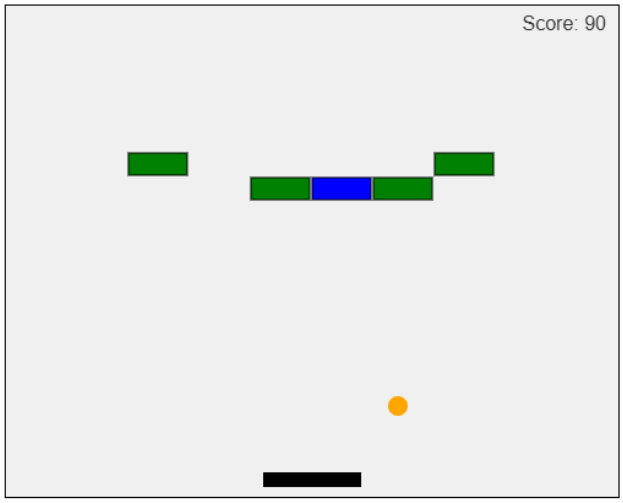
Łukasz Zawodziński gr. 3
nr Indexu: 136699

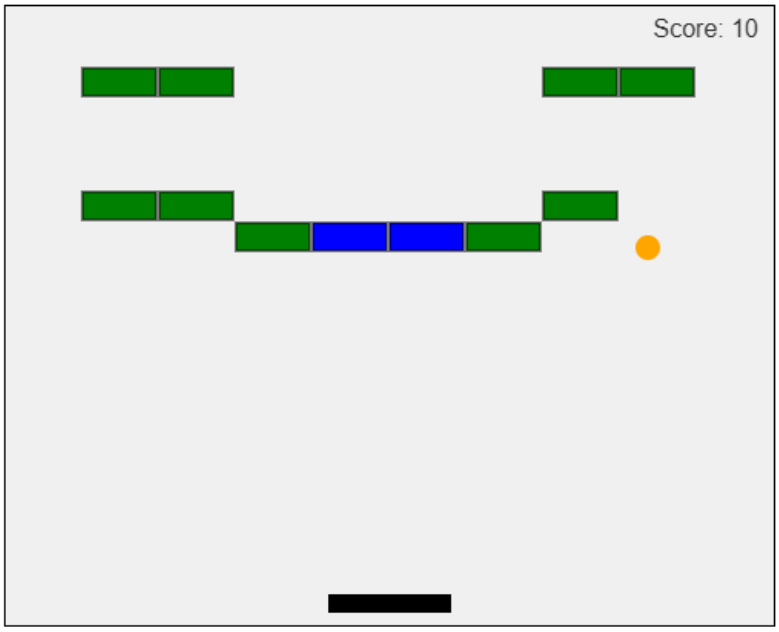
Interaktywna grafika i prezentacja danych

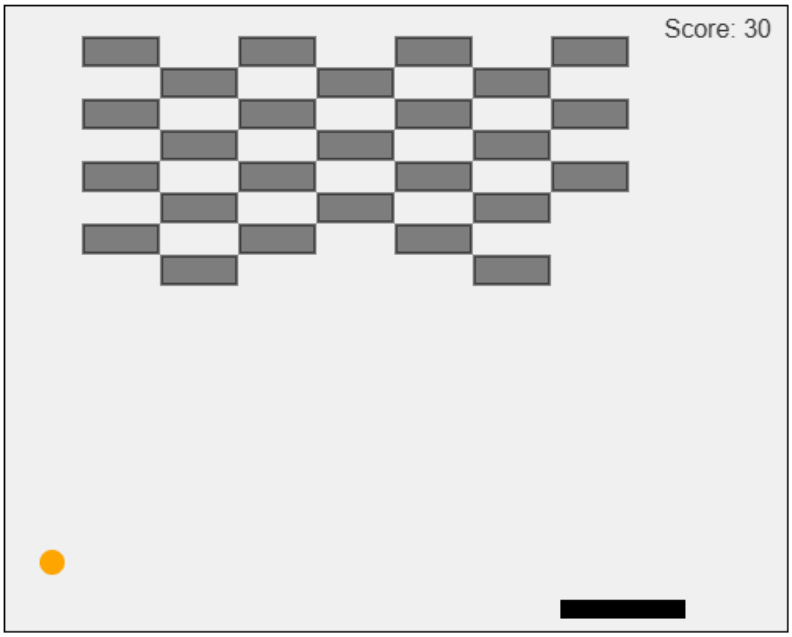
Laboratorium 8

Podstawy obsługi Canvas

Gra Arkanoid z kilkoma poziomami








```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Gra Arkanoid</title>
    <style>
      canvas {
        border: 1px solid #000;
        background-color: #f0f0f0;
      }
    </style>
  </head>
```

```
<body>
  <canvas id="gameCanvas" width="500" height="400"></canvas>
  <script>
    const brick = {
      width: 48,
      height: 18,
      fillStyles: ["green", "blue", "yellow", "red"],
      strokeStyle: "black",
    };

    const levels = [
      [
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 1, 1, 0, 0, 0, 0, 0, 1, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 1, 1, 0, 0, 0, 0, 1, 1, 0],
        [0, 0, 0, 1, 2, 2, 1, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
      ],
      [
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 4, 0, 4, 0, 4, 0, 4, 0, 0],
        [0, 0, 5, 0, 5, 0, 5, 0, 0, 0],
        [0, 6, 0, 6, 0, 6, 0, 6, 0, 0],
        [0, 0, 7, 0, 7, 0, 7, 0, 0, 0],
        [0, 4, 0, 4, 0, 4, 0, 4, 0, 0],
        [0, 0, 5, 0, 5, 0, 5, 0, 0, 0],
        [0, 6, 0, 6, 0, 6, 0, 6, 0, 0],
        [0, 0, 7, 0, 7, 0, 7, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
      ],
      [
        [0, 4, 5, 6, 7, 4, 5, 6, 7, 0],
        [0, 3, 2, 1, 2, 3, 2, 1, 2, 0],
        [0, 2, 1, 2, 3, 2, 1, 2, 3, 0],
        [0, 1, 2, 3, 2, 1, 2, 3, 2, 0],
        [0, 4, 5, 6, 7, 4, 5, 6, 7, 0],
        [0, 3, 2, 1, 2, 3, 2, 1, 2, 0],
        [0, 2, 1, 2, 3, 2, 1, 2, 3, 0],
        [0, 1, 2, 3, 2, 1, 2, 3, 2, 0],
        [0, 4, 5, 6, 7, 4, 5, 6, 7, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
      ],
      [
        [7, 6, 5, 4, 3, 2, 1, 2, 3, 4],
        [4, 3, 2, 1, 2, 3, 4, 5, 6, 7],
        [7, 6, 5, 4, 3, 2, 1, 2, 3, 4],
        [4, 3, 2, 1, 2, 3, 4, 5, 6, 7],
        [7, 6, 5, 4, 3, 2, 1, 2, 3, 4],
        [4, 3, 2, 1, 2, 3, 4, 5, 6, 7],
        [7, 6, 5, 4, 3, 2, 1, 2, 3, 4],
        [4, 3, 2, 1, 2, 3, 4, 5, 6, 7],
        [7, 6, 5, 4, 3, 2, 1, 2, 3, 4],
```

```
      [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    ],
  ];

  let currentLevel = 0;
  let bricks = JSON.parse(JSON.stringify(levels[currentLevel]));

  let score = 0;

  let ball = {
    x: 250,
    y: 300,
    radius: 8,
    dx: 2,
    dy: -2,
  };

  let balls = [Object.assign({}, ball)];

  let paddle = {
    width: 80,
    height: 12,
    x: 210,
    y: 380,
    dx: 5,
  };

  let rightPressed = false;
  let leftPressed = false;

  let levelTransition = false;

  const canvas = document.getElementById("gameCanvas");
  const ctx = canvas.getContext("2d");

  document.addEventListener("keydown", function (e) {
    if (e.key === "ArrowRight") rightPressed = true;
    else if (e.key === "ArrowLeft") leftPressed = true;
  });
  document.addEventListener("keyup", function (e) {
    if (e.key === "ArrowRight") rightPressed = false;
    else if (e.key === "ArrowLeft") leftPressed = false;
  });

  function drawBricks() {
    for (let j = 0; j < 10; j++) {
      for (let i = 0; i < 10; i++) {
        if (bricks[j][i] > 0) {
          let type = bricks[j][i];
          ctx.beginPath();
          if (type >= 1 && type <= 3) {
            ctx.fillStyle = brick.fillStyles[type - 1];
          } else {
            ctx.fillStyle = "gray";
          }
          ctx.fillRect(
            i * (brick.width + 2),
```

```
            j * (brick.height + 2),
            brick.width,
            brick.height
          );
          ctx.strokeStyle = brick.strokeStyle;
          ctx.strokeRect(
            i * (brick.width + 2),
            j * (brick.height + 2),
            brick.width,
            brick.height
          );
          ctx.closePath();
        }
      }
    }
  }

  function drawBall(ballObj) {
    ctx.beginPath();
    ctx.arc(ballObj.x, ballObj.y, ballObj.radius, 0, Math.PI * 2);
    ctx.fillStyle = "orange";
    ctx.fill();
    ctx.closePath();
  }

  function drawPaddle() {
    ctx.beginPath();
    ctx.rect(paddle.x, paddle.y, paddle.width, paddle.height);
    ctx.fillStyle = "black";
    ctx.fill();
    ctx.closePath();
  }

  function drawScore() {
    ctx.font = "16px Arial";
    ctx.fillStyle = "#333";
    ctx.textAlign = "right";
    ctx.fillText("Score: " + score, canvas.width - 10, 20);
  }

  function collisionDetection(ballObj) {
    for (let j = 0; j < 10; j++) {
      for (let i = 0; i < 10; i++) {
        if (bricks[j][i] > 0) {
          let bx = i * (brick.width + 2);
          let by = j * (brick.height + 2);
          if (
            ballObj.x + ballObj.radius > bx &&
            ballObj.x - ballObj.radius < bx + brick.width &&
            ballObj.y + ballObj.radius > by &&
            ballObj.y - ballObj.radius < by + brick.height
          ) {
            let type = bricks[j][i];
            if (type >= 1 && type <= 3) {
              score += 10 * type;
              bricks[j][i]--;
            } else {
```

```
              bricks[j][i] = 0;
            }
            ballObj.dy = -ballObj.dy;
            return;
          }
        }
      }
    }

    function isLevelCleared() {
      for (let j = 0; j < 10; j++) {
        for (let i = 0; i < 10; i++) {
          if (bricks[j][i] > 0) return false;
        }
      }
      return true;
    }

    function nextLevel() {
      levelTransition = false;
      currentLevel++;
      if (currentLevel >= levels.length) {
        setTimeout(() => {
          alert(
            "Gratulacje! Ukończono wszystkie poziomy!\nTwój wynik: " + score
          );
          currentLevel = 0;
          score = 0;
          bricks = JSON.parse(JSON.stringify(levels[currentLevel]));
          balls = [Object.assign({}, ball)];
          paddle.width = 80;
          levelTransition = false;
          requestAnimationFrame(draw);
        }, 100);
        return;
      }
      bricks = JSON.parse(JSON.stringify(levels[currentLevel]));
      balls = [Object.assign({}, ball)];
      paddle.width = 80;
      levelTransition = false;
      requestAnimationFrame(draw);
    }

    function draw() {
      ctx.clearRect(0, 0, canvas.width, canvas.height);
      drawBricks();
      drawPaddle();
      drawScore();

      for (let b = balls.length - 1; b >= 0; b--) {
        let ballObj = balls[b];
        drawBall(ballObj);
        collisionDetection(ballObj);

        ballObj.x += ballObj.dx;
        ballObj.y += ballObj.dy;
```

```
        if (
          ballObj.x + ballObj.radius > canvas.width ||
          ballObj.x - ballObj.radius < 0
        ) {
          ballObj.dx = -ballObj.dx;
        }
        if (ballObj.y - ballObj.radius < 0) {
          ballObj.dy = -ballObj.dy;
        }

        if (
          ballObj.y + ballObj.radius > paddle.y &&
          ballObj.x > paddle.x &&
          ballObj.x < paddle.x + paddle.width
        ) {
          ballObj.dy = -Math.abs(ballObj.dy);
        }

        if (ballObj.y - ballObj.radius > canvas.height) {
          balls.splice(b, 1);
        }
      }

      if (balls.length === 0) {
        balls = [Object.assign({}, ball)];
        paddle.width = 80;
      }

      if (rightPressed && paddle.x + paddle.width < canvas.width) {
        paddle.x += paddle.dx;
      }
      if (leftPressed && paddle.x > 0) {
        paddle.x -= paddle.dx;
      }

      if (isLevelCleared()) {
        if (!levelTransition) {
          levelTransition = true;
          setTimeout(nextLevel, 500);
        }
        return;
      }

      requestAnimationFrame(draw);
    }

    draw();
  </script>
 </body>
</html>
```