CmpE 160 Assignment 1
DX-BALL

Contact:
Deniz Baran Aksoy (deniz.aksoy@bogazici.edu.tr)

Due: March 17th 5 a.m. in the morning

Description

In this assignment, you are expected to implement a simple brick-breaking game similar to DX-BALL using Java and StdDraw graphics library. You can watch a game-play video of the implementation of DX_BALL game here. In the game, the player controls a paddle at the bottom of the screen to bounce the ball and break the bricks at the top. If the ball hits a brick, the ball bounces off, and the brick is destroyed. The goal of the game is to break all the bricks with the ball without letting the ball fall below the paddle.

Game Environment

As shown in Figures 1 and 2, the game environment consists of the following elements:

- **Paddle :** A gray colored rectangle is located at the bottom of the screen. It moves along the x-axis and bounces the ball.
- **Ball :** A blue-colored circle. It moves in a straight line unless it collides with a surface.
- **Bricks :** Colored rectangles that disappear when hit by the ball, awarding the player 10 points per brick.
- **Shooting Direction :** A red line indicates the initial trajectory of the ball.
- **Shooting Angle :** Shows the initial angle of the ball.
- **Score :** Shows the player's score.
- **Game Status :** Displays the game status. If the game is paused, it displays **"PAUSED"**.

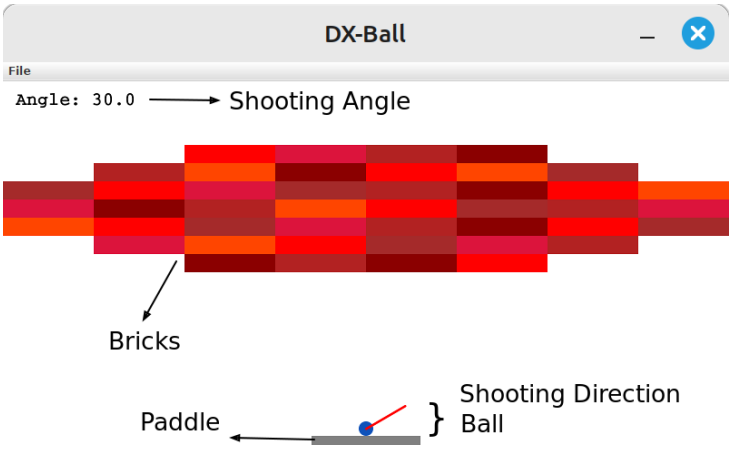

Figure 1: Initial Game

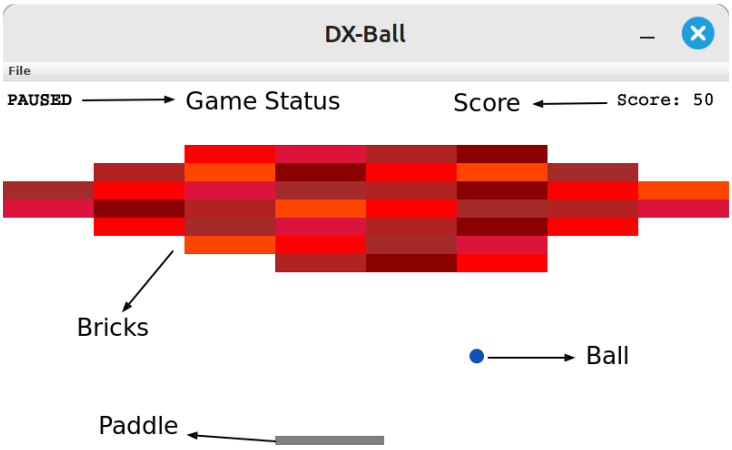

Figure 2: Paused Game

Game Mechanics

a) Paddle Control

The paddle should be controllable using keyboard inputs (left ⟵ and right ⟶ arrow keys). If the player presses either the left ⟵ or the right ⟶ arrow key, the paddle should move in the corresponding direction. However, the paddle must remain within the predetermined canvas boundaries, preventing it from moving beyond the leftmost and rightmost edges. This movement of the paddle is determined by Equation (1) where

- x_0 and y_0 are the initial coordinates of the paddle's center
- x_1 and y_1 are the updated coordinates of the paddle's center
- v_x and v_y are the x and y components of the paddle's velocity
- dt is the time step

The paddle's velocity in the y direction v_y is typically zero since the paddle only moves along the x-axis. The time step can be set to 1 or another value, depending on the implementation.

$$(x_1, y_1) = (x_0 + v_x dt,\ y_0 + v_y dt) \tag{1}$$

b) Ball Movement

The ball moves in a straight line following Equation (1) until it collides with a surface. If the ball hits a surface (such as a brick, paddle, or a wall), it must bounce off the surface according to elastic collision principles. There are two main types of collisions to consider: bouncing off a flat surface and bouncing off a corner.

Surface Collision

If the ball collides with a flat surface, it bounces off, as illustrated in Figure 3 below.

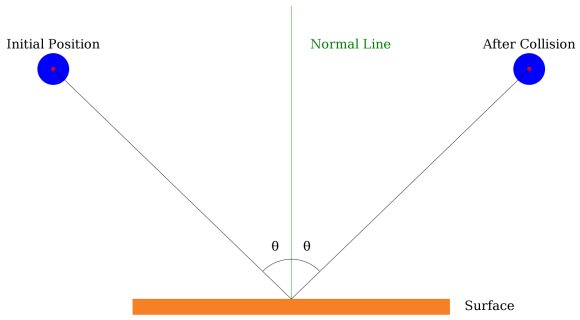

Figure 3: Surface Collision

Corner Collision

On the other hand, if the ball collides with a corner, its final velocity and direction change depending on the exact point of collision. Thus, the exact point of collision should be determined first. Once the collision point is found, a tangent line representing the effective collision surface can be constructed. Subsequently, a normal line perpendicular to the tangent is found, and the ball's final velocity is calculated based on the reflection principle. This process is illustrated in Figure 4.
As for the precision, you can simply assume the level of accuracy used in the lab exercises. However, if you want to calculate the reflection angle more precisely, you are free to do so. In either case, you must **clearly explain** your approach, assumptions, and calculations in the **report**.

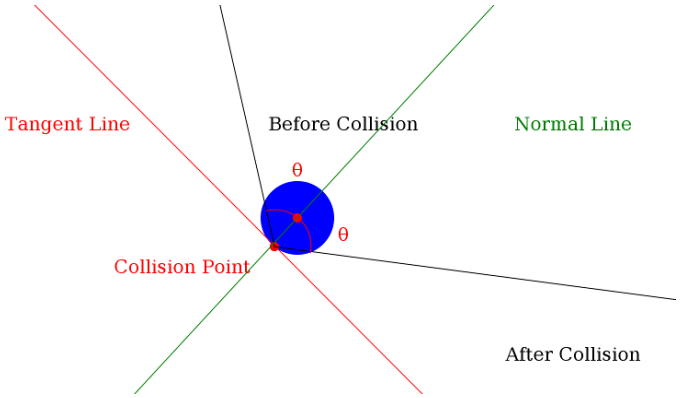

Figure 4: Corner Collision

c) Bricks

Bricks are positioned at the top of the canvas, arranged in rows and columns. They are the primary obstacles that the ball interacts with during the game. When the ball collides with a brick, the brick disappears, and the player earns **10 points**. The game continues until the player clears all the bricks.

Gameplay

- The game's objective is to clear all the bricks without letting the ball hit the bottom wall.
- The game starts with the ball and paddle positioned at the center of the canvas along the x-axis.
- The player can adjust the initial shooting angle using the left (←) and the right (→) arrow keys. The shooting angle is illustrated with a thin red line, and its value is displayed at the top-left corner of the canvas.
- Once the desired angle is set, the player presses the **SPACE** key and launches the ball, initiating the game.
- During the game, the player can **pause** the game at any time by pressing the **SPACE** key; pressing it again resumes the play.
- The ball moves straight and bounces off surfaces upon collision. If the ball collides with a **brick**, the brick disappears, and the player is awarded **10 Points**.
- The player can control the paddle using the left (←) and the right (→) arrow keys.
- If the ball collides with the bottom surface, the game is over, and a **GAME OVER!** message is displayed along with the player's final score, as illustrated in Figure 5.
- If the player clears all the bricks, the game ends with a **VICTORY!** message, displaying the final score, as illustrated in Figure 6.

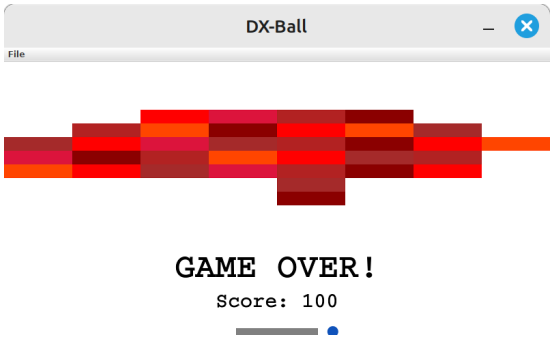

Figure 5: Game over screen

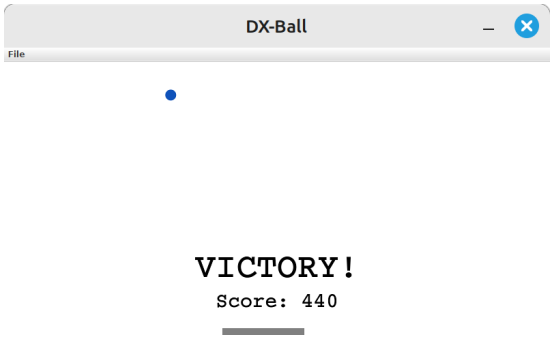

Figure 6: Victory screen

Game Parameters

Use the game parameters provided in Code Box 1 below in your implementation. You may modify the brick positions or colors to create new and challenging scenarios.

Code 1 : Game Parameters Java `java`

```java
// Canvas properties, scale and set the canvas with the given parameters
double xScale = 800.0, yScale = 400.0;
StdDraw.setCanvasSize(800, 400);
StdDraw.setXscale(0.0, xScale);
StdDraw.setYscale(0.0, yScale);

// Color array for bricks (first import java.awt.Color )
Color[] colors = { new Color(255, 0, 0), new Color(220, 20, 60),
                   new Color(178, 34, 34), new Color(139, 0, 0),
                   new Color(255, 69, 0), new Color(165, 42, 42)
};

// Game Components (These can be changed for custom scenarios)

double ballRadius = 8; // Ball radius
double ballVelocity = 5;  // Magnitude of the ball velocity
Color ballColor = new Color(15, 82, 186); // Color of the ball
double[] initialBallPos = {400,18}; //Initial position of the ball in the format {x, y}

double[] paddlePos = {400, 5}; // Initial position of the center of the paddle
double paddleHalfwidth = 60; // Paddle halfwidth
double paddleHalfheight = 5; // Paddle halfheight
double paddleSpeed = 20;   // Paddle speed
Color paddleColor = new Color(128, 128, 128);  // Paddle color

double brickHalfwidth = 50; // Brick halfwidth
double brickHalfheight = 10; // Brick halfheight

// 2D array to store center coordinates of bricks in the format {x, y}
double[][] brickCoordinates = new double[][]{
                {250, 320},{350, 320},{450, 320},{550, 320},
        {150,300},{250, 300},{350, 300},{450, 300},{550, 300},{650, 300},
{50, 280},{150, 280},{250, 280},{350, 280},{450, 280},{550, 280},{650, 280},{750, 280},
{50, 260},{150, 260},{250, 260},{350, 260},{450, 260},{550, 260},{650, 260},{750, 260},
{50, 240},{150, 240},{250, 240},{350, 240},{450, 240},{550, 240},{650, 240},{750, 240},
        {150, 220},{250, 220},{350, 220},{450, 220},{550, 220},{650, 220},
                {250, 200},{350, 200},{450, 200},{550, 200}};

// Brick colors
Color [] brickColors = new Color[] {
                    colors[0], colors[1], colors[2], colors[3],
           colors[2], colors[4], colors[3], colors[0], colors[4], colors[5],
colors[5], colors[0], colors[1], colors[5], colors[2], colors[3], colors[0], colors[4],
colors[1], colors[3], colors[2], colors[4], colors[0], colors[5], colors[2], colors[1],
colors[4], colors[0], colors[5], colors[1], colors[2], colors[3], colors[0], colors[5],
           colors[1], colors[4], colors[0], colors[5], colors[1], colors[2],
                    colors[3], colors[2], colors[3], colors[0]};
```

Report Format

Your report should be concise and consist of the following sections:

- **Introduction:** Briefly explain the game, and the library(StdDraw) you used. The introduction should not exceed half a page.
- **Game Mechanism:** Explain the core mechanics of your implementation, including how the ball moves and collisions are handled.
- **Implementation Details:** Provide a brief overview of your code with screenshots.

Your report should **properly reference all figures, tables, and equations**. Figures and tables should be numbered and cited appropriately (as shown in Figure 1, etc.). Each figure must have a caption placed below it, clearly explaining its content. Each table must have a caption positioned above it. Lastly, all equations should be centered and numbered for reference.

Notes

- Use `xScale`, `yScale`, `brickCoordinates[][]` and `brickColors[]` variable names in your implementation.
- You **must** use the **StdDraw graphics library**.
- **Do not use** object-oriented programming concepts in this assignment.
- In all assignments, do not use topics we have not covered in the lectures.

Submission Files

1. Java source code (**.java file**). Submission details are explained in greater detail in the Submission Guide section.
2. Report (**.pdf format**) explaining your implementation, including screenshots of your gameplay covering different cases. Additionally, create your own game scene by changing parameters and brick positions. Provide screenshots of your setup.
3. Provide **two YouTube links at the beginning of your report**: one showing the game played with the game settings we provided and another one for the game you designed. Ensure that all possible features of the game are visible in the videos.

Submission Guide

Submission Files

Submit a single compressed (`.zip`) file, named as `NameSurname.zip`, to Moodle. It should contain all source code files (under the `\code` directory), the report (in **PDF** format, under the `\report` directory), and all other files if needed (under the `\misc` directory). Name the Java file with given parameters as `NameSurname.java` and the Java file with your configuration as `NameSurnameModified.java`. Name your report as `NameSurname.pdf`. Do not use Turkish characters in filenames, code, or comments.

Late Submission Policy

The maximum late submission duration is two days. Late submissions will be graded at 50% of the original grade.

Mandatory Submission

Submission of assignments is **mandatory**. If you do not submit an assignment, you **fail** the course.

Plagiarism

This leads to a grade of F, and YÖK regulations will be applied.