**Assignment 1 Report**

**Youtube Links:**

**Main game:** [**https://youtu.be/tBmkqZIBL0c**](https://youtu.be/tBmkqZIBL0c)

**Updated game:** [**https://youtu.be/aMzHOANtHeo**](https://youtu.be/aMzHOANtHeo)

**Introduction**:

In this assignment I developed a variant of “DX Ball” game where player controls a mobile paddle and aiming to hit and break all the bricks with ball without letting the ball touch to the bottom of the canvas. The game consists of a dynamically moving ball, a thin rod which is for player to throw the ball at what angle he/she wants , a paddle that the player controls using arrow keys, and several bricks that the ball can hit and break upon collision. Game have features like pause/start and status like angle, score and finishing messages to let the user know.

I used StdDraw library which is useful for basic graphical applications. I used it for drawing shapes (filled circle, filled rectangle), writing texts (angle, score, game status, victory/game over) onto canvas, to detect key events (with isKeyPressed() method) for ball and paddle movement and pause and start the game and also used double buffering and show() for faster animations.

**Game Mechanisms:**

Line Movement Mechanism: Firstly i drew a line at proper location and length. I calculated the length of the line by Pythogoras Theorem ;

LineLength= √( (x2​−x1​)2 + (y2​−y1​)2) ) (1)

where x1 and y1 are starting points of the line and x2 and y2 are end points of the line respectively. Later that, when left key or right key is pressed angle variable is increasing or decreasing respectively. Then i decide the new end points of the line by equations:

*x2 = x1 + r \* cos(angle) , y2 = y1 + r \* sin(angle). (2)*

*Lastly displaying the new line on the canvas.*

*Paddle Control Mechanism: Firstly i drew a rectangle with respect to given height and width corresponding to the paddle. Then in a while loop i checked if left key or right key is pressed , if so i incremented or decremented the x component of paddle position at a magnitüde of given paddle\_speed variable respectively. Then i cleared and drew the paddle again with new coordinates after each movement.*

*(ex.: paddle\_pos[0] -= paddle\_speed;) (3)*

*Ball Movement Mechanism: Firstly i drew the ball with given features (radius, color) then after the very first space bar press ball starts moving with respect to the angle we calculated using a thin rod. To do so i used the ball\_velocity and the angle the rod directed in equations;*

*x\_Velocity = ball\_velocity \* cos(movingAngle)*

*y\_Velocity = ball\_velocity\*sin(movingAngle). (4)*

*Then summed the ball’s x and y coordinate with x\_Velocity and y\_Velocity in a while loop respectively. So ball will move until it collides a brick or wall. I determined if the ball collided with anything and changed the velocity of it properly. To do so, firstly, i determined if the ball collided any way then if this collision occured with a corner or not. For the first decision i used statements like;*

*xBall + 8 >= xPaddle - paddle\_halfwidth &&*

*xBall - 8 <= xPaddle + paddle\_halfwidth && … etc. (5)*

*For the second one i used the equation;*

*cornerHit = (|xBall - xBrick| > brick\_halfwidth) &&*

*(|yBall - yBrick| > brick\_halfheight) (6)*

*(for brick corner collision) and if it is true it means collision occured with a corner. Lastly i determined the x and y component of ball’s velocity after collisions. If the collision occured with a horizontal edge i changed y component’s sign, if it is with a vertical edge i changed x component’s sign, if it is with a corner i changed both x and y components by drawing a standard normal vector where normal vector*

*N = (xBall - xBrick, yBall – yBrick) (7)*

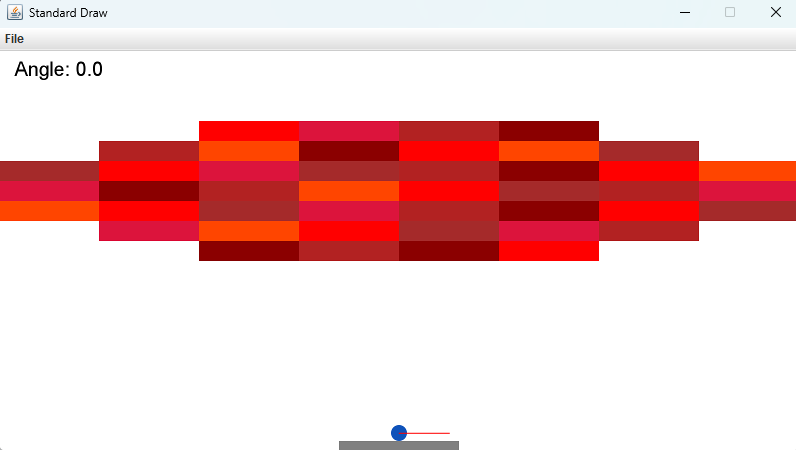
*and by diving N to its magnitude i found the standard normal vector. Then i used*

*Vnew = Vold - 2(Vold . N) N (8)*

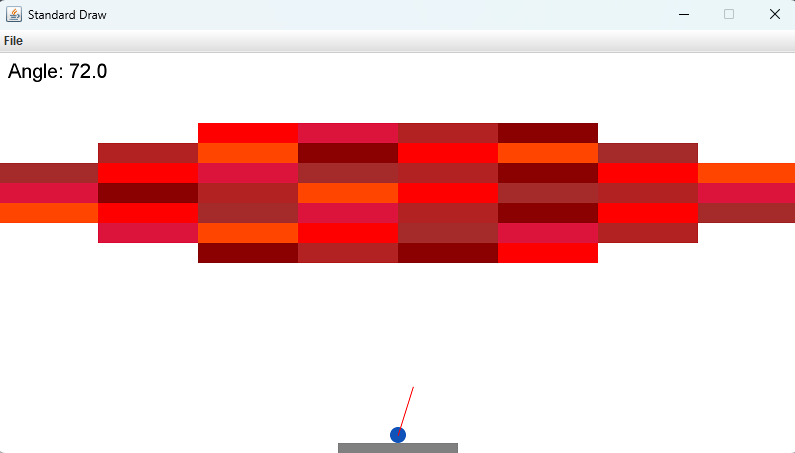
*formula to calculate new velocity vector. Then with the new components*

*Bricks: Firstly i drew the brick as filled rectangles with given features ( height, width, color, coordinates). Then after then every collision i remove the collided brick and its color from respective arrays and incremented the score 10 points. Then after those i drew the whole bricks rest again.*

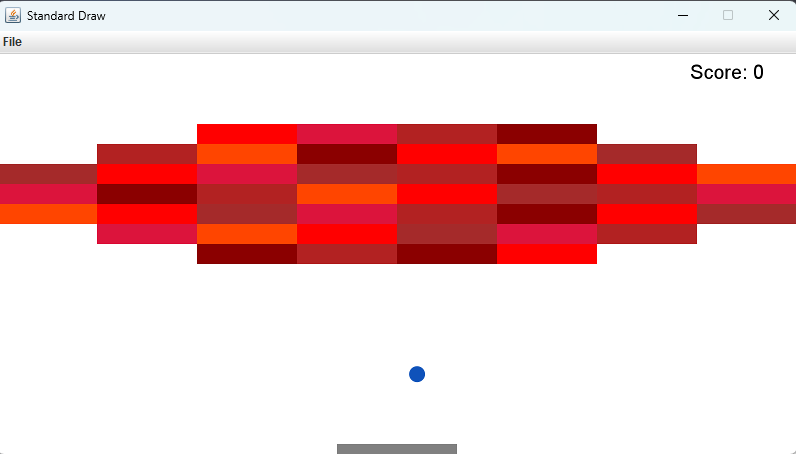
*Implementation Details:*

**

*Figure 1: Initial Game*

**

*Figure 2: Angle determining*

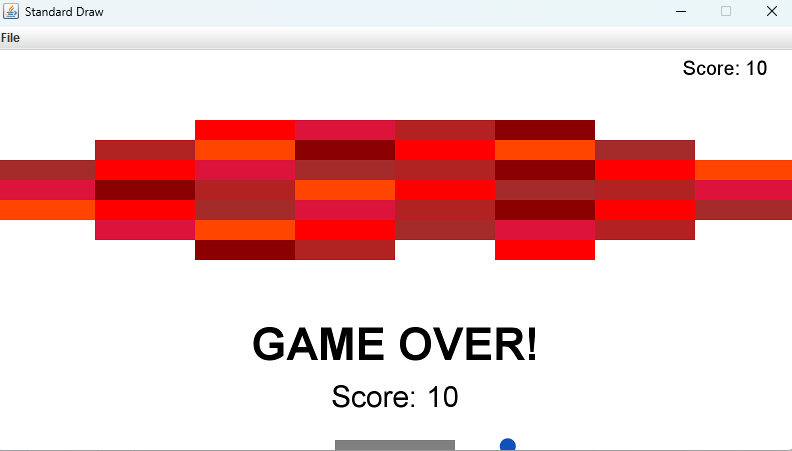
**

*Figure 3: First shooting*

ekran görüntüsü, metin, multimedya yazılımı, yazılım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Figure 4: Pause screen

**

*Figure 5: Game over screen*

metin, ekran görüntüsü, yazılım, işletim sistemi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Figure 6: Victory screen

metin, ekran görüntüsü, çizgi, renklilik içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

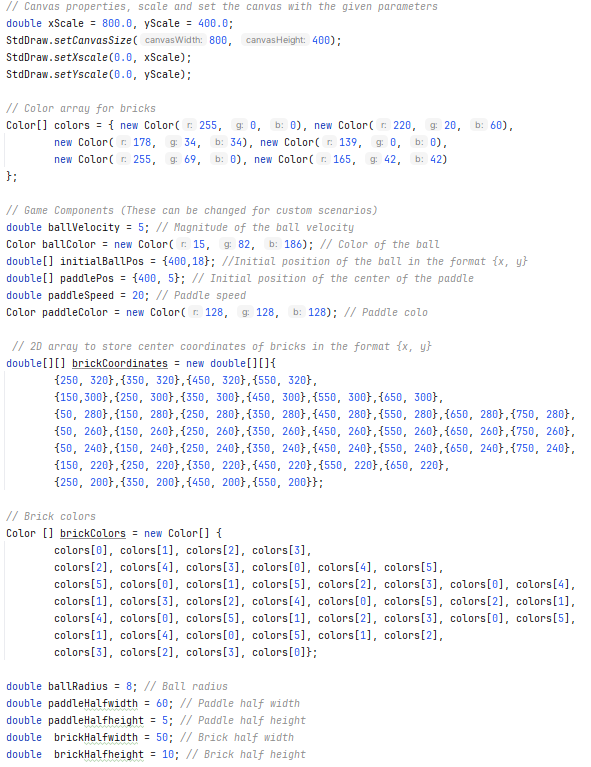
Figure 7: Updated game Game Over scree

Code 1: Angle determining rod movement

metin, ekran görüntüsü, yazı tipi, doküman, belge içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Code 2: Game parameters and variables (same as in description)



Code 3: Ball and paddle movement, collision and game status arrangement

metin, ekran görüntüsü, doküman, belge, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Code 4: Methods for corner collisions

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Code 5: Wall and paddle collisions handling

metin, ekran görüntüsü, doküman, belge, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Code 6: Brick collisions handling

metin, ekran görüntüsü, doküman, belge, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Code 7: Updated game variables

metin, ekran görüntüsü, doküman, belge, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.