```
const canvas = document.querySelector('canvas')
const c = canvas.getContext('2d')

canvas.width = 960
canvas.height = 640


//
// SPLITING THE COLLISION AND ENCOUNTER ARRAYS INTO ROWS AND COLUMNS=========
// (from the data folder)

//create an empty array called collisionsMap
const collisionsMap = []
//iterating over the collisions array
for (var i = 0; i < collisions.length; i += 80) {
    //slicing the collisions array by 70 items and adding each slice to the collisionsMap array
    collisionsMap.push(collisions.slice(i, i + 80))
}

const encountersMap = []
//iterating over the encounters array
for (var i = 0; i < encountersData.length; i += 80) {
    //slicing the collisions array by 70 items and adding each slice to the collisionsMap array
    encountersMap.push(encountersData.slice(i, i + 80))
}

//Create an empty array called boundaries where the boundry blocks will be added
const boundaries = []
//An object that has two properties x and y which are used for offsetting the drawing position.
const offset = {
    x: -1132,
    y: -300
}


//iterating over the collisionsMap array
collisionsMap.forEach((row, i) => {
    //iterating over each sub array of the collisionsMap
    row.forEach((symbol, j) => {
        //If the symbol is 5504, the id for the collisionn block,
        //a new boundary object is created with the specified positions
        if (symbol === 5504)
            boundaries.push(
                new Boundary({
                    position: {
                        x: j * Boundary.width + offset.x,
                        y: i * Boundary.height + offset.y
                    }
                })
            )
    })
})

const encounters = []
//iterating over the encountersMap array
encountersMap.forEach((row, i) => {
    //iterating over each sub array of the encountersMap
    row.forEach((symbol, j) => {
        //If the symbol is 5504, the id for the encounter block,
        //a new grass object is created with the specified positions
        if (symbol === 5504)
            encounters.push(
                new Boundary({
                    position: {
                        x: j * Boundary.width + offset.x,
                        y: i * Boundary.height + offset.y
                    }
```

```
                })
            )
    })
})

//
// ============================================================================
//

//create new Image elements and variables for all the images used---------
const backgroundImage = new Image()
backgroundImage.src = './img/StarterTown.png'

const foregroundImage = new Image()
foregroundImage.src = './img/foregroundObjects.png'

const playerUpImage = new Image()
playerUpImage.src = './img/mayUp.png'

const playerDownImage = new Image()
playerDownImage.src = './img/mayDown.png'

const playerLeftImage = new Image()
playerLeftImage.src = './img/mayLeft.png'

const playerRightImage = new Image()
playerRightImage.src = './img/mayRight.png'

//----------------------------------------------------------------------------

// Create a new player sprite object
const player = new Sprite({
    position: {
        x: canvas.width / 2 - 32,
        y: canvas.height / 2 - 48
    },
    // start the player facig down, asign that sprite to the player
    image: playerDownImage,
    // Specify the number of horizontal frames in the sprite
    frames: {
        max: 4
    },
    // add all four direction sprites to the player sprite
    sprites: {
        up: playerUpImage,
        down: playerDownImage,
        left: playerLeftImage,
        right: playerRightImage
    },
})

// Create new sprite object for background
const background = new Sprite({
    position: {
        x: offset.x,
        y: offset.y
    },
    // assign backgroundImage to the image property
    image: backgroundImage
})

// Create new sprite object for foreground
const foreground = new Sprite({
    position: {
        x: offset.x,
        y: offset.y
    },
    // assign foregroundImage to the image property
```

```
    image: foregroundImage
})

//Create an object called keys that stores if the respective key is pressed
const keys = {
    w: {
        pressed: false
    },
    a: {
        pressed: false
    },
    s: {
        pressed: false
    },
    d: {
        pressed: false
    }
}
// create an array that contains every movables object
const movables = [background, ...boundaries, foreground, ...encounters]


// The collision detection
function rectCollision({ rect1, rect2 }) {
    // Set the the hit box to be shorter
    // so the player's head covers objects behind it.
    return (
        rect1.position.x + rect1.width - 4 >= rect2.position.x &&
        rect1.position.x + 4 <= rect2.position.x + rect2.width &&
        rect1.position.y + 56 <= rect2.position.y + rect2.height &&
        rect1.position.y + rect1.height - 8 >= rect2.position.y

    )
}

const battle = {
    initiated: false
}

// Everything that needs to update every frame in the world goes here
function animate() {
    //this line calls animate again looping the function
    const animationID = window.requestAnimationFrame(animate)
    //draw the background, player, boundries(for debug) and foreground
    background.draw()
    boundaries.forEach((boundary) => {
        boundary.draw()
    })
    encounters.forEach((encounter) => {
        encounter.draw()
    })
    player.draw()
    foreground.draw()
    //declare a miving variable to control when player gets to move
    let moving = true
    player.moving = false

    //
    //ENCOUNTER LOGIC=================================================
    //
    //if already in a battle dont start a battle
    if (battle.initiated) return

    //activate a battle if the player is walking in grass, with a small chance finding an enemy
    //Once the player enounter an enemy flash a black screen and switch to the battle scene
    if (keys.w.pressed || keys.s.pressed || keys.a.pressed || keys.d.pressed) {
        for (var i = 0; i < encounters.length; i++) {
            const encounter = encounters[i]
```

```
        //Check for collisions
        if (rectCollision({ rect1: player, rect2: encounter }) && Math.random() < 0.01) {
            //stop the current animation loop
            window.cancelAnimationFrame(animationID)
            // play the battle initiated animation using the gsap libraby
            battle.initiated = true
            gsap.to('#overlappingDiv', {
                opacity: 1,
                repeat: 4,
                yoyo: true,
                duration: 0.3,
                onComplete() {
                    gsap.to('#overlappingDiv', {
                        opacity: 1,
                        duration: 0.3,
                        onComplete() {
                            //activate a new animation loop
                            initBattle()
                            animateBattle()
                            gsap.to('#overlappingDiv', {
                                opacity: 0,
                                duration: 0.3
                            })

                        }
                    })
                }
            })
            break
        }
    }
}
//
//=========================================================
//
//
//PLAYER MOVEMENT=========================================
//
console.log("OUT")
//listen for the w,a,s,d keys and if pressed, move the player up,left,down, or right respectively
//also check for collisions every time the player movers
if (keys.w.pressed && lastKey === 'w') {
    player.moving = true
    player.image = player.sprites.up
    for (var i = 0; i < boundaries.length; i++) {
        const boundary = boundaries[i]
        //Check for collisions
        if (
            rectCollision({
                rect1: player,
                rect2: { ...boundary,
                    position: {
                        x: boundary.position.x,
                        y: boundary.position.y + 5
                    }
                }
            })
        ) {
            moving = false;
            break
        }
    }

    if (moving)
        movables.forEach((movable) => {
            movable.position.y += 5
        })
} else if (keys.a.pressed && lastKey === 'a') {
```

```
    player.moving = true
    player.image = player.sprites.left
    for (var i = 0; i < boundaries.length; i++) {
        const boundary = boundaries[i]
        if (
            rectCollision({
                rect1: player,
                rect2: { ...boundary,
                    position: {
                        x: boundary.position.x + 5,
                        y: boundary.position.y
                    }
                }
            })
        ) {
            moving = false;
            break
        }
    }

    if (moving)
        movables.forEach((movable) => {
            movable.position.x += 5
        })
} else if (keys.s.pressed && lastKey === 's') {
    player.moving = true
    player.image = player.sprites.down
    for (var i = 0; i < boundaries.length; i++) {
        const boundary = boundaries[i]
        if (
            rectCollision({
                rect1: player,
                rect2: { ...boundary,
                    position: {
                        x: boundary.position.x,
                        y: boundary.position.y - 5
                    }
                }
            })
        ) {
            moving = false;
            break
        }
    }

    if (moving)
        movables.forEach((movable) => {
            movable.position.y -= 5
        })
} else if (keys.d.pressed && lastKey === 'd') {
    player.moving = true
    player.image = player.sprites.right
    for (var i = 0; i < boundaries.length; i++) {
        const boundary = boundaries[i]
        if (
            rectCollision({
                rect1: player,
                rect2: { ...boundary,
                    position: {
                        x: boundary.position.x - 5,
                        y: boundary.position.y
                    }
                }
            })
        ) {
            moving = false;
            break
        }
```

```
        }

        if (moving)
            movables.forEach((movable) => {
                movable.position.x -= 5
            })
    }
}
//
//=============================================================
//

//Debuggig
animate()
// initBattle()
// animateBattle()


document.querySelector('#dialogBox').addEventListener('click', (e) => {
    //if there is somehting in the queue
    if (queue.length >= 1) {
        // call the first item in the queue
        queue[0]()
        //remove that item from the array
        queue.shift()
    } //once all attacks are done, the player can continue
    else {
        e.currentTarget.style.display = 'none'
    }
})


//
//=============================================================================
//

//Listening to key inputs
let lastKey = ''
window.addEventListener('keydown', (e) => {
    switch (e.key) {
        case 'w':
        case 'W':
            keys.w.pressed = true
            lastKey = 'w'
            break
        case 'a':
        case 'A':
            keys.a.pressed = true
            lastKey = 'a'
            break
        case 's':
        case 'S':
            keys.s.pressed = true
            lastKey = 's'
            break
        case 'd':
        case 'D':
            keys.d.pressed = true
            lastKey = 'd'
            break
    }
})

window.addEventListener('keyup', (e) => {
    switch (e.key) {
        case 'w':
        case 'W':
            keys.w.pressed = false
```

```
            break
        case 'a':
        case 'A':
            keys.a.pressed = false
            break
        case 's':
        case 'S':
            keys.s.pressed = false
            break
        case 'd':
        case 'D':
            keys.d.pressed = false
            break
    }
})
```