sketch.js

```
/* Project part 4 - Midterms
Student: Timna Aversa
Class: Introduction to Programming */


var floorPos_y = 432;
var max_x = 2000;
var cameraPosX;

var gameChar_x;
var gameChar_y;

var trees_x;

var canyon;
var collectable;
var mountain;
var cloud;

var isLeft;
var isRight;
var isFalling;
var isPlummeting;

// I know the video commented that having things separated by functions was one of the give
// ways for plagiarism but since my code was already organized in this set up 3 hours didn't
// seem like enough time to rewrite it all and make sure it worked. I did wrote the whole thing!

//character set up
//Character hair set up
function hair(gameChar_x,gameChar_y,size)
{
    fill(210, 83, 128);
    ellipse(gameChar_x - 0.18 * size,gameChar_y- 4.6 * size,0.45 * size,0.45 * size);
    ellipse(gameChar_x - 0.4 * size,gameChar_y- 4.3 * size,0.4 * size,0.4 * size);
    ellipse(gameChar_x,gameChar_y- 4.3 * size,0.5 * size,0.4 * size);
    ellipse(gameChar_x + 0.18 * size,gameChar_y- 4.6 * size,0.45 * size,0.45 * size);
    ellipse(gameChar_x + 0.4 * size,gameChar_y- 4.3 * size,0.4 * size,0.4 * size);

}
//Character eye set up
function eye(gameChar_x,gameChar_y,size,direction)
{
    dirNum = 0;
    if (direction === "left")
    {
        dirNum = -3;
    }
    if (direction === "right")
    {
        dirNum = 3;
    }
    fill(255);
    ellipse(gameChar_x + dirNum ,gameChar_y -3.1 * size,2.3 * size,2.3 * size);
    fill(0);
    ellipse(gameChar_x + dirNum * 2.4,gameChar_y - 3.1 * size+ dirNum * 0.2,1.4 * size,1.4 * size);
    fill(255);
    ellipse(gameChar_x + dirNum * 3.2,gameChar_y - 3.25 * size,0.7 * size,0.7 * size);
    ellipse(gameChar_x - 2 + dirNum,gameChar_y - 2.7 * size,0.3 * size,0.3 * size);
}
//Character mouth set up
function mouth(gameChar_x,gameChar_y,size,direction)
{
    dirNum = 0;
    if (direction === "left")
    {
        dirNum = -4.2;
    }
    if (direction === "right")
    {
        dirNum = 4.2;
    }
    triangle(gameChar_x - 0.8 * size + dirNum,
            gameChar_y - 1.3  * size,
            gameChar_x - 0.6 * size +dirNum,
            gameChar_y - 1.6 * size,
            gameChar_x - 0.4 * size+dirNum,
            gameChar_y - 1.3 * size);
    triangle(gameChar_x + 0.8 * size + dirNum,
            gameChar_y - 1.3 * size,
            gameChar_x + 0.6 * size + dirNum,
            gameChar_y - 1.6 * size,
            gameChar_x + 0.4 * size + dirNum,
            gameChar_y - 1.3 * size);
    stroke(147, 118, 224);
    line(gameChar_x - 0.8 * size + dirNum,
        gameChar_y - 1.3 * size,
        gameChar_x + 0.8 * size + dirNum,
        gameChar_y - 1.3 * size);
}
//Character head set up
function headBackground(gameChar_x,gameChar_y,size)
{
    rightHorn(gameChar_x,gameChar_y,size);
    leftHorn(gameChar_x,gameChar_y,size);
    fill(255,116,177);
    ellipse(gameChar_x,gameChar_y-2.4 * size,4 * size,4 * size);
    noStroke();
    fill(128, 70, 116, 60);
    arc(gameChar_x,gameChar_y-2.4 * size,4 * size,4 * size,4.6,1.2,24);
    fill(255,116,177);
    ellipse(gameChar_x,gameChar_y-2.4 * size,3.4 * size,4 * size);
}
//Character position of the leg for standing still set up
function stand(gameChar_x,gameChar_y,size,side)
{
    beginShape();
    vertex(gameChar_x + 0.3 * size * side, gameChar_y);
    vertex(gameChar_x + 0.35 * size * side, gameChar_y - 0.5 * size);
    vertex(gameChar_x + 0.8 * size * side, gameChar_y - 0.7 * size);
    vertex(gameChar_x + 1 * size * side, gameChar_y);
    endShape(CLOSE);
}
// Character making the call for each function according to the leg movement, call
// made for both legs.
function legs(gameChar_x,gameChar_y,size,action,direction)
{
    fill(255,116,177);
    noStroke();
    dirNum = 0;
    if (action === "stand")
    {
        stand(gameChar_x,gameChar_y,size,-1);
        stand(gameChar_x,gameChar_y,size,1);
    }
    if (action === "jump")
    {
        jump(gameChar_x,gameChar_y,size,-1,0);
        jump(gameChar_x,gameChar_y,size,1,0);
    }
    if (action === "walk" & direction === "left")
    {
        walk(gameChar_x,gameChar_y,size,1, -12);
        walk(gameChar_x,gameChar_y,size,1, 0);
    }
    if (action === "walk" & direction === "right")
    {
        walk(gameChar_x,gameChar_y,size,-1, +12);
        walk(gameChar_x,gameChar_y,size,-1, 0);
    }
}
//Character position of the leg for walking set upx
function walk(gameChar_x,gameChar_y,size,side,direction)
{
    beginShape();
    vertex(gameChar_x + 0.6 * size * side + direction, gameChar_y);
    vertex(gameChar_x + 0.35 * size * side + direction, gameChar_y - 0.2 * size);
    vertex(gameChar_x + 0.4 * size * side + direction, gameChar_y - 0.5 * size);
    vertex(gameChar_x + 0.9 * size * side + direction, gameChar_y - 0.6 * size);
    vertex(gameChar_x + 1 * size * side + direction, gameChar_y - 0.4 * size);
    vertex(gameChar_x + 1.4 * size * side + direction, gameChar_y- 0.2 * size);
    endShape(CLOSE);
}
//Character position of the leg for jumping and falling set up
function jump(gameChar_x,gameChar_y,size,side,direction)
{
    beginShape();
    stroke(128, 70, 116, 60);
    vertex(gameChar_x + 0.9 * size * side + direction, gameChar_y - 0.2 * size +
direction);
    vertex(gameChar_x + 0.8 * size * side + direction, gameChar_y - 0.55 * size +
direction);
    vertex(gameChar_x + 1.3 * size * side + direction, gameChar_y - 1 * size +
direction);
    vertex(gameChar_x + 2 * size * side + direction, gameChar_y - 0.7 * size +
direction);
    endShape(CLOSE);
    noStroke();
}
//Character right horn set up
function rightHorn(gameChar_x,gameChar_y,size)
```

```
{
    fill(255);
    arc(gameChar_x + 0.15 * size,gameChar_y-3.8 * size,3 * size,3 * size,6,2, PI);
}
//Character left horn set up
function leftHorn(gameChar_x,gameChar_y,size)
{
    fill(255);
    arc(gameChar_x - 0.15 * size,gameChar_y-3.8 * size,3 * size,3 * size,45,3.5,
PI);
}
// Call functions to draw all parts of the character
function characterMove(gameChar_x,gameChar_y,size,move,direction)
{
    legs(gameChar_x,gameChar_y,size,move,direction);
    headBackground(gameChar_x,gameChar_y,size);
    hair(gameChar_x,gameChar_y,size);
    eye(gameChar_x,gameChar_y,size, direction);
    mouth(gameChar_x,gameChar_y,size, direction);
}
//game view
//Random generation of numbers function for simplification of code
function randNumb(maxNumber)
{
    return Math.floor(Math.random() * maxNumber);
}
// Draw a tree
function tree(x,y,size)
{
    noStroke();
    y=432-size*6;
    fill(60, 35, 23);
    rect(x-size,y-size*2,size*2,size*8);
    fill(46, 79, 79);
    stroke(44, 51, 51);
    triangle(x+size*10,y,x,y- size*26,x-size*10,y);
    noStroke();
    fill(44, 51, 51);
    beginShape();
    vertex(x+size*8, y-size*5);
    vertex(x, y-size*13);
    vertex(x-size*8, y-size*5);
    vertex(x,y- size*11);
    endShape();
}
// Draw a mountain
function mountain(x,y,size)
{
    y=432;
    fill(212, 173, 252);
    triangle(x,y,x,y- size*26,x+size*10,y);
    fill(92, 70, 156);
    triangle(x,y,x,y- size*26,x-size*10,y);
    fill(255);
    beginShape();
    vertex(x, y-size*18);
    vertex(x, y-size*26);
    vertex(x+size*3, y- size*18);
    vertex(x+size/2,y- size*20);
    endShape();
    fill(210);
    beginShape();
    vertex(x, y-size*18);
    vertex(x, y-size*26);
    vertex(x-size*3, y- size*18);
    vertex(x-size/2,y- size*20);
    endShape();
}
// Draw a star
function star(x,y,size)
{
    i = randNumb(5)
    if (i==0){fill(255, 95, 158);}
    else if (i==1){fill(233, 0, 100);}
    else if (i==2){fill(249, 217, 73);}
    else if (i==3){fill(240, 240, 240);}
    else if (i==4){fill(58, 180, 242);}
    else{fill(39, 225, 193);}
    ellipse(x,y,1,1);
}
// Draw a rock
function rock(x,y,size)
{
    // console.log(y)
    size = size + 2;
    i = size % 3
    if (i==0){fill(65, 53, 67,50);}
    else if (i==1){fill(240, 235, 141,50);}
    else{fill(143, 67, 238, 50);}
    ellipse(x,floorPos_y + 25 + y,size,size-3);
}
// Draw a cloud
function cloud(x,y,size)
{
    noStroke();
    fill(255);
    rect(x + size * 4, y + size/2 * 10, size * 13, size * 3, size*2);
    rect(x + size * 10, y + size/2 *5, size * 5, size * 4, size*2);
    rect(x + size * 7, y + size, size * 4.5, size * 6, size*2);
}
// Draw a token
function token(x,y,size)
{
    noStroke();
    fill(255, 211, 163);
    triangle(x - size,y,x,y + 2.5 * size,x + size,y);
    fill(225, 18, 153);
    arc(x, y-1, 2.3 * size, 3.3  * size, PI, 0 , CHORD);
}
// Draw a canyon
function canyonDraw(x,size)
{
    noStroke();
    fill(169, 113, 85);
    rect(x,floorPos_y,size,200);
}
//initiating classes of random objects positioning
class positionClass
{
    constructor(maxX,maxY,maxSize) {
        this.x = randNumb(maxX);
        this.y = randNumb(maxY);
        this.size = randNumb(maxSize);
    }
}
// object to control the random generated objects
let sceneryObjs = {rocks:{'rand':850,'x':max_x,'y':120,'size':8,'obj':[]},
                    stars:{'rand':1050,'x':max_x,'y':425,'size':8,'obj':[]},
                    mountains:{'rand':7,'x':max_x,'y':300,'size':8,'obj':[]},
                    trees:{'rand':8,'x':max_x,'y':0,'size':6,'obj':[]},
                    clouds:{'rand':7,'x':max_x,'y':100,'size':8,'obj':[]}
                };

// Because I wanted something that would look prettier I had created a code in which
// the positions for the items are randomly generated and assigned to an array, since
// the request was for generate an array of objects for the clouds and mountains I believe
// this code still fulfill the purpose. So I only recreated an array for trees.
function drawObjectsInArray(array_obj, obj_type)
{
    for (let i = 0; i < array_obj.length; i++)
    {
        switch (obj_type) {
            // case 'trees':
            //  tree(array_obj[i].x,array_obj[i].y,array_obj[i].size);
            //  break;
            case 'rocks':
                rock(array_obj[i].x,array_obj[i].y,array_obj[i].size);
                break;
            case 'stars':
                star(array_obj[i].x,array_obj[i].y,array_obj[i].size);
                break;
            case 'clouds':
                cloud(array_obj[i].x,array_obj[i].y,array_obj[i].size);
                break;
            case 'mountains':
                mountain(array_obj[i].x,array_obj[i].y,array_obj[i].size);
                break;
            default:
        }

    }
}
//game main
function setup()
{
    createCanvas(1024, 576);
    cameraPosX = 0;
    gameChar_x = 200;
    gameChar_y = floorPos_y;

    isLeft = false;
    isRight = false;
    isFalling = false;
    isPlummeting = false;
    // making the requested array for the trees
    trees_x = [50,300,500,800,1000,1200];

    //generate objects in the arrays
    Object.entries(sceneryObjs).forEach(([key, value]) => {
        for (let j = 0; j < value.rand; j++)
        {
            sceneryObjs[key].obj.push(new 
positionClass(value.x,value.y,value.size));
        }
    });

    canyon = {x_pos: width - 400, width: 70}
    collectable = {
        x_pos: width - 150,
        y_pos: 410,
        size: 7,
        isFound: false
    }
}

function draw()
{
    background(6, 0, 71);
    noStroke();
    fill(26, 95, 122);
    rect(0, 432, 1024, 20);
    fill(5, 45, 72);
    rect(0,452,1024, 120);
    push();
    translate(-cameraPosX, 0);
    // drawing each object in the arrays
    Object.entries(sceneryObjs).forEach(([key, value]) => {
        drawObjectsInArray(sceneryObjs[key].obj,key);
    });
    // looping through the array of trees positions
    for(var i = 0; i < trees_x.length; i++)
    {
        tree(trees_x[i],floorPos_y,5);
    }

    canyonDraw(canyon.x_pos,canyon.width);
    // controlling canyon and token interaction with character
    if (dist(cameraPosX + gameChar_x,gameChar_y,collectable.x_pos,
collectable.y_pos) < 25)
    {
        collectable.isFound = true;
    }
    if (collectable.isFound == false)
    {
        token(collectable.x_pos,collectable.y_pos,collectable.size);
    }
    if((canyon.x_pos + canyon.width > cameraPosX + gameChar_x) && (cameraPosX + 
gameChar_x > canyon.x_pos) && gameChar_y >= floorPos_y)
    {
        isPlummeting = true;
    }

    //character and camera position control
    if (isPlummeting == true)
    {
        gameChar_y += 8;
    }
    else if (isLeft == true)
    {
            cameraPosX -= 5;
        }
        else if (isRight == true)
        {
            cameraPosX += 5;
        }

        if (gameChar_y < floorPos_y)
        {
            gameChar_y += 4;
            isFalling = true;
        }
        else
        {
            isFalling = false;
        }

        pop();

        //character design
        if (isLeft && isFalling)
        {
            characterMove(gameChar_x,gameChar_y,10,'jump','left');
        }
        else if (isRight && isFalling)
        {
            characterMove(gameChar_x,gameChar_y,10,'jump','right');
        }
        else if (isLeft)
        {
            characterMove(gameChar_x,gameChar_y,10,'walk','left');
        }
        else if (isRight)
        {
            characterMove(gameChar_x,gameChar_y,10,'walk','right');
        }
        else if (isFalling || isPlummeting)
        {
            characterMove(gameChar_x,gameChar_y,10,'jump','');
        }
        else
        {
            characterMove(gameChar_x,gameChar_y,10,'stand','');
        }

}
// user actions control
function keyPressed()
{
    if (isPlummeting == false)
    {
        if (keyCode == 37)
        {
            isLeft = true;
        }
        else if (keyCode == 39)
        {
            isRight = true;
        }
        else if ((keyCode == 32) && (isFalling != true))
        {
            gameChar_y -= 100;
        }
        else
        {
            console.log(key)
        }
    }

}
function keyReleased()
{
    if (keyCode == 37)
    {
        isLeft = false;
    }
    else if (keyCode == 39)
    {
        isRight = false;
    }
}

```