const canvas = document.getElementById("gameCanvas");

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

const startButton = document.getElementById("startButton");

const pauseButton = document.getElementById("pauseButton");

const endButton = document.getElementById("endButton");

const nivelButtons = {

  facil: document.getElementById("nivelFacil"),

  medio: document.getElementById("nivelMedio"),

  dificil: document.getElementById("nivelDificil"),

};

const tileSize = 20;

const rows = 17;

const cols = 19;

canvas.width = cols \* tileSize;

canvas.height = rows \* tileSize;

let jogoRodando = false;

let pausado = false;

let lastTime = 0;

let ultimasPosicoesPredador = [];

const MAX\_MEMORIA\_PREDADOR = 5;

let velocidadeAtual = 200;

// Sons

const somMoeda = new Audio("coins.mp3");

const somCaptura = new Audio("gameover.mp3");

const somVitoria = new Audio("vitoria.mp3");

function tocarSom(som) {

  som.currentTime = 0;

  som.play();

}

// Mensagem Visual

const mensagem = document.createElement("div");

mensagem.style.position = "absolute";

mensagem.style.top = "50%";

mensagem.style.left = "50%";

mensagem.style.transform = "translate(-50%, -50%)";

mensagem.style.backgroundColor = "rgba(0, 0, 0, 0.85)";

mensagem.style.color = "#fff";

mensagem.style.padding = "20px 40px";

mensagem.style.borderRadius = "10px";

mensagem.style.fontSize = "24px";

mensagem.style.fontFamily = "Poppins, sans-serif";

mensagem.style.zIndex = 999;

mensagem.style.display = "none";

document.body.appendChild(mensagem);

function exibirMensagem(texto) {

  mensagem.textContent = texto;

  mensagem.style.display = "block";

}

// Mapa do Jogo

const map = [

  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

  [0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0],

  [0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0],

  [0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0],

  [0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0],

  [0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0],

  [0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0],

  [0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0],

  [0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0],

  [0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0],

  [0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0],

  [0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0],

  [0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0],

  [0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0],

  [0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0],

  [0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0],

  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

];

let pacman = { x: 1, y: 1, dx: 0, dy: 0, direction: "right" };

let predador = { x: cols - 2, y: rows - 2, dx: 0, dy: -1, direction: "up" };

let moedas = [];

let pontuacao = 0;

let vidas = 3;

// Referências aos elementos HTML

const pontuacaoElement = document.getElementById("pontuacao");

const vidasElement = document.getElementById("vidas");

// Função para atualizar a pontuação no HTML

function atualizarPontuacao() {

  pontuacaoElement.textContent = `Pontuação: ${pontuacao}`;

}

// Função para atualizar as vidas no HTML

function atualizarVidas() {

  vidasElement.textContent = `Vidas: ${vidas}`;

}

function criarMoedas() {

  moedas = [];

  for (let y = 0; y < rows; y++) {

    for (let x = 0; x < cols; x++) {

      if (map[y][x] === 1) {

        moedas.push({ x, y });

      }

    }

  }

}

function selecionarNivel(nivel) {

  switch (nivel) {

    case "facil":

      velocidadeAtual = 300;

      break;

    case "medio":

      velocidadeAtual = 200;

      break;

    case "dificil":

      velocidadeAtual = 120;

      break;

  }

  Object.entries(nivelButtons).forEach(([key, btn]) => {

    btn.classList.toggle("selected", key === nivel);

  });

}

function updatePredador() {

  const direcoes = [

    { dx: 0, dy: -1, direction: "up" },

    { dx: 0, dy: 1, direction: "down" },

    { dx: -1, dy: 0, direction: "left" },

    { dx: 1, dy: 0, direction: "right" },

  ];

  const opcoes = [];

  for (const dir of direcoes) {

    const nx = predador.x + dir.dx;

    const ny = predador.y + dir.dy;

    const destino = `${nx},${ny}`;

    if (nx >= 0 && nx < cols && ny >= 0 && ny < rows && map[ny][nx] === 1) {

      const distancia = Math.abs(pacman.x - nx) + Math.abs(pacman.y - ny);

      let penalidade = 0;

      for (let i = 0; i < ultimasPosicoesPredador.length; i++) {

        if (ultimasPosicoesPredador[i] === destino) {

          penalidade += (ultimasPosicoesPredador.length - i) \* 2;

        }

      }

      const pontuacao = distancia + penalidade;

      opcoes.push({ ...dir, nx, ny, pontuacao });

    }

  }

  if (opcoes.length > 0) {

    opcoes.sort((a, b) => a.pontuacao - b.pontuacao);

    const melhor = opcoes[0];

    ultimasPosicoesPredador.unshift(`${predador.x},${predador.y}`);

    if (ultimasPosicoesPredador.length > MAX\_MEMORIA\_PREDADOR) {

      ultimasPosicoesPredador.pop();

    }

    predador.x = melhor.nx;

    predador.y = melhor.ny;

    predador.dx = melhor.dx;

    predador.dy = melhor.dy;

    predador.direction = melhor.direction;

  }

  if (predador.x === pacman.x && predador.y === pacman.y) {

    vidas--;

    tocarSom(somCaptura);

    atualizarVidas(); // Atualiza as vidas no HTML

    if (vidas <= 0) {

      jogoRodando = false;

      exibirMensagem("Game Over!");

    } else {

      resetarJogador();

    }

  }

}

function resetarJogador() {

  pacman.x = 1;

  pacman.y = 1;

  pacman.dx = 0;

  pacman.dy = 0;

  pacman.direction = "right";

}

function update() {

  const nx = pacman.x + pacman.dx;

  const ny = pacman.y + pacman.dy;

  if (nx >= 0 && nx < cols && ny >= 0 && ny < rows && map[ny][nx] === 1) {

    pacman.x = nx;

    pacman.y = ny;

    const index = moedas.findIndex(

      (moeda) => moeda.x === pacman.x && moeda.y === pacman.y

    );

    if (index !== -1) {

      moedas.splice(index, 1);

      pontuacao += 10;

      tocarSom(somMoeda);

      atualizarPontuacao(); // Atualiza a pontuação no HTML

    }

  }

  updatePredador();

  if (moedas.length === 0) {

    jogoRodando = false;

    tocarSom(somVitoria);

    exibirMensagem("Você venceu!");

  }

}

function draw() {

  ctx.clearRect(0, 0, canvas.width, canvas.height);

  drawMap();

  drawMoedas();

  drawPacman();

  drawPredador();

}

function drawMap() {

  for (let y = 0; y < rows; y++) {

    for (let x = 0; x < cols; x++) {

      ctx.fillStyle = map[y][x] === 0 ? "blue" : "black";

      ctx.fillRect(x \* tileSize, y \* tileSize, tileSize, tileSize);

    }

  }

}

function drawMoedas() {

  ctx.fillStyle = "yellow";

  moedas.forEach(({ x, y }) => {

    ctx.beginPath();

    ctx.arc(

      x \* tileSize + tileSize / 2,

      y \* tileSize + tileSize / 2,

      5,

      0,

      Math.PI \* 2

    );

    ctx.fill();

  });

}

function drawPacman() {

  const angleMap = {

    right: 0,

    left: Math.PI,

    up: -Math.PI / 2,

    down: Math.PI / 2,

  };

  const rotation = angleMap[pacman.direction] || 0;

  ctx.save();

  ctx.translate(

    pacman.x \* tileSize + tileSize / 2,

    pacman.y \* tileSize + tileSize / 2

  );

  ctx.rotate(rotation);

  ctx.beginPath();

  ctx.moveTo(0, 0);

  ctx.arc(0, 0, tileSize / 2 - 2, Math.PI \* 0.25, Math.PI \* 1.75);

  ctx.fillStyle = "white";

  ctx.fill();

  ctx.restore();

}

function drawPredador() {

  const angleMap = {

    right: 0,

    left: Math.PI,

    up: -Math.PI / 2,

    down: Math.PI / 2,

  };

  const rotation = angleMap[predador.direction] || 0;

  ctx.save();

  ctx.translate(

    predador.x \* tileSize + tileSize / 2,

    predador.y \* tileSize + tileSize / 2

  );

  ctx.rotate(rotation);

  ctx.beginPath();

  ctx.moveTo(0, 0);

  ctx.arc(0, 0, tileSize / 2 - 2, Math.PI \* 0.25, Math.PI \* 1.75);

  ctx.fillStyle = "red";

  ctx.fill();

  ctx.restore();

}

function gameLoop(timestamp) {

  if (!jogoRodando || pausado) return;

  const deltaTime = timestamp - lastTime;

  if (deltaTime >= velocidadeAtual) {

    update();

    lastTime = timestamp;

  }

  draw();

  requestAnimationFrame(gameLoop);

}

function startGame() {

  jogoRodando = true;

  pausado = false;

  pacman = { x: 1, y: 1, dx: 0, dy: 0, direction: "right" };

  predador = { x: cols - 2, y: rows - 2, dx: 0, dy: -1, direction: "up" };

  ultimasPosicoesPredador = [];

  pontuacao = 0;

  vidas = 3;

  criarMoedas();

  lastTime = performance.now();

  mensagem.style.display = "none";

  atualizarPontuacao(); // Atualiza a pontuação inicial

  atualizarVidas(); // Atualiza as vidas iniciais

  requestAnimationFrame(gameLoop);

}

pauseButton.addEventListener("click", () => {

  if (!jogoRodando) return;

  pausado = !pausado;

  pauseButton.textContent = pausado ? "Continuar" : "Pausar";

  if (!pausado) {

    lastTime = performance.now();

    requestAnimationFrame(gameLoop);

  }

});

function endGame() {

  jogoRodando = false;

  pausado = false;

  pacman = { x: 1, y: 1, dx: 0, dy: 0, direction: "right" };

  predador = { x: cols - 2, y: rows - 2, dx: 0, dy: -1, direction: "up" };

  ultimasPosicoesPredador = [];

  moedas = [];

  pontuacao = 0;

  vidas = 3;

  pauseButton.textContent = "Pausar";

  mensagem.style.display = "none";

  draw();

}

window.onload = () => {

  selecionarNivel("facil");

};

// Referências aos botões virtuais

const btnUp = document.getElementById("btnUp");

const btnDown = document.getElementById("btnDown");

const btnLeft = document.getElementById("btnLeft");

const btnRight = document.getElementById("btnRight");

// Função para atualizar a direção do Pac-Man

function atualizarDirecao(dx, dy, direction) {

  if (!jogoRodando || pausado) return;

  pacman.dx = dx;

  pacman.dy = dy;

  pacman.direction = direction;

}

// Configurar eventos de clique para os botões virtuais

btnUp.addEventListener("touchstart", () => atualizarDirecao(0, -1, "up"));

btnDown.addEventListener("touchstart", () => atualizarDirecao(0, 1, "down"));

btnLeft.addEventListener("touchstart", () => atualizarDirecao(-1, 0, "left"));

btnRight.addEventListener("touchstart", () => atualizarDirecao(1, 0, "right"));

// Para desktop, também suportar cliques normais

btnUp.addEventListener("click", () => atualizarDirecao(0, -1, "up"));

btnDown.addEventListener("click", () => atualizarDirecao(0, 1, "down"));

btnLeft.addEventListener("click", () => atualizarDirecao(-1, 0, "left"));

btnRight.addEventListener("click", () => atualizarDirecao(1, 0, "right"));

startButton.addEventListener("click", startGame);

endButton.addEventListener("click", endGame);

draw();