Computer Graphics 2D Project Presentation
Muazzam Bin Aqeel: CTS-5

1. Game Logic:

```
if (play) {
  // Game logic
}
```

The game logic is executed only if the variable play is true. This condition determines whether the game should be actively running.x

2. Brick Collision and Score Update:

```
for (int i = 0; i < map.map_game.length; i++) {
   for (int j = 0; j < map.map_game[0].length; j++) {
      // Check for collision with bricks and update score

      if (map.map_game[i][j] > 0) {
         // ... (collision detection and handling)
      }
   }
}
```

Iterates through the 2D array representing the game's bricks. If a brick is present, it checks for collision with the ball, updates the score, and removes the brick.

3. Ball Position Update and Wall Reflection:

```
// Update ball position based on direction
ballposX += ballXdir;
ballposY += ballYdir;

// Reflect ball off walls
if (ballposX < 0 || ballposX > 670) {

   ballXdir = -ballXdir;
}
if (ballposY < 0) {

   ballYdir = -ballYdir;
}
```

Updates the ball's position based on its direction and reflects it off the walls if it reaches the screen boundaries.

```
m Bin Aqeel *
de
void actionPerformed(ActionEvent e) {
er.start();
Game logic
(play) {       // Handle ball and paddle collision
 if (new Rectangle(ballposX, ballposY, 20, 20).intersects(new Rectangle(playerX, 550, 100, 8))) {
     ballYdir = -ballYdir;
     hitPaddleSound.setFramePosition(0);
     hitPaddleSound.start();     // Play the paddle-hit sound
 } // Check for collision with bricks and update score
 A:
 for (int i = 0; i < map.map_game.length; i++) {
     for (int j = 0; j < map.map_game[0].length; j++) {
         if (map.map_game[i][j] > 0) {
             int brickX = j * map.Blocks_Width + 80;
             int brickY = i * map.Blocks_Length + 50;
             int bricksWidth = map.Blocks_Width;
             int bricksHeight = map.Blocks_Length;
             Rectangle rect = new Rectangle(brickX, brickY, bricksWidth, bricksHeight);
             Rectangle ballrect = new Rectangle(ballposX, ballposY, 20, 20);
             Rectangle brickrect = rect;
             // Handle ball and brick collision
             if (ballrect.intersects(brickrect)) {
                 map.Total_Blocks(0, i, j);
                 totalbricks--;
                 score += 5;
                 // Change ball direction based on collision point
                 if (ballposX + 19 <= brickrect.x || ballposX + 1 >= brickrect.x + bricksWidth) {
                     ballXdir = -ballXdir;
                 } else {
                     ballYdir = -ballYdir;
                 }
                 // Play sound if the ball hits the paddle
                 if (ballYdir > 0 && ballrect.intersects(new Rectangle(playerX, 550, 100, 8))) {
                     toolkit.beep();}
                 break A;
             }
         }
     }
 }
```

```
        // Update ball position based on direction
        ballposX += ballXdir;
        ballposY += ballYdir;

        // Reflect ball off walls
        if (ballposX < 0) {
            ballXdir = -ballXdir;
        }
        if (ballposY < 0) {
            ballYdir = -ballYdir;
        }
        if (ballposX > 670) {
            ballXdir = -ballXdir;
        }
    }
    // Repaint the game components
    repaint();
}
```