```
/*
Hi! I already know coding and tried following coursework requirements as closely as possible, apologizing for any unintentionally advanced implementations used.
I carefully watched the plagiarism warning video because I was scared of getting flagged for academic misconduct.
Please check my GitHub profile (github.com/itzmaniss) to see my prior programming experience and understanding of these concepts.
In case of any issues, I am happy to discuss and clarify my work.
*/

// Game character variables
var gameChar_x = 200;
var gameChar_y = 300;
var gameChar_width = 30;
var velocityX = 0;
var velocityY = 0;
var facing = "front";

// Movement state variables
var isLeft = false;
var isRight = false;
var isFalling = false;
var isWalking = false;
var isPlummeting = false;

// Game world variables
var floorPos_y = 300;
var cameraPosX = 0;
var coinCount = 0;

// Game object arrays
var clouds = [];
var trees = [];
var mountainRange = [];
var collectables = [];
var canyons = [];

// Movement constants
var cloudSpeed = 0.25;

function setup() {
    createCanvas(1024, 576);
    noStroke();

    // Create clouds at different x positions
    for (var i = 0; i < 50; i++) {
        clouds[i] = {x_pos: i * 200 - 300, y_pos: 80};
    }

    // Create trees at different x positions
    for (var i = 0; i < 100; i++) {
        trees[i] = {x_pos: i * 150 - 200, y_pos: 240};
    }

    // Create mountains at different x positions
    for (var i = 0; i < 100; i++) {
        mountainRange[i] = {x_pos: i * 400 - 300, y_pos: 300};
    }

    // Create collectables at different x positions
    for (var i = 0; i < 80; i++) {
        collectables[i] = {x_pos: i * 300 - 100, y_pos: 175, size: 30, isFound: false};
    }

    // Create canyons at different x positions
    for (var i = 0; i < 40; i++) {
        canyons[i] = {x_pos: i * 800 + 400, width: 85};
    }
}

function draw() {
    // Update camera to follow character
    cameraPosX = gameChar_x - width/2;

    // Draw sky background
    background(135, 206, 235);

    // Draw extended ground
    fill(34, 139, 34);
    rect(-width, floorPos_y, width * 3, height - floorPos_y);

    // Apply camera transformation for world objects
    push();
    translate(-cameraPosX, 0);

    // Draw mountains
    for (var i = 0; i < mountainRange.length; i++) {
        if (mountainRange[i].x_pos > cameraPosX - 200 && mountainRange[i].x_pos < cameraPosX + width + 200) {
            fill(125, 125, 125);
            noStroke();
            triangle(mountainRange[i].x_pos - 60, mountainRange[i].y_pos, mountainRange[i].x_pos, mountainRange[i].y_pos - 240, mountainRange[i].x_pos + 60, mountainRange[i].y_pos);
            triangle(mountainRange[i].x_pos - 120, mountainRange[i].y_pos, mountainRange[i].x_pos - 60, mountainRange[i].y_pos - 180, mountainRange[i].x_pos, mountainRange[i].y_pos);
            triangle(mountainRange[i].x_pos, mountainRange[i].y_pos, mountainRange[i].x_pos + 60, mountainRange[i].y_pos - 150, mountainRange[i].x_pos + 120, mountainRange[i].y_pos);
        }
    }

    // Draw and animate clouds
    for (var i = 0; i < clouds.length; i++) {
        if (clouds[i].x_pos > cameraPosX - 200 && clouds[i].x_pos < cameraPosX + width + 200) {
            fill(255);
            noStroke();
            ellipse(clouds[i].x_pos, clouds[i].y_pos, 60, 40);
            ellipse(clouds[i].x_pos - 20, clouds[i].y_pos + 10, 50, 30);
            ellipse(clouds[i].x_pos + 20, clouds[i].y_pos + 10, 50, 30);
            ellipse(clouds[i].x_pos - 10, clouds[i].y_pos - 10, 40, 20);
            ellipse(clouds[i].x_pos + 10, clouds[i].y_pos - 10, 40, 20);
        }

        // Move clouds to the left
        clouds[i].x_pos = clouds[i].x_pos - cloudSpeed;

        // Reset cloud position when it goes off screen
        if (clouds[i].x_pos < -100) {
            clouds[i].x_pos = clouds[i].x_pos + 10000;
        }
    }

    // Draw trees
    for (var i = 0; i < trees.length; i++) {
        if (trees[i].x_pos > cameraPosX - 200 && trees[i].x_pos < cameraPosX + width + 200) {
            fill(139, 69, 19);
            rect(trees[i].x_pos, trees[i].y_pos, 20, 60);
            fill(34, 139, 34);
            ellipse(trees[i].x_pos + 10, trees[i].y_pos - 30, 60, 60);
            ellipse(trees[i].x_pos - 20, trees[i].y_pos - 10, 50, 50);
            ellipse(trees[i].x_pos + 40, trees[i].y_pos - 10, 50, 50);
            ellipse(trees[i].x_pos + 10, trees[i].y_pos - 50, 40, 40);
            ellipse(trees[i].x_pos + 10, trees[i].y_pos - 70, 40, 40);
        }
    }

    // Draw collectable coins
    for (var i = 0; i < collectables.length; i++) {
        if (collectables[i].isFound == false && collectables[i].x_pos > cameraPosX - 200 && collectables[i].x_pos < cameraPosX + width + 200) {
            fill(255, 215, 0, 100);
            ellipse(collectables[i].x_pos, collectables[i].y_pos, 34, 34);
            fill(255, 215, 0);
            ellipse(collectables[i].x_pos, collectables[i].y_pos, 30, 30);
            fill(255, 235, 0);
            ellipse(collectables[i].x_pos, collectables[i].y_pos, 24, 24);
            fill(255, 255, 200);
            push();
            translate(collectables[i].x_pos - 5, collectables[i].y_pos - 5);
            rotate(PI / 4);
            ellipse(0, 0, 5, 12);

            pop();
            textAlign(CENTER, CENTER);
            textSize(16);
            fill(0);
            text('$', collectables[i].x_pos, collectables[i].y_pos);
        }
    }

    // Draw canyons
    for (var i = 0; i < canyons.length; i++) {
        if (canyons[i].x_pos > cameraPosX - 200 && canyons[i].x_pos < cameraPosX + width + 200) {
            fill(40, 40, 40);
            rect(canyons[i].x_pos, floorPos_y, canyons[i].width, height - floorPos_y);
            fill(139, 69, 19);
            rect(canyons[i].x_pos - 5, floorPos_y, 5, height - floorPos_y);
            rect(canyons[i].x_pos + canyons[i].width, floorPos_y, 5, height - floorPos_y);
            for(var j = 0; j < 5; j++) {
                fill(0, 0, 0, 50 - j * 10);
                rect(canyons[i].x_pos, floorPos_y + j * 5, canyons[i].width, 5);
            }
        }
    }

    // Check coin collection
    for (var i = 0; i < collectables.length; i++) {
        if (collectables[i].isFound == false) {
            var charLeft = gameChar_x - 15;
            var charRight = gameChar_x + 15;
            var charTop = gameChar_y - 60;
            var charBottom = gameChar_y;

            var coinLeft = collectables[i].x_pos - 15;
            var coinRight = collectables[i].x_pos + 15;
            var coinTop = collectables[i].y_pos - 15;
            var coinBottom = collectables[i].y_pos + 15;

            if (charRight > coinLeft && charLeft < coinRight && charBottom > coinTop && charTop < coinBottom) {
                collectables[i].isFound = true;
                coinCount = coinCount + 1;
            }
        }
    }

    // Check canyon collision
    for (var i = 0; i < canyons.length; i++) {
        if (gameChar_x > canyons[i].x_pos && gameChar_x < canyons[i].x_pos + canyons[i].width && gameChar_y >= floorPos_y && isPlummeting == false) {
            isPlummeting = true;
        }
    }

    // Handle canyon falling
    if (isPlummeting == true) {
        gameChar_y = gameChar_y + 5;
        velocityX = 0;
        velocityY = 0;

        for (var i = 0; i < canyons.length; i++) {
            if (gameChar_x > canyons[i].x_pos && gameChar_x < canyons[i].x_pos + canyons[i].width) {
                if (gameChar_x < canyons[i].x_pos) {
                    gameChar_x = canyons[i].x_pos;
                }
                if (gameChar_x > canyons[i].x_pos + canyons[i].width) {
                    gameChar_x = canyons[i].x_pos + canyons[i].width;
                }
            }
        }
    } else {
        // Apply gravity and movement
        velocityY = velocityY + 0.8;
        gameChar_x = gameChar_x + velocityX;
        gameChar_y = gameChar_y + velocityY;

        // Check ground collision
        if (gameChar_y >= floorPos_y) {
            gameChar_y = floorPos_y;
            velocityY = 0;
            velocityX = 0;
            isFalling = false;
        }
    }

    // Draw character based on current state
    if (isFalling == true) {
        if (facing === "right") {
            push();
            translate(gameChar_x, gameChar_y);
            fill('#FFFF00');
            ellipse(0, -35, 35, 35);
            ellipse(3, -60, 30, 30);
            fill('#FFFFFF');
            ellipse(11, -62, 10, 10);
            fill('#00BFFF');
            ellipse(12, -62, 7, 7);
            fill('#000000');
            ellipse(13, -62, 4, 4);
            fill("#BA8E23");
            triangle(16, -62, 23, -58, 16, -54);
            stroke(0);
            strokeWeight(3);
            line(-5, -18, -3, -10);
            line(-3, -10, 2, -10);
            noStroke();
            pop();
        } else if (facing === "left") {
            push();
            translate(gameChar_x, gameChar_y);
            scale(-1, 1);
            fill('#FFFF00');
            ellipse(0, -35, 35, 35);
            ellipse(3, -60, 30, 30);
            fill('#FFFFFF');
            ellipse(11, -62, 10, 10);
            fill('#00BFFF');
            ellipse(12, -62, 7, 7);
            fill('#000000');
            ellipse(13, -62, 4, 4);
            fill("#BA8E23");
            triangle(16, -62, 23, -58, 16, -54);
            stroke(0);
            strokeWeight(3);
            line(-5, -18, -3, -10);
            line(-3, -10, 2, -10);
            noStroke();
            pop();
        } else {
            push();
            translate(gameChar_x, gameChar_y);
            fill('#FFFF00');
            ellipse(0, -35, 35, 35);
            ellipse(0, -60, 30, 30);
            fill('#FFFFFF');
            ellipse(-6, -62, 8, 8);
            fill('#00BFFF');
            ellipse(-6, -62, 6, 6);
            fill('#000000');
            ellipse(-6, -62, 3, 3);
            fill('#FFFFFF');
            ellipse(6, -62, 8, 8);
            fill('#00BFFF');
            ellipse(6, -62, 6, 6);
            fill('#000000');
            ellipse(6, -62, 3, 3);
            fill("#BA8E23");
            triangle(0, -55, -4, -50, 4, -50);
            stroke(0);
            strokeWeight(3);
            noFill();
            line(-8, -18, -12, -10);
            line(-12, -10, -15, -10);
            line(8, -18, 12, -10);
            line(12, -10, 15, -10);
            pop();
        }
    } else if (facing === "left") {
        push();
        translate(gameChar_x, gameChar_y);
        scale(-1, 1);
        fill('#FFFF00');
        ellipse(-3, -25, 35, 35);
        ellipse(0, -50, 30, 30);
        fill('#FFFFFF');
        ellipse(8, -52, 10, 10);
        fill('#00BFFF');
        ellipse(9, -52, 7, 7);
        fill('#000000');
        ellipse(10, -52, 4, 4);
        fill("#BA8E23");
        triangle(13, -52, 20, -48, 13, -44);
        stroke(0);
        strokeWeight(3);
        line(-8, -8, -8, -2);
```

```
            line(-8, -2, -3, -2);
            noStroke();
            pop();
        } else if (facing === "right") {
            push();
            translate(gameChar_x, gameChar_y);
            fill('#FFFF00');
            ellipse(-3, -25, 35, 35);
            ellipse(0, -50, 30, 30);
            fill('#FFFFFF');
            ellipse(8, -52, 10, 10);
            fill('#00BFFF');
            ellipse(9, -52, 7, 7);
            fill('#000000');
            ellipse(10, -52, 4, 4);
            fill("#BA8E23");
            triangle(13, -52, 20, -48, 13, -44);
            stroke(0);
            strokeWeight(3);
            line(-8, -8, -8, -2);
            line(-8, -2, -3, -2);
            noStroke();

            pop();
        } else {
            push();
            translate(gameChar_x, gameChar_y);
            fill('#FFFF00');
            ellipse(0, -25, 35, 35);
            ellipse(0, -50, 30, 30);
            fill('#FFFFFF');
            ellipse(-6, -52, 8, 8);
            fill('#00BFFF');
            ellipse(-6, -52, 6, 6);
            fill('#000000');
            ellipse(-6, -52, 3, 3);
            fill('#FFFFFF');
            ellipse(6, -52, 8, 8);
            fill('#00BFFF');
            ellipse(6, -52, 6, 6);
            fill('#000000');
            ellipse(6, -52, 3, 3);
            fill("#BA8E23");
            triangle(0, -45, -4, -40, 4, -40);
            stroke(0);
            strokeWeight(3);
            noFill();
            line(-8, -8, -15, -2);
            line(-15, -2, -18, -2);
            line(8, -8, 15, -2);
            line(15, -2, 18, -2);
            noStroke();
            pop();
        }

        // End camera transformation
        pop();

        // Draw UI elements (coin counter)
        fill(255, 215, 0);
        textAlign(LEFT, TOP);
        textSize(24);
        text('Coins: ' + coinCount, 20, 20);

        // Handle horizontal movement
        if (isPlummeting == false) {
            if (isLeft == true) {
                velocityX = -5;
            } else if (isRight == true) {
                velocityX = 5;
            }
        }
    }

    function keyPressed() {
        if (isPlummeting == false) {
            if (key === 'A' || key === 'a' || keyCode == 37) {
                if (isRight == false) {
                    isLeft = true;
                    facing = "left";
                    isWalking = true;
                }
            } else if (key === 'D' || key === 'd' || keyCode == 39) {
                if (isLeft == false) {
                    isRight = true;
                    facing = "right";
                    isWalking = true;
                }
            } else if (key === ' ' || keyCode == 38) {
                if (isFalling == false) {
                    velocityY = -12;
                    isFalling = true;
                    isWalking = false;
                }
            }
        }
```

```
}

function keyReleased() {
    if (isPlummeting == false) {
        if (key === 'A' || key === 'a' || keyCode == 37) {
            isLeft = false;
            if (isRight == true) {
                facing = "right";
                isWalking = true;
            } else {
                velocityX = 0;
                isWalking = false;
                facing = "front";
            }
        } else if (key === 'D' || key === 'd' || keyCode == 39) {
            isRight = false;
            if (isLeft == true) {
                facing = "left";
                isWalking = true;
            } else {
                velocityX = 0;
                isWalking = false;
                facing = "front";
            }
        }
    }
}
```