Computer Graphics (UCS505)

Project on: Brick Breaker Game

Submitted by:

Bhavya Kakwani	101903365
Aganya Bajaj	101903360
Ananya Agarwal	102083036

Group No: 2

B.E. Third Year – COE

Submitted To:

Dr. Harpreet Singh

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology

Patiala – 147001

Table of Contents

1. Introduction to Project

Brick Breaker is a Breakout clone which the player must smash a wall of bricks by deflecting a bouncing ball with a paddle. The paddle may move horizontally and is controlled with the keyboard. There are many versions of brick breaker, some in which you can shoot flaming fireballs or play with more than one ball if the player gets a power up.

OpenGL (Open Graphics Library) is the computer industry's standard application program interface (API) for defining 2-D and 3-D graphic images. It specifies a set of "commands" or immediately executed functions. Each command directs a drawing action or causes special effects. A list of these commands can be created for repetitive effects. It provides special "glue" routines for each operating system that enable OpenGL to work in that system's windowing environment. One of the most important features of OpenGL is viewing and modelling transformations(3D/2D).

Our project uses OpenGL to remake the iconic Brick Breaker game.

2. Computer Graphics Concepts Used

1) Translation movng one objct from 1 poston to another
2) MatrixMode
3) Various Shape Drawing functions such as Quadrilaterals, Spheres & Polygons.
4) BitMapCharacter for displaying text

OpenGL keeps a stack of matrices to quickly apply and remove transformations. glPushMatrix copies the top matrix and pushes it onto the stack, while glPopMatrix pops the top matrix off the stack

3. User Defined Functions

1) **void draw_paddle():** Function to draw the paddle.

2) **void brick(GLfloat x,GLfloat y, GLfloat z):** Function to draw a single brick.

3) **void draw_bricks():** Function to draw the grid of bricks.

4) **void draw_ball():** Function to draw the spherical ball.

5) **void text(int sc):** Function to print the score on the screen.

6) **void display():** The main display function.

7) **void keyboard(unsigned char key, int x, int y):** Function to take in keyboard entries.

8) **void hit():** Function to handle the case when the ball strikes the bricks.

9) **void idle():** Function that handles the motion of the ball along with rebounding from various surfaces.

4. Code

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

int score = 0;
int level = 0, size = 1;

int game_level[] = {35, 25, 15};
float rate = game_level[level];
GLfloat paddle_size[] = {2, 4, 6};

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

// Structure to store the co-ordinates 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, dirx = 0, diry = 0, start = 0;

// Function to draw the paddle
void draw_paddle()
{
        glColor3f(0,1,0);
        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)
{
        glColor3f(1, 0, 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++)
	{
		for(j=1; j<=columns; j++)
		{
			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()
{
	glPushMatrix();
	glColor3f(1, 0.94, 0.12);
	glTranslatef(bx, by, 0);
	glutSolidSphere(1.0, 200, 200);
	glPopMatrix();
}

//Function to print the score on the screen
void text(int sc)
{
	char text[40];
	char difficulty[10];
	if(level == 0)
	{
		sprintf(difficulty,"Easy");
	}

	if(level == 1)
	{
		sprintf(difficulty,"Medium");
	}
```

```
		if(level == 2)
		{
			sprintf(difficulty,"Hard");
		}
		if(sc < 40)
	sprintf(text,"Difficulty: %s    Your Score: %d",difficulty, sc);
		else
		{
	sprintf(text,"Congratulations! You won!");
	start = 0;
	by = -12.8;
	bx = 0;
	dirx = 0;
	diry = 0;
	px = 0;
		}

		// The color
		glColor4f(0,1,1,1);

		// Position of the text to be printer
		glPushMatrix();

		glTranslatef(-1,0,0);
		glRasterPos3f(0, 0, 20);

		for(int i = 0; text[i] != '\0'; i++)
			glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, text[i]);

		glPopMatrix();
}

//The main display function
void display()
{
		glClearColor(0.0, 0.0, 0.0, 1.0);
		glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
		glLoadIdentity();
		glTranslatef(0,0,-25);
		draw_paddle();
		draw_bricks();
		draw_ball();
		text(score);
		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':
        case 'D':
                    px += 7;
                    break;
                case 'a':
        case 'A':
                    px -= 7;
                    break;
                case 'q':
        case 'Q':
                    exit(0);
                case 's':
        case 'S':
           if(!start)
           {
              dirx = diry = 1;
              rate = game_level[level];
              start = 1;
              score = 0;
           }
           break;
             }

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

//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;
                                        score++;
                                }
                        }
                        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;
                                        score++;
                                }
                        }
                }
}

//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)
        {
                dirx = dirx * -1;
        }
        if(by < -15 || by > 14 && start == 1)
        {
                diry = diry * -1;
        }

        bx += dirx / (rate);
        by += diry / (rate);
        rate -= 0.0001; // 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_SINGLE);
screen size     glutInitWindowSize(1500, 750);
                glutInitWindowPosition(0, 0);
                glutCreateWindow ("Brick Breaker");
                glutDisplayFunc(display);
                glutReshapeFunc(reshape);
                glutIdleFunc(idle);
                glutKeyboardFunc(keyboard);
                glutMainLoop();
                return 0;
}
```

initlztn of display image and codnte system

buffer pr code store nahi hota, dsply pr show hota ha drctly

tells the wndow system whoch func we want to process

the nrml key events

strt the event loop,1 baari call krna must, atlast rkho kyuki trmte nahi hota

reshape: Not only does it set the size of the OpenGL context to the size of the window it is in, but it makes sure everything is perspectively correct.

func idle: so a GLUT program can perform background processing tasks or continuous animation when window system events are not being received

5. Output/ Screenshots