(
                        x + (radius * cos(i * twicePi / triangleAmount)),
                        y + (radius * sin(i * twicePi / triangleAmount))
                );
        }
        glEnd();

}

//handle level
void change_difficulty(int action)
{
        level = action - 1;
}

//handle menu
void handle_menu(int action)
{

}
```

```
//handle paddle size
void change_paddle_size(int action)
{
	size = action - 1;
}

//add menu
void addMenu()
{

	int submenu1 = glutCreateMenu(change_difficulty);
	glutAddMenuEntry("Level-1", 1);
	glutAddMenuEntry("Level-2", 2);
	glutAddMenuEntry("Level-3", 3);

	int submenu2 = glutCreateMenu(change_paddle_size);
	glutAddMenuEntry("Small", 1);
	glutAddMenuEntry("Medium", 2);
	glutAddMenuEntry("Large", 3);

	glutCreateMenu(handle_menu);
	glutAddSubMenu("Difficulty", submenu1);
	glutAddSubMenu("Paddle Size", submenu2);
	glutAttachMenu(GLUT_RIGHT_BUTTON);

}

void text()
{
	char text[40] = "Press 's' to release the ball";
	char text1[40] = "Press 'a' to move the paddle left";
	char text2[40] = "Press 'd' to move the paddle right";
	char text3[40] = "Press 'q' to close the window";
	// The color
	glColor3fv(text_color_array[0]);
	// Position of the text to be printer
	glPushMatrix();
	glTranslatef(-1, 0, 0);
	glRasterPos3f(0.2, 0, 20);
	for (int i = 0; text[i] != '\0'; i++)
		glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, text[i]);
	glRasterPos3f(0.2, -0.15, 20);
	for (int i = 0; text1[i] != '\0'; i++)
		glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, text1[i]);
	glRasterPos3f(0.2, -0.30, 20);
	for (int i = 0; text2[i] != '\0'; i++)
		glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, text2[i]);
	glRasterPos3f(0.2, -0.45, 20);
	for (int i = 0; text3[i] != '\0'; i++)
		glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, text3[i]);
	glPopMatrix();

}
```

```
//The main display function
void display(void) {

            glClearColor(0.0, 0.0, 0.0, 1.0);
            glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
            glLoadIdentity();
            gluLookAt(0, 0, 0, 0, 0, -25, 0, 1, 0);
            glTranslatef(0, 0, -25);
            draw_paddle();
            draw_bricks();
            draw_ball(bx, by, 1.0);
            text();
            glutSwapBuffers();
}

void reshape(int w, int h) {
            glViewport(0, 0, (GLsizei)w, (GLsizei)h);
            glMatrixMode(GL_PROJECTION);
            glLoadIdentity();
            gluPerspective(60, (GLfloat)w / (GLfloat)h, 1.0, 1000.0);
            glMatrixMode(GL_MODELVIEW);
}


//function to take in keyboard entries
void keyboard(unsigned char key, int x, int y)
{
            switch (key)
            {
            case 'd': px += 3; break;
            case 'a': px -= 3; break;
            case 'q': exit(0); break;
            case 's':
                        if (!start)
                        {
                                    dirx = diry = 1;
                                    rate = game_level[level];
                                    start = 1;
                                    glutSetCursor(GLUT_CURSOR_NONE);

                        }
                        break;
            }
```

```
	if (px > 15)
	{
		px = 15;
	}
	if (px < -15)
	{
		px = -15;
	}
	if (start == 0)
	{
		px = 0;
	}
	glutPostRedisplay();
}


//Function to handle the case when the ball strikes the bricks
void hit()
{
	int i, j;
	for (i = 1; i <= rows; i++)
		for (j = 1; j <= columns; j++)
		{
			if ((bx >= brick_array[i][j].x - 19.5 - 0.1) && (bx <= brick_array[i][j].x + 3 -
19.5 + 0.1))
			{
				if (by >= brick_array[i][j].y + 5 - 0.1 && by <= brick_array[i][j].y + 5 +
1.2 + 0.1)
				{
					brick_array[i][j].x = 0;
					brick_array[i][j].y = 0;
					diry = diry * -1;
				}
				//cout<<bx<<" "<<by<<"\t"<<brick_array[i][j].x<<"
"<<brick_array[i][j].y;
			}
			else if (by >= brick_array[i][j].y + 5 - 0.1 && by <= brick_array[i][j].y + 5 +
1.2 + 0.1)
			{
				if ((bx >= brick_array[i][j].x - 19.5 - 0.1) && (bx <= brick_array[i][j].x
+ 3 - 19.5 + 0.1))
				{
					brick_array[i][j].x = 0;
					brick_array[i][j].y = 0;
					dirx = dirx * -1;
				}
				//cout<<bx<<" "<<by<<"\t"<<brick_array[i][j].x<<"
"<<brick_array[i][j].y;
			}
		}
}
```

```
//The idle function. Handles the motion of the ball along with rebounding from various surfaces
void idle()
{
	hit();
	if (bx < -16 || bx>16 && start == 1) // Reflection from right left wall
	{
		dirx = dirx * -1;
	}
	if (by < -15 || by>14 && start == 1)  // Reflection from top bottom wall
	{
		diry = diry * -1;
	}
	bx += dirx / (rate);
	by += diry / (rate);
	rate -= 0.001; // Rate at which the speed of ball increases

	float x = paddle_size[size];
	//Make changes here for the different position of ball after rebounded by paddle
	if (by <= -12.8 && bx<(px + x * 2 / 3) && bx>(px + x / 3) && start == 1)
	{
		dirx = 1;
		diry = 1;
	}
	else if (by <= -12.8 && bx<(px - x / 3) && bx>(px - x * 2 / 3) && start == 1)
	{
		dirx = -1;
		diry = 1;
	}
	else if (by <= -12.8 && bx<(px + x / 3) && bx>(px - x / 3) && start == 1)
	{
		dirx = dirx;
		diry = 1;
	}
	else if (by <= -12.8 && bx<(px - (x * 2 / 3)) && bx>(px - (x + 0.3)) && start == 1)
	{
		dirx = -1.5;
		diry = 0.8;
	}
	else if (by <= -12.8 && bx<(px + (x + 0.3)) && bx>(px + x / 3) && start == 1)
	{
		dirx = 1.5;
		diry = 0.8;
	}
	else if (by < -13)
	{
		start = 0;
		by = -12.8;
		bx = 0;
		dirx = 0;
		diry = 0;
		px = 0;
	}
	glutPostRedisplay();
}
```

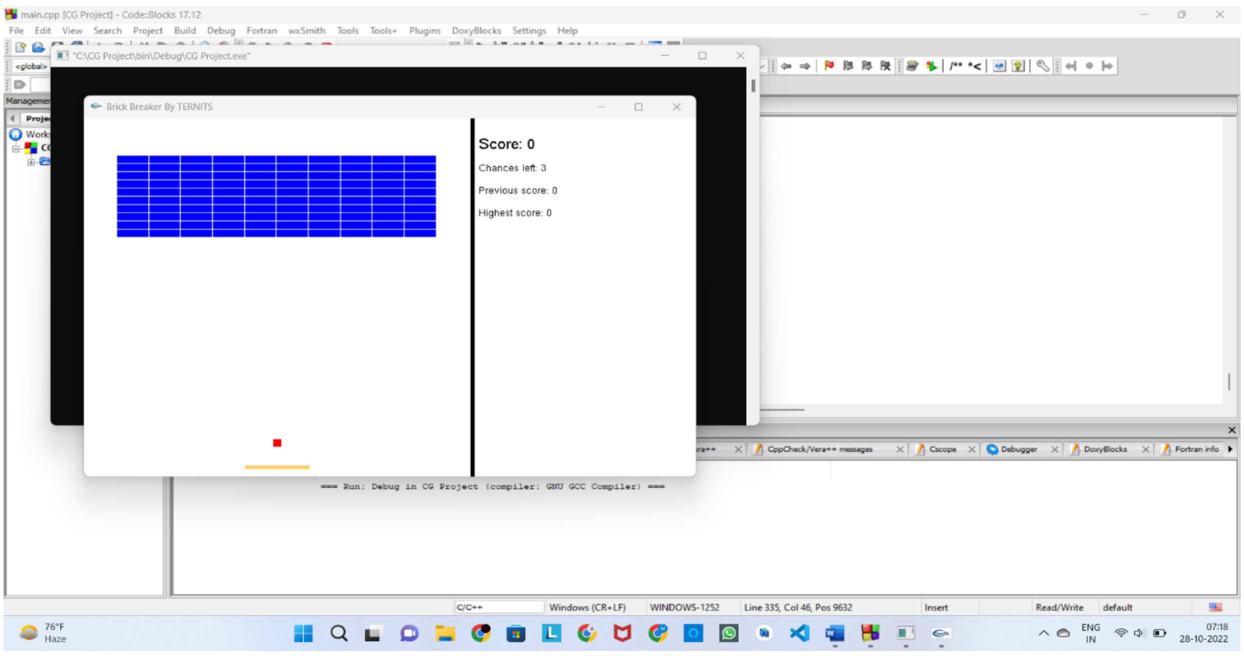
```
int main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DOUBLE | GLUT_DEPTH);
	glutInitWindowSize(1100, 900);
	glutInitWindowPosition(100, 100);
	glutCreateWindow("Brick Breaker");
	glutDisplayFunc(display);
	glutReshapeFunc(reshape);
	glEnable(GL_DEPTH_TEST);
	glutIdleFunc(idle);
	glutKeyboardFunc(keyboard);
	addMenu();

	glutMainLoop();
	return 0;
}
```

Chapter 7

Output Snapshots

1) Initial view of Game:

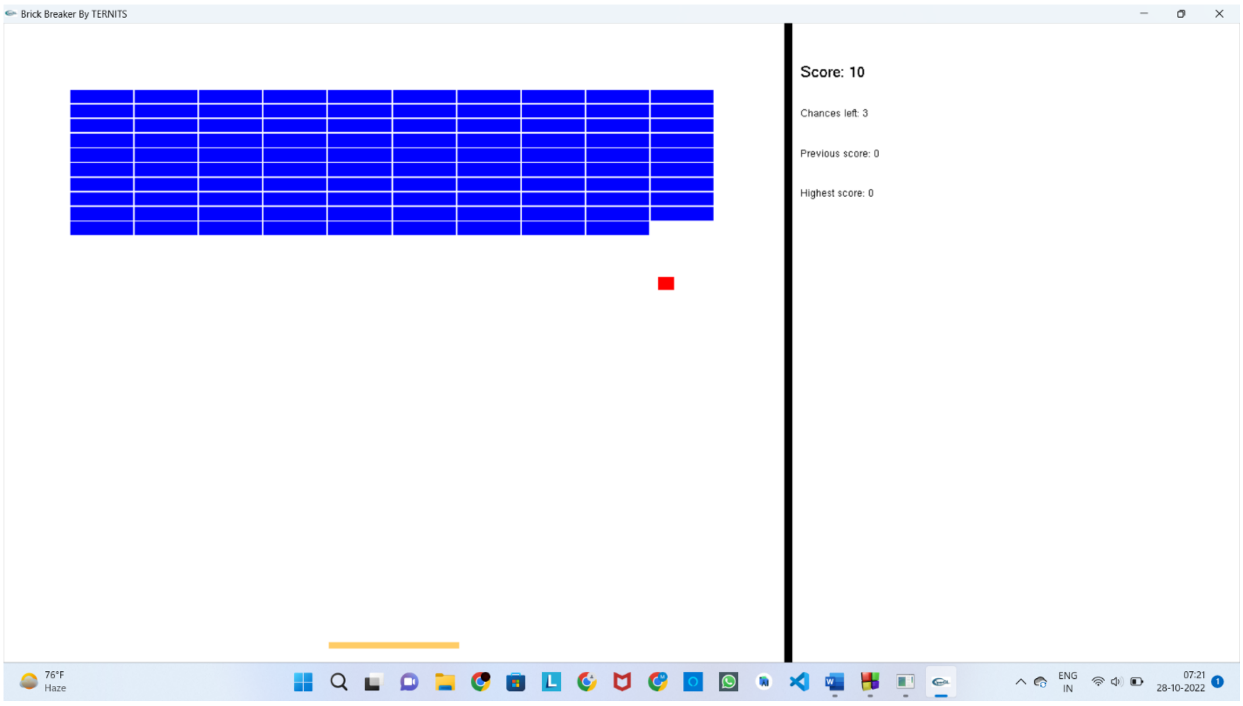

main.cpp [CG Project] - Code::Blocks 17.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
"C:\CG Project\bin\Debug\CG Project.exe"
Brick Breaker By TERNITS
CppCheck/Vera++ messages
Cscope
Debugger
DoxyBlocks
Fortran info
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 335, Col 46, Pos 9632
Insert
Read/Write
default
76°F
Haze
ENG
IN
07:18
28-10-2022

```
Score: 0
Chances left: 3
Previous score: 0
Highest score: 0
```

```
=== Run: Debug in CG Project (compiler: GNU GCC Compiler) ===
```

Fig: 7.1

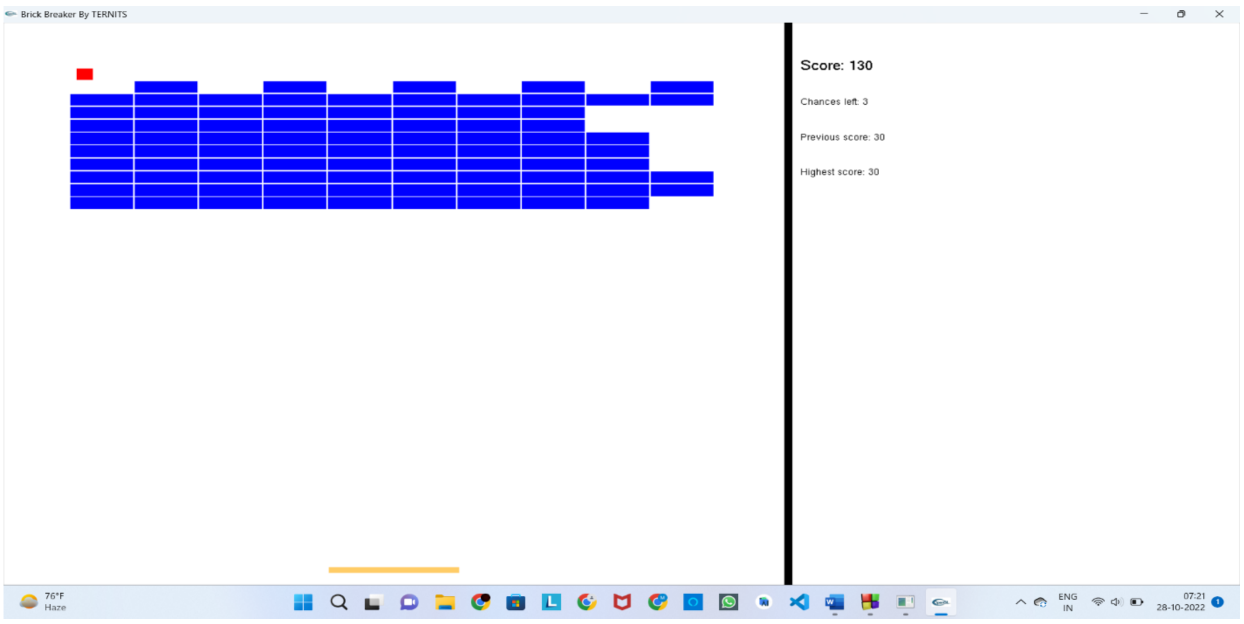
2) When user tap to start:



```
Score: 10
Chances left: 3
Previous score: 0
Highest score: 0
```

Fig: 7.2

3) When user playing:



Score: 130

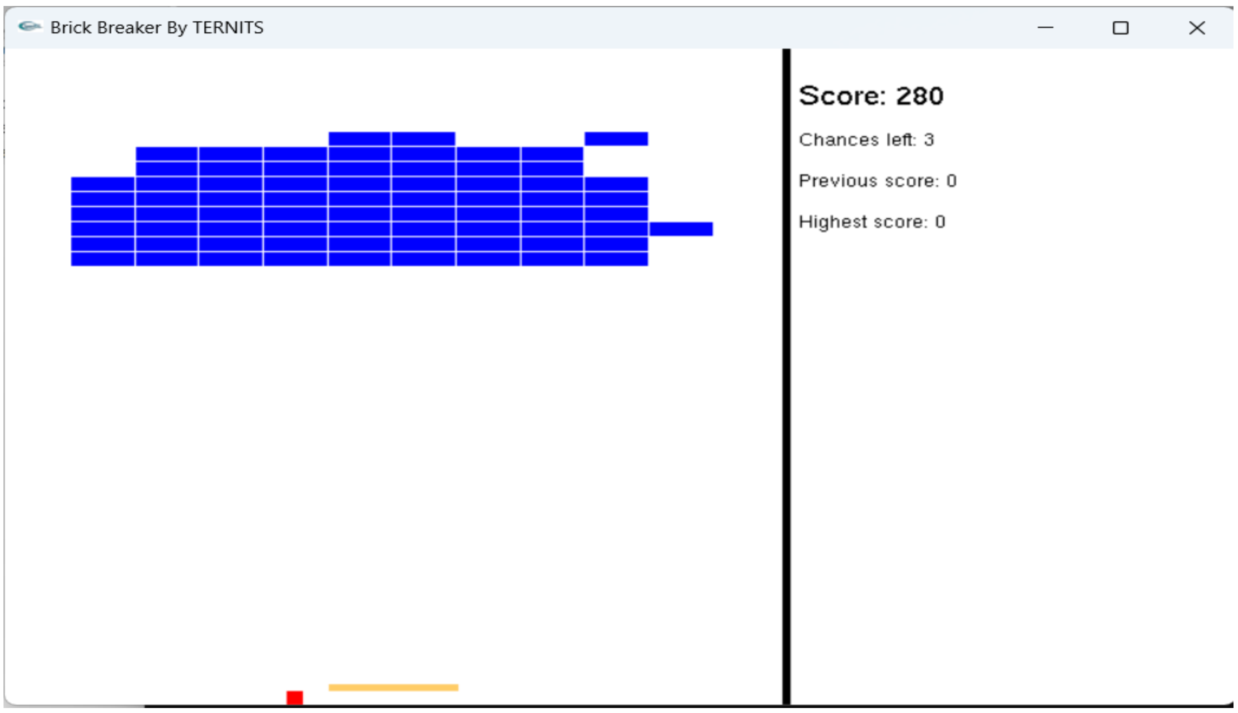
Chances left: 3

Previous score: 30

Highest score: 30

Fig: 7.3

4) When user missed the ball first time:



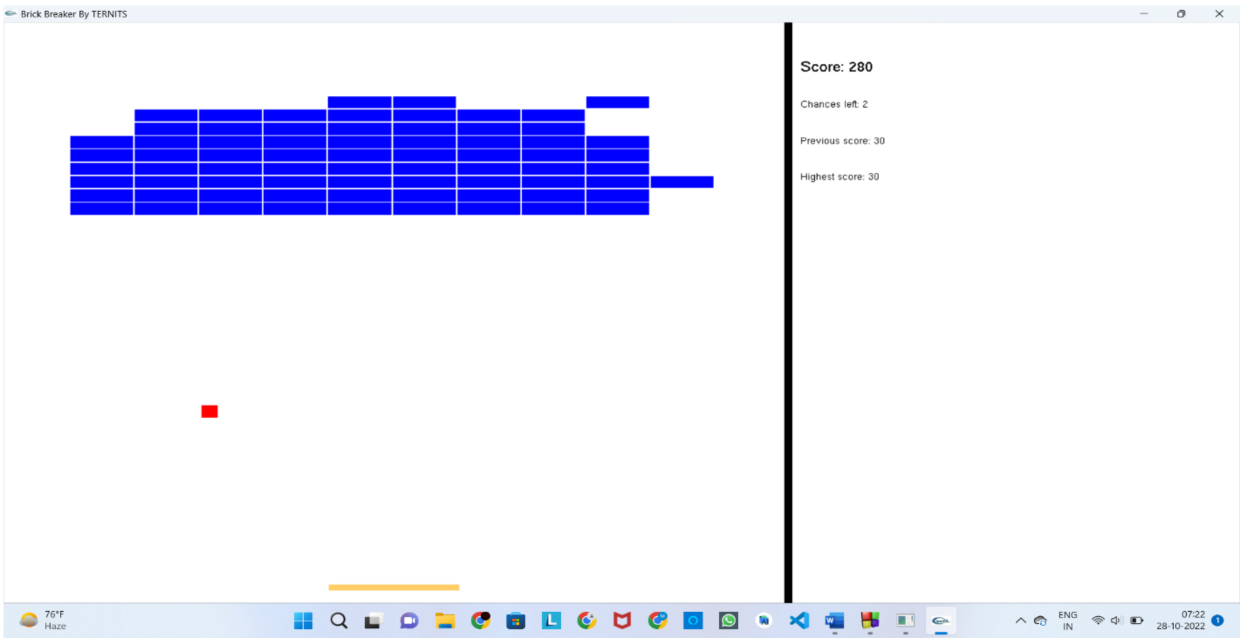
Score: 280

Chances left: 3

Previous score: 0

Highest score: 0

Fig: 7.4



Score: 280

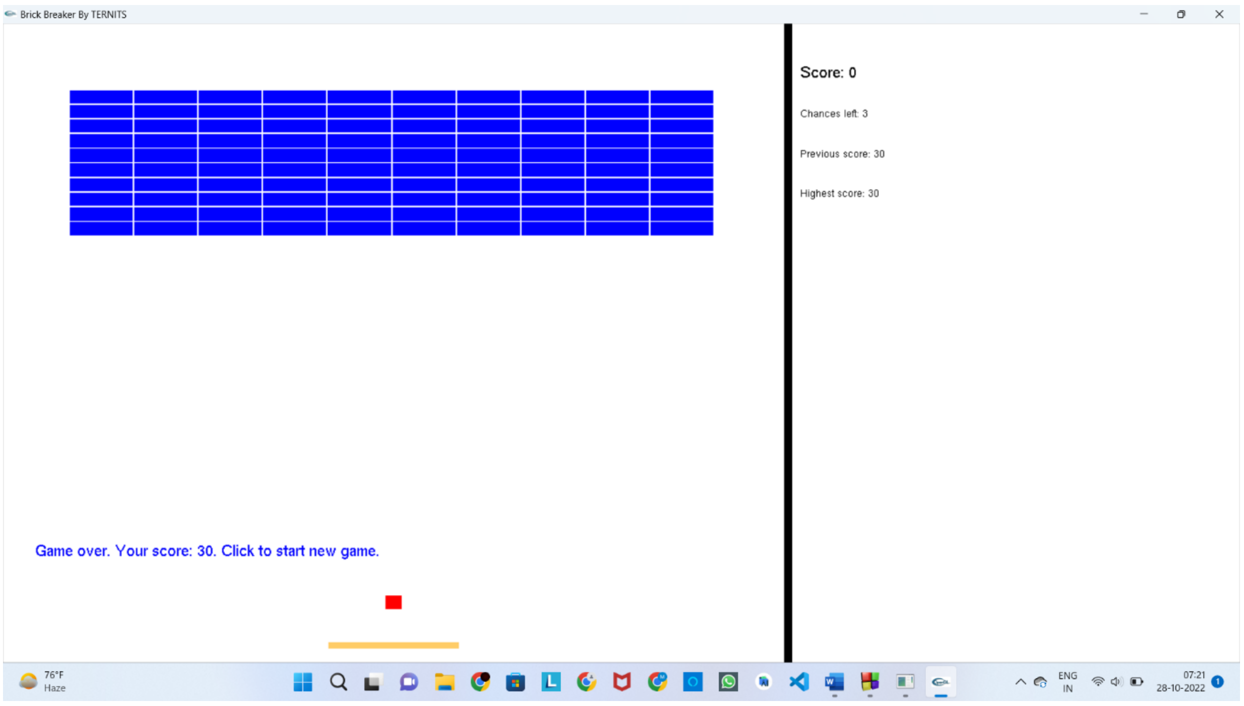
Chances left: 2

Previous score: 30

Highest score: 30

Fig: 7.5

5) When user used all his life chances:



Score: 0

Chances left: 3

Previous score: 30

Highest score: 30

Game over. Your score: 30. Click to start new game.

Fig: 7.6

Chapter 8

Conclusion

- Thus, we were able to implement the code of Animated 2D game using C++ language (with graphics.h, dos.h and process.h header files).
- The project is well suited for designing 2D objects, as well as for carrying outbasic graphics functionalities.
- However, if implemented on professional level with sufficient resources, it has the potentialto become a standard stand-alone GUI based application for fun for Operating System.
- The major aim of this project is to increase the problem-solving skills. The project aims to create a 2D game in pc is successful . Where users can access the game with ease. In this we learned how computer graphics is used in game development and in other applications. We have discussed about how brick breakout game is developed in C++.
- So, concluding our project, the Brick Breaker Game demonstrated our learning and new found expertise in Verilog coding, the understanding of basic CodeBlocks IDE, basic C++ Programming Language, Teamwork & all, How to handle a real life project, And many more things; Also last but not the least i.e. solution for our aim of the project.

Chapter 9

Reference

- TEC, University of Mumbai, Mumbai, India
  “PC Based 2D Game Design”, May-June-2021

- Stanley College of Engineering and Technology for Women, Hyderabad,
  “GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES, BRICK BREAKOUT”, 2018

- http://www.cs.columbia.edu/~sedwards/classes/2019/4840-spring/designs/BrickBreaker.pdf

- http://web.mit.edu/6.111/www/f2013/projects/jabbott_Project_Final_Report.pdf

- LinkedIn
- Github
- Youtube

THANK YOU !