Experiment No. 10
Program for Animation
Name: AARYAN CHANDRAKANT GOLE
Roll Number: 12
Date of Performance:
Date of Submission:

CSL305: Computer Graphics Lab

Experiment No. 10

Aim: To develop programs for making animations.

Objective:

Draw an object and apply various transformation techniques to this object. Translation, scaling and rotation is applied to object to perform animation.

Theory:

- For moving any object, we incrementally calculate the object coordinates and redraw the picture to give a feel of animation by using for loop.
- Suppose if we want to move a circle from left to right means, we have to shift the position of circle along x-direction continuously in regular intervals.
- The below programs illustrate the movement of objects by using for loop and also using transformations like rotation, translation etc.
- For windmill rotation, we use 2D rotation concept and formulas.

Program:

```
package brickBracker;
import javax.swing.JFrame;
public class Main {
public static void main(String[] args) {
JFrame obj = new JFrame();
Gameplay gamePlay = new Gameplay();
obj.setBounds(10, 10, 700, 600);
obj.setTitle("Breakout Ball");
obj.setResizable(false);
obj.setVisible(true);
obj.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
obj.add(gamePlay);
}
}
package brickBracker;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
```

```java
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import javax.swing.JPanel;
import javax.swing.Timer;
public class Gameplay extends JPanel implements KeyListener, ActionListener {
private boolean play = false;
private int score = 0;
private int totalBricks = 21;
private Timer timer;
private int delay = 15;
private int playerX = 310;
private int ballposX = 120;
private int ballposY = 350;
private int ballXdir = -2;
private int ballYdir = -1;
private MapGenerator map;
public Gameplay() {
map = new MapGenerator(3, 7);
addKeyListener(this);
setFocusable(true);
setFocusTraversalKeysEnabled(false);
timer = new Timer(delay, this);
timer.start();
} public void paint(Graphics g) {
super.paint(g);
Background g.setColor(Color.black);
g.fillRect(1, 1, 692, 592);
Drawing map map.draw((Graphics2D) g);
Borders g.setColor(Color.yellow);
g.fillRect(0, 0, 3, 592);
g.fillRect(0, 0, 692, 3);
g.fillRect(691, 0, 3, 592);
g.setColor(Color.white);
g.setFont(new Font("serif", Font.BOLD, 25));
g.drawString("" + score, 590, 30);
g.setColor(Color.green);
g.fillRect(playerX, 550, 100, 8);
g.setColor(Color.yellow);
g.fillOval(ballposX, ballposY, 20, 20);
if (totalBricks <= 0) {
play = false;
```

```
ballXdir = 0;
ballYdir = 0;
g.setColor(Color.RED);
g.setFont(new Font("serif", Font.BOLD, 30));
g.drawString("YOU WON", 260, 300); }
if (ballposY > 570) {
play = false;
ballXdir = 0;
ballYdir = 0;
g.setColor(Color.RED);
g.setFont(new Font("serif", Font.BOLD, 30));
g.drawString("Game over, Scores: " + score, 190, 300);
g.setFont(new Font("serif", Font.BOLD, 20));
g.drawString("Press Enter to Restart", 230, 350);
}
}
public void actionPerformed(ActionEvent e) {
timer.start();
if (play) {
if (new Rectangle(ballposX, ballposY, 20, 20).intersects(new Rectangle(playerX, 550, 100,
8))){
ballYdir = -ballYdir;
}
. for (int i = 0; i < map.map.length; i++) {
for (int j = 0; j < map.map[0].length; j++) {
if (map.map[i][j] > 0) {
int brickX = j * map.brickWidth + 80;
int brickY = i * map.brickHeight + 50;
int brickWidth = map.brickWidth;
int brickHeight = map.brickHeight;
Rectangle rect = new Rectangle(brickX, brickY, brickWidth, brickHeight);
Rectangle ballRect = new Rectangle(ballposX, ballposY, 20, 20);
Rectangle brickRect = rect;
if (ballRect.intersects(brickRect)) {
map.setBrickValue(0, i, j);
totalBricks--;
score += 5;
if (ballposX + 19 <= brickRect.x || ballposX + 1 >= brickRect.x + brickRect.width) {
ballXdir = -ballXdir;
} else {
ballYdir = -ballYdir;
}
break A;
}
```

```
}
}
}
ballposX += ballXdir;
ballposY += ballYdir;
if (ballposX < 0) {
ballXdir = -ballXdir;
} if (ballposY < 0) {
ballYdir = -ballYdir;
} if (ballposX > 670) {
ballXdir = -ballXdir;
}
} repaint();
} public void keyTyped(KeyEvent e) {
} public void keyReleased(KeyEvent e) {
} public void keyPressed(KeyEvent e) {
if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
if (playerX >= 600) { playerX = 600;
}
else {
moveRight();
}
}
if (e.getKeyCode() == KeyEvent.VK_LEFT) {
if (playerX < 10) {
playerX = 10;
}
else {
moveLeft();
}
}
if (e.getKeyCode() == KeyEvent.VK_ENTER) {
if (!play) {
play = true;
ballposX = 120;
ballposY = 350;
ballXdir = -1;
ballYdir = -2;
playerX = 310;
score = 0;
totalBricks = 21;
map = new MapGenerator(3, 7);
repaint();
}
```

```java
}
}
public void moveRight() {
play = true;
playerX += 20;
}
public void moveLeft() {
play = true;
playerX -= 20;
}
}
package brickBracker;
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics2D;
public class MapGenerator {
public int map[][];
public int brickWidth;
public int brickHeight;
public MapGenerator(int row, int col) {
map = new int[row][col];
for (int i = 0; i < map.length; i++) {
for (int j = 0; j < map[0].length; j++) {
map[i][j] = 1;
}
}

brickWidth = 540 / col;
brickHeight = 150 / row;
}
public void draw(Graphics2D g) {
for (int i = 0; i < map.length; i++) {
for (int j = 0; j < map[0].length; j++) {
if (map[i][j] > 0) {
g.setColor(Color.white);
g.fillRect(j * brickWidth + 80, i * brickHeight + 50, brickWidth, brickHeight);
g.setStroke(new BasicStroke(3));
g.setColor(Color.black);
g.drawRect(j * brickWidth + 80, i * brickHeight + 50, brickWidth, brickHeight);
}
}
}
}
public void setBrickValue(int value, int row, int col) {
map[row][col] = value;
```
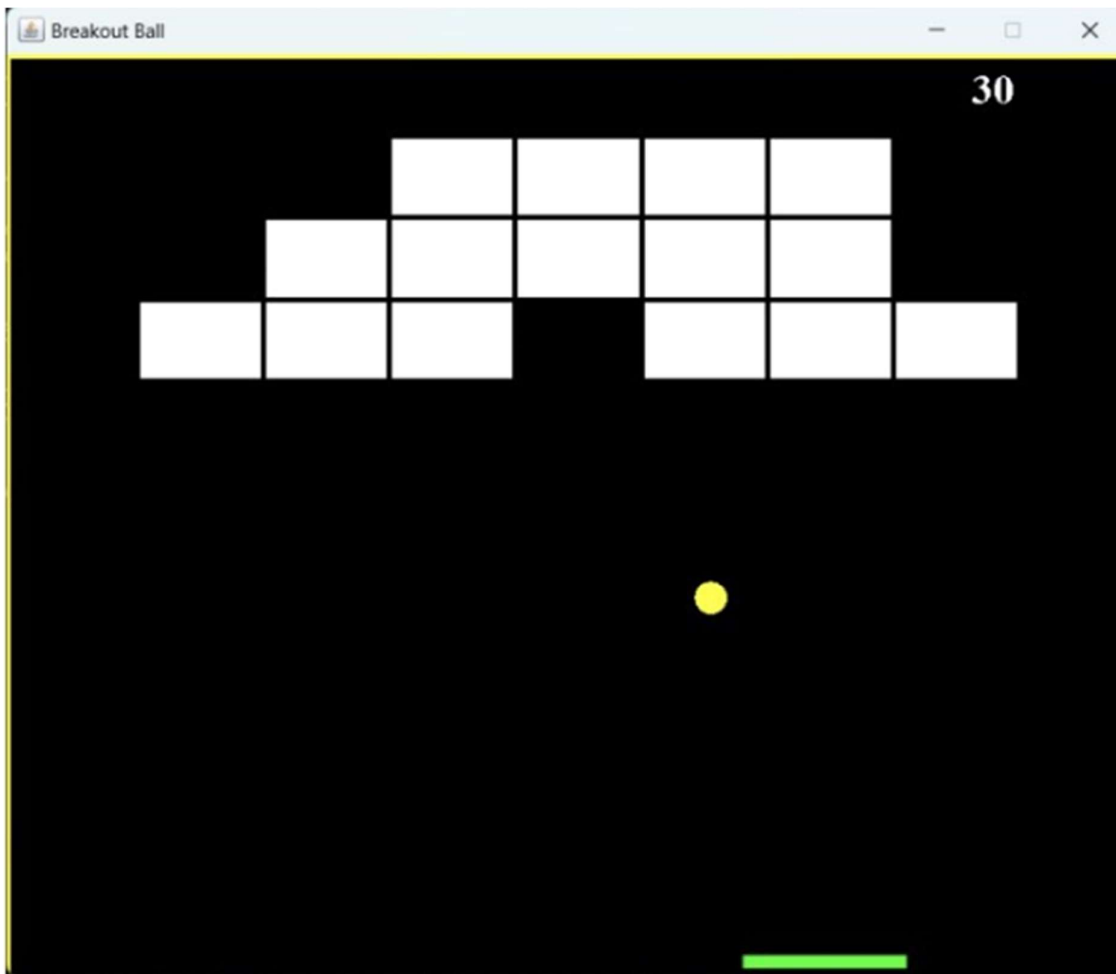
```
}
}
```

Output:

Conclusion:

In this program, we utilized various header files like, the graphics.h library for drawing, dos.h for timing, and math.h for mathematical calculations. We set up a simulation of a solar system, which involves initializing the graphics window, running the simulation loop, and finally, cleaning up resources and exiting the program. This program allows for the visual representation of a solar system using basic C graphics capabilities.

Comment on –

1. **Importance of story building-** Story building is crucial as it forms the backbone of any narrative, providing structure, depth, and coherence to the plot. It engages the audience, creates emotional connections, and allows for meaningful exploration of themes and characters.

2. **Defining the basic character of story**- The basic character of a story encompasses its fundamental elements, including the protagonist, antagonist, setting, and central conflict. These elements lay the foundation for the narrative, shaping the events and driving the plot forward. A well-defined basic character is essential for a compelling and relatable story.

3. **Apply techniques to these characters-** Applying techniques to characters involves imbuing them with depth, complexity, and relatability. This can be achieved through methods such as character development arcs, dialogue that reveals their personalities, and vivid descriptions. Techniques also include providing characters with motivations, flaws, and growth, allowing them to resonate with the audience and contribute meaningfully to the narrative.