main.js

```
// Btn
const playBtn = document.getElementById('playBtn');
const pauseBtn = document.getElementById('pauseBtn');
const resetBtn = document.getElementById('resetBtn');
const closeBtn = document.getElementById('closeBtn');

// Page
const homePage = document.querySelector('.home');
const gamePage = document.querySelector('.game');

// Navbar Game
const navName = document.getElementById('navName');
const navScore = document.getElementById('navScore');
const navTime = document.getElementById('navTime');

// Notif
const notifPage = document.querySelector('.notif');
const notifHead = document.getElementById('notif-head');
const notifScore = document.getElementById('notif-score');

// Assets
const enemyImg = document.getElementById('enemyImg');
const playerImg = document.getElementById('playerImg');
const smallObjectImg = document.getElementById('smallObjectImg');
const bigObjectImg = document.getElementById('bigObjectImg');
const bgImg = document.getElementById('bgImg');

// Global state
let globalGameScore = 0;
let globalGameTime = 0;
let globalGamePause = false;
let playerData = {};
let globalSmallObjectProp = [];
let globalBigObjectProp = [];
let totalSmallObject = 0;
let totalBigObject = 0;

// Canvas
const canvas = document.getElementById('game-canvas');
const ctx = canvas.getContext('2d');

// Play Game
playBtn.addEventListener('click', () => {
    // input
    const inputName = document.getElementById('inputName').value;
    const inputTime = document.getElementById('inputTime').value;
    const inputObject = document.getElementById('inputObject').value;
    let currentObject = 0;

    if (!inputName) {
        alert('Name must be required');
        return;
    }

    if (!inputTime) {
        alert('Time must be required')
        return;
    }

    if (!inputObject) {
        currentObject = 20;
    } else {
        currentObject = inputObject
    }

    playerData = {
        name: inputName,
        time: Number(inputTime),
        object: Number(currentObject)
    }

    globalGameTime = playerData.time;
    totalSmallObject = Math.floor(4 / 5 * playerData.object);
    totalBigObject = playerData.object - totalSmallObject;

    // page
    homePage.classList.add('inactive');
    gamePage.classList.add('active');

    // Start game
    navbarGame();
    updateScoreProp();
    gameLoop();
});

// Close Game
closeBtn.addEventListener('click', () => {
    window.location.reload();
});

// Pause game
let pauseProp = false;
pauseBtn.addEventListener('click', () => {
    gamePause();
});

document.addEventListener('keydown', (e) => {
    if (e.key === 'p') pauseProp = true;
});

document.addEventListener('keyup', (e) => {
    if (e.key === 'p') pauseProp = false;
});

function gamePauseKey() {
```

```
    if (pauseProp) {
        gamePause();
    }
}

function gamePause() {
    globalGamePause = !globalGamePause;
    pauseBtn.textContent = 'Resume';
    notifPage.classList.add('active');
    notifHead.textContent = 'Game Pause';

    if (globalGamePause === false) {
        pauseBtn.textContent = 'Pause';
        notifPage.classList.remove('active');
        notifHead.textContent = '';
        requestAnimationFrame(gameLoop);
    }
}

// Reset Game
resetBtn.addEventListener('click', () => {
    gameReset();
});

function gameReset() {
    globalSmallObjectProp = [];
    globalBigObjectProp = [];
    globalGameScore = 0;
    globalGameTime = playerData.time;

    playerProp.x = 50;
    playerProp.y = 50;
    enemyProp.x = canvas.width + 50,
    enemyProp.y = canvas.height + 50,

    notifPage.classList.remove('active');
    notifHead.textContent = '';
    notifScore.textContent = '';
    globalGamePause = false;
    navbarGame();
    updateScoreProp();
    requestAnimationFrame(gameLoop);
}

// Navbar Game
function navbarGame() {
    navName.textContent = `Name: ${playerData.name}`;
    gameTime();
}

let timer;
function gameTime() {
    clearInterval(timer);
    timer = setInterval(() => {
```



```
        if (globalGamePause === false) {
            globalGameTime--;
            navTime.textContent = `Time: ${globalGameTime}`;
        }

        if (globalGameTime === 0) {
            globalGamePause = true;
            clearInterval(timer);
            notifPage.classList.add('active');
            notifHead.textContent = 'Time Out';
            notifScore.textContent = `Score: ${globalGameScore}`
        }
    }, 1000);
}

// player key
const keys = {
    w: false,
    a: false,
    s: false,
    d: false,
    ArrowUp: false,
    ArrowLeft: false,
    ArrowDown: false,
    ArrowRight: false,
}

document.addEventListener('keydown', (e) => {
    if (e.key === 'w') keys.w = true;
    if (e.key === 'a') keys.a = true;
    if (e.key === 's') keys.s = true;
    if (e.key === 'd') keys.d = true;
    if (e.key === 'ArrowUp') keys.ArrowUp = true;
    if (e.key === 'ArrowLeft') keys.ArrowLeft = true;
    if (e.key === 'ArrowDown') keys.ArrowDown = true;
    if (e.key === 'ArrowRight') keys.ArrowRight = true;

    updatePlayerAngle();
});

document.addEventListener('keyup', (e) => {
    if (e.key === 'w') keys.w = false;
    if (e.key === 'a') keys.a = false;
    if (e.key === 's') keys.s = false;
    if (e.key === 'd') keys.d = false;
    if (e.key === 'ArrowUp') keys.ArrowUp = false;
    if (e.key === 'ArrowLeft') keys.ArrowLeft = false;
    if (e.key === 'ArrowDown') keys.ArrowDown = false;
    if (e.key === 'ArrowRight') keys.ArrowRight = false;

    updatePlayerAngle();
});

// Player update position
```

```
function updatePlayerMove() {
    if (keys.w || keys.ArrowUp) playerProp.y -= playerProp.speed;
    if (keys.a || keys.ArrowLeft) playerProp.x -= playerProp.speed;
    if (keys.s || keys.ArrowDown) playerProp.y += playerProp.speed;
    if (keys.d || keys.ArrowRight) playerProp.x += playerProp.speed;

    if (playerProp.x > canvas.width) {
        playerProp.x = -playerProp.size
    } else if (playerProp.x + playerProp.size < 0) {
        playerProp.x = canvas.width
    }

    if (playerProp.y > canvas.height) {
        playerProp.y = -playerProp.size
    } else if (playerProp.y + playerProp.size < 0) {
        playerProp.y = canvas.height
    }
}

// angle
function updatePlayerAngle() {
    if (keys.w || keys.ArrowUp) playerProp.angle = 0;
    if (keys.a || keys.ArrowLeft) playerProp.angle = -(Math.PI / 2);
    if (keys.s || keys.ArrowDown) playerProp.angle = Math.PI;
    if (keys.d || keys.ArrowRight) playerProp.angle = Math.PI / 2;

    if (keys.w && keys.d || keys.ArrowUp && keys.ArrowRight) playerProp.angle = Math.PI / 4;
    if (keys.d && keys.s || keys.ArrowRight && keys.ArrowDown) playerProp.angle = 3 * Math.PI / 4;
    if (keys.s && keys.a || keys.ArrowDown && keys.ArrowLeft) playerProp.angle = -(3 * Math.PI / 4);
    if (keys.a && keys.w || keys.ArrowLeft && keys.ArrowUp) playerProp.angle = -(Math.PI / 4);
}

// updateEnemyMove
function updateEnemyMove() {
    const dx = playerProp.x - enemyProp.x;
    const dy = playerProp.y - enemyProp.y;
    const distance = Math.sqrt(dx * dx + dy * dy);

    if (distance > 1) {
        enemyProp.x += (dx / distance) * enemyProp.speed;
        enemyProp.y += (dy / distance) * enemyProp.speed;
    }

    enemyProp.angle = Math.atan2(dy, dx) + Math.PI / 2;
}

// Collusion
function gameCollusion(a, b) {
    return (
        a.x < b.x + b.size &&
        a.x + a.size > b.x &&
        a.y < b.y + b.size &&
        a.y + a.size > b.y
    )
}

function gameCheckPointCollusion() {
    navScore.textContent = `Score: ${globalGameScore}`;
    for (let index = 0; index < globalBigObjectProp.length; index++) {
        const bigObject = globalBigObjectProp[index];
        if (gameCollusion(playerProp, bigObject)) {
            globalGameScore+=20;
            globalBigObjectProp.splice(index, 1);
        }
    }

    for (let index = 0; index < globalSmallObjectProp.length; index++) {
        const smallObject = globalSmallObjectProp[index];
        if (gameCollusion(playerProp, smallObject)) {
            globalGameScore+=5;
            globalSmallObjectProp.splice(index, 1);
        }
    }
}

function gameCheckPlayerEnemyCollusion() {
    if (gameCollusion(playerProp, enemyProp)) {
        globalGamePause = true;
        notifPage.classList.add('active');
        notifHead.textContent = 'Game Over';
        notifScore.textContent = `Score: ${globalGameScore}`
    }
}

// prop
const playerProp = {
    x: 50,
    y: 50,
    size: 50,
    speed: 4,
    angle: Math.PI / 2
}

const enemyProp = {
    x: canvas.width + 50,
    y: canvas.height + 50,
    size: 60,
    speed: 3,
    angle: 0
}

function updateScoreProp() {
    for (let index = 0; index < totalSmallObject; index++) {
        globalSmallObjectProp.push({
```

```
            x: Math.random() * (canvas.width - 100),
            y: Math.random() * (canvas.height - 100),
            size: 25,
        });
    }

    for (let index = 0; index < totalBigObject; index++) {
        globalBigObjectProp.push({
            x: Math.random() * (canvas.width - 100),
            y: Math.random() * (canvas.height - 100),
            size: 50,
        });
    }
}

// drawImage
function drawBackground() {
    ctx.drawImage(bgImg, 0, 0, canvas.width, canvas.height);
}

function drawPlayer() {
    ctx.save();
    ctx.translate(playerProp.x + playerProp.size / 2, playerProp.y + playerProp.size / 2);
    ctx.rotate(playerProp.angle);
    ctx.drawImage(playerImg, -playerProp.size / 2, -playerProp.size / 2, playerProp.size,
playerProp.size);
    ctx.restore();

    updatePlayerMove();
}

function drawEnemy() {
    ctx.save();
    ctx.translate(enemyProp.x + enemyProp.size / 2, enemyProp.y + enemyProp.size / 2);
    ctx.rotate(enemyProp.angle);
    ctx.drawImage(enemyImg, -enemyProp.size / 2, -enemyProp.size / 2, enemyProp.size,
enemyProp.size);
    ctx.restore();

    updateEnemyMove();
}

function drawSmallObject() {
    for (let index = 0; index < globalSmallObjectProp.length; index++) {
        const smallObject = globalSmallObjectProp[index];
        ctx.drawImage(smallObjectImg, smallObject.x, smallObject.y, smallObject.size,
smallObject.size)
    }
}

function drawBigObject() {
    for (let index = 0; index < globalBigObjectProp.length; index++) {
        const bigObject = globalBigObjectProp[index];
```



```
        ctx.drawImage(bigObjectImg, bigObject.x, bigObject.y, bigObject.size,
bigObject.size)
    }
}

// game win
function gameWin() {
    const countBigObject = totalBigObject * 20;
    const countSmallObject = totalSmallObject * 5;
    const total = countBigObject + countSmallObject;

    if (globalGameScore === total) {
        globalGamePause = true;
        notifPage.classList.add('active');
        notifHead.textContent = 'Victory';
        notifScore.textContent = `Score: ${globalGameScore}`
    }
}

// GameLoop
function gameLoop() {
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    // Image
    drawBackground();
    drawBigObject();
    drawSmallObject();
    drawPlayer();
    drawEnemy();

    // status
    gameWin();
    gamePauseKey()

    // Collusion
    gameCheckPointCollusion();
    gameCheckPlayerEnemyCollusion();

    if (globalGamePause === false) {
        requestAnimationFrame(gameLoop);
    }
}
```