```html
<!DOCTYPE html>
<html>

<head>
	<title>Mineblox Battle Royale</title>
	<style>
		body,
		html {
			width: 100%;
			height: 100%;
			margin: 0;
			padding: 0;
			overflow: hidden;
		}

		canvas {
			position: absolute;
		}
	</style>
</head>

<body>

	<canvas id="game" width='1250' height="750" style="border:1px solid #000000;
background-color:#333333"></canvas>



	<script src='utils.js'></script>
	<script src='vector.js'></script>
	<script src='obj.js'></script>
	<script src='item.js'></script>
	<script src='bullet.js'></script>
	<script src='gun.js'></script>
	<script src='player.js'></script>
	<script src='scene.js'></script>


	<script src='game.js'></script>
</body>

</html>
```

Utils.js:

```
// Canvas setup
const canvas = document.getElementById("game");
canvas.width = window.innerWidth;
canvas.height = window.innerHeight;
const ctx = canvas.getContext("2d");
ctx.fillStyle = "#FF0000";
ctx.font = "50px Arial";
ctx.textAlign = 'center';



var iteration = 0;
var lost = false;
var loser;
function drawCircle(x, y, r, color) {
	ctx.beginPath();
	ctx.arc(x, y, r, 0, Math.PI * 2);
	ctx.fillStyle = color;
	ctx.fill();
}

function distance(x, y, x1, y1) {
	const xDist = Math.abs(x1 - x);
	const yDist = Math.abs(y1 - y);
	return Math.sqrt(xDist * xDist + yDist * yDist);
}

function drawLine(x, y, x1, y1, width) {
	ctx.lineWidth = width;
	ctx.beginPath();
	ctx.moveTo(x, y);
	ctx.lineTo(x1, y1);
	ctx.stroke();
}

function constraint(input, lower, upper) {
	var result = input;
	if (input < lower) {
		result = lower;
	} else if (input > upper) {
		result = upper;
```

```
        }
        return result;
}

function rd(min, max) {
        return Math.floor(Math.random() * (max - min + 1) + min);
}

function renderImage(file, x, y, width, height) {
        base_image = new Image();
        base_image.src = file;
        ctx.imageSmoothingEnabled = false;
        ctx.drawImage(base_image, x, y, width, height);
}

function makeBase(obj) {
        renderImage(obj.file, obj.x, obj.y, obj.width, obj.height);
}
```

Vector.js

```
class Vector {
   constructor(xV, yV) {
      this.xV = xV;
      this.yV = yV;
      this.vec = [this.xV, this.yV];
      this.mg = Math.sqrt(this.xV * this.xV + this.yV * this.yV);
      this.unitVec = [this.xV / this.mg, this.yV / this.mg];
   }
}
```

Obj.js

```
class Obj {
        constructor(x, y, width, height, file) {

                this.x = x;
                this.y = y;
                this.width = width;
                this.height = height;
```

```
		this.file = file;

		this.lBound = this.x;
		this.rBound = this.x + this.width;
		this.uBound = this.y;
		this.dBound = this.y + this.height;

		this.centerX = this.lBound + this.width / 2;
		this.centerY = this.uBound + this.height / 2;
	}

	touched() {
		console.log('touched: ');
		console.log(this.constructor.name);
	}

	place(scene) {
		scene.addObj(this);
	}

	draw() {
		makeBase(this);
	}

	nextFrame() {
		this.draw();
	}
}

class Wall extends Obj {
	constructor(x, y) {
		super(x, y, 30, 30, './assets/brick.png');
		this.health = 3;
	}
	draw() {
		makeBase(this);
		ctx.font = "20px Arial";
		ctx.fillStyle = "#DDDDDD";
		ctx.fillText(this.health, this.x + 20, this.y + 20);
		ctx.fillStyle = "#FF0000";
```

```
            ctx.font = "50px Arial";
      }
}
```

Item.js

```
class Item extends Obj {
  constructor(name, x, y, file) {
    super(x, y, 20, 20, file);
    this.name = name;
    this.birthTime = iteration;
  }

  touched(scene, player) {
    if (this.name === 'HealthPack') {
      if (player.health < 5) {
        player.health += 1;
      }
      scene.objs.splice(scene.objs.indexOf(this), 1);
    }
  }


}
```

Bullet.js

```
class Bullet {
      constructor(x, y, player, range, speed) {
            this.x = x;
            this.y = y;
            this.initX = this.x;
            this.initY = this.y;
            this.width = 4;
            this.height = 4;
            this.color = '#FFFFFF';
            this.lBound = this.x;
            this.rBound = this.x + this.width;
            this.uBound = this.y;
            this.dBound = this.y + this.height;
```

```
		this.direction = player.direction;
		this.distanceTravelled = 0;
		this.maxDistance = range;
		this.moveMentSpeed = speed; //bigger num = faster
		//index of this bullet in the player's list
		this.ind = player.bullets.length;
		this.vec;
		if (this.direction === 'left') {
			this.vec = new Vector(-this.moveMentSpeed, 0);
		} else if (this.direction === 'right') {
			this.vec = new Vector(this.moveMentSpeed, 0);
		} else if (this.direction === 'up') {
			this.vec = new Vector(0, -this.moveMentSpeed);
		} else if (this.direction === 'down') {
			this.vec = new Vector(0, this.moveMentSpeed);
		}
	}
	setPos(player) {
		this.x += this.vec.xV;
		this.y += this.vec.yV;
		//if it has reachs the end
		const diff = this.maxDistance - distance(this.x, this.y, this.initX, this.initY);
		if (diff <= 5) {
			player.bullets.splice(player.bullets.indexOf(this), 1);
		}
	}

	playerCollision(player) {
		var xInRange = player.enemy.lBound < this.x && this.x < player.enemy.rBound;
		var yInRange = player.enemy.uBound < this.y && this.y < player.enemy.dBound;
		if (xInRange && yInRange) {
			player.enemy.health -= 1;
			player.bullets.splice(player.bullets.indexOf(this), 1);
		}
	}
	wallCollision(scene, player) {
		for (var i = 0; i < scene.objs.length; i++) {
			const wall = scene.objs[i];
			var xInRange = wall.lBound < this.x && this.x < wall.rBound;
			var yInRange = wall.uBound < this.y && this.y < wall.dBound;
			if (xInRange && yInRange && scene.objs[i].constructor.name === 'Wall') {
				scene.objs[i].health -= 1;
				player.bullets.splice(player.bullets.indexOf(this), 1);
```

```
                }

            }
        }
        draw() {
            drawCircle(this.x, this.y, this.width, this.color);
        }
        nextFrame() {
            this.draw();
        }
}
```

Gun.js

```
class Gun {
        constructor(name, reloadSpeed, range, bulletSpeed, bulletNum, length, width) {
            this.name = name;
            this.reloadSpeed = reloadSpeed;
            this.range = range;
            this.bulletSpeed = bulletSpeed;
            this.bulletNum = bulletNum;
            this.length = length;
            this.width = width;
        }
}

const Pistol = new Gun('pistol', 20, 400, 3, 1, 13, 8);
Pistol.shoot = function (player) {
        player.bullets.push(new Bullet(player.x, player.y,
            player, this.range, this.bulletSpeed));
}
const Shotgun = new Gun('shotgun', 100, 250, 3, 4, 17, 11);
Shotgun.shoot = function (player) {
        const rs = 30;
        for (var load = 0; load < this.bulletNum; load++) {
            var bullet = new Bullet(player.x,
                player.y, player,
                this.range, this.bulletSpeed);

            switch (player.direction) {
                case 'left':
                case 'right':
                    //so that the shots spread out
                    bullet.vec.yV += (load - 1.5) / 2;
                    break;
                case 'up':
                case 'down':
                    // so the shots spread out
                    bullet.vec.xV += (load - 1.5) / 2;
                    break;
            }
            player.bullets.push(bullet);
        }
}
const Sniper = new Gun('sniper', 70, canvas.width - 300, 20, 2, 23, 7);
var rs = 0;
Sniper.shoot = function (player) {
        for (var load = 0; load < this.bulletNum; load++) {
            player.bullets.push(new Bullet(player.x + rd(-rs, rs),
                player.y + rd(-rs, rs), player,
                this.range, this.bulletSpeed));
        }
}

const Rocket = new Gun('rocket', 350, 650, 0.7, 14, 15, 15);
rs = 2;
Rocket.shoot = Sniper.shoot;
```

Player.js

```
class Player {
        constructor(name, x, y, color) {
            this.name = name;
            this.x = x;
            this.y = y;
            this.radius = 10;
            this.lBound = this.x - this.radius;
            this.rBound = this.x + this.radius;
            this.uBound = this.y - this.radius;
            this.dBound = this.y + this.radius;
            this.color = color;
            this.direction = 'left'; //by default
            this.movementScale = 3;
            this.bullets = [];
            //movement flags to allow smooth movement
            this.mvL = false;
            this.mvR = false;
```

```
                this.mvU = false;
                this.mvD = false;
                //so you won't be killed by ur own bullets LOL
                this.enemy;
                //so the game can actually end
                this.gun = Pistol; //by default
                this.health = 5;
                this.canShoot = true;

                this.numWalls = 5;
        }


        checkTouch(obj) {
                var xRange = (obj.lBound < this.x + this.radius) && (this.x - this.radius <
obj.rBound);
                var yRange = (obj.uBound < this.y + this.radius) && (this.y - this.radius <
obj.dBound);
                return xRange && yRange;
        }

        objCollision(scene, prevX, prevY, objName) {
                //makes sure the player doesn't go into an object
                for (var i = 0; i < scene.objs.length; i++) {
                        //only wall collision, doesn't include other objs
                        if (this.checkTouch(scene.objs[i])) {
                                if (scene.objs[i].constructor.name === 'Wall') {
                                        this.x = prevX;
                                        this.y = prevY;
                                } else if (scene.objs[i].constructor.name === 'Item') {
                                        scene.objs[i].touched(scene, this);
                                }
                        }
                }
        }

        move(scene) {
                const objName = 'Wall';
                if (this.mvL) {
                        var prevX = this.x;
                        this.x -= this.movementScale;
                        this.objCollision(scene, prevX, this.y, objName);
                        this.direction = 'left';
                }
                if (this.mvR) {
                        var prevX = this.x;
                        this.x += this.movementScale;
                        this.objCollision(scene, prevX, this.y, objName);
                        this.direction = 'right';
                }
                if (this.mvU) {
                        var prevY = this.y;
                        this.y -= this.movementScale;
                        this.objCollision(scene, this.x, prevY, objName);
                        this.direction = 'up';
                }
                if (this.mvD) {
                        var prevY = this.y;
                        this.y += this.movementScale;
                        this.objCollision(scene, this.x, prevY, objName);
                        this.direction = 'down'
                }
                //update left and right bounds everytime it moves
                this.lBound = this.x - this.radius;
                this.rBound = this.x + this.radius;
                this.uBound = this.y - this.radius;
                this.dBound = this.y + this.radius;
                //make sure it doesn't go off the canvas
                this.x = constraint(this.x, 0, canvas.width);
                this.y = constraint(this.y, 0, canvas.height);
        }
        changeGun() {
                switch (this.gun) {
                        case Pistol:
                                this.gun = Shotgun;
                                break;
                        case Shotgun:
                                this.gun = Sniper;
                                break;
                        case Sniper:
                                this.gun = Rocket;
                                break;
                        case Rocket:
                                this.gun = Pistol;
                                break;
                }
```

```
	}

	shoot() {
		if (this.canShoot) {
			this.gun.shoot(this);
			this.canShoot = false;
		}
	}

	placeWall(scene) {
		var newWall;
		if (this.direction === 'left') {
			newWall = new Wall(this.x - 50, this.y - 15);
		} else if (this.direction === 'right') {
			newWall = new Wall(this.x + 20, this.y - 15);
		} else if (this.direction === 'up') {
			newWall = new Wall(this.x - 20, this.y - 15 - 35);
		} else if (this.direction === 'down') {
			newWall = new Wall(this.x - 20, this.y - 15 + 35);
		}
		if (this.numWalls > 0) {
			scene.objs.push(newWall);
			this.numWalls -= 1;
		}

	}

	draw() {
		ctx.font = "15px Arial";
		ctx.fillStyle = "#000000";
		drawCircle(this.x, this.y, this.radius, this.color);
		ctx.fillText(this.gun.name, this.x, this.y - 26);
		ctx.fillText('hp: ' + this.health, this.x, this.y - 13);
		var ex;
		var why;
		if (this.name === 'player1') {
			ex = 50;
			why = 50;
		} else if (this.name === 'player2') {
			ex = canvas.width - 200;
			why = 50;
		}
		ctx.font = "20px Arial";
		ctx.fillText(this.name, ex, why - 20);
		ctx.fillText('hp:' + this.health, ex, why);
		ctx.fillText('walls:' + this.numWalls, ex, why + 20);
		ctx.fillText('gun:' + this.gun.name, ex, why + 40);

		//draw guns, it's alot of code, but worth it...
		//also so players could tell which directions they are facing in
		var ex = this.x;
		var why = this.y;
		var ex1 = this.x;
		var why1 = this.y;
		const length = this.gun.length;
		if (this.direction === 'left') {
			ex = this.x - 10;
			ex1 = ex - length;
		} else if (this.direction === 'right') {
			ex = this.x + 10;
			ex1 = ex + length;
		} else if (this.direction === 'up') {
			why = this.y - 10;
			why1 = why - length;
		} else if (this.direction === 'down') {
			why = this.y + 10;
			why1 = why + length;
		}
		ctx.strokeStyle = "#C0C0C0";
		drawLine(ex, why, ex1, why1, this.gun.width);

		//reset it to normal
		ctx.fillStyle = "#FF0000";
		ctx.font = "50px Arial";


	}

	shootEnemy() {
		for (var i = 0; i < this.bullets.length; i++) {
			this.bullets[i].playerCollision(this, this.enemy);
		}
	}

	checkHealth() {
		//so the game doesn't go on forever LOL
		if (this.health <= 0) {
			lost = true;
			loser = this;

		}

	}
	bulletWallCollision(scene) {
		for (var i = 0; i < this.bullets.length; i++) {
			this.bullets[i].wallCollision(scene, this);
		}
	}

	nextFrame(scene) {
		this.move(scene);
		this.draw();

		for (var i = 0; i < this.bullets.length; i++) {
			this.bullets[i].setPos(this);

			//could have gotten popped
			if (this.bullets.length !== 0) {
				try {
					this.bullets[i].draw();
				} catch (err) {
					console.log(this.bullets);
				}
			}
		}
		if (this.name === 'player1') {
			this.enemy = player2;
		} else if (this.name === 'player2') {
			this.enemy = player1;
		}
		this.shootEnemy();

		this.checkHealth();
		this.bulletWallCollision(scene);


		if (iteration % this.gun.reloadSpeed === 0) {
			this.canShoot = true;
		}
		if (iteration % 100 === 0 && this.numWalls < 10) {
			this.numWalls += 1
		}

		// if (iteration % 1300 === 0 && this.health < 5) {
		//	this.health += 1;
		// }
	}



}

var player1 = new Player('player1', 50, canvas.height / 2, '#66FF66');
var player2 = new Player('player2', canvas.width - 50, canvas.height / 2, '#FF7633');

player1.direction = 'right';


player1.enemy = player2;
player2.enemy = player1;
```

Scene.js

```
class Scene {
	constructor(objs) {
		this.objs = objs;
	}
	addObj(obj) {
		this.objs.push(obj);

	}
	draw() {
		renderImage('./assets/bg.jpg', 0, 0, canvas.width, canvas.height);

		for (var i = 0; i < this.objs.length; i ++) {
			this.objs[i].draw();
		}
	}
	spawnItems() {
		if (iteration % 500 === 0) {
			this.objs.push(new Item('HealthPack', rd(0, canvas.width), rd(0,
canvas.height), './assets/health.jpg'));
		}
	}
	rmItems() {
		for (var i = 0; i < this.objs.length; i ++) {
			const obj = this.objs[i];
			if (obj.constructor.name === 'Item' && (iteration - obj.birthTime > 1000)) {
				this.objs.splice(i, 1);
			}
		}
	}

	showDescription() {
		ctx.font = "20px Arial";
		ctx.fillStyle = '#FFFFFF';
		ctx.fillText('player1: wasd to move, f to shoot, g to build, q to change gun',
canvas.width / 2, 50);
		ctx.fillText("player2: arrow keys to move, / to shoot, '.' to build, ',' to change gun",
canvas.width / 2, 70);
	}
	nextFrame() {
		for (var i = 0; i < this.objs.length; i ++) {
			var obj = this.objs[i];
			if (obj.health <= 0) {
				this.objs.splice(this.objs.indexOf(obj), 1);
			}
		}
		this.spawnItems();
		this.rmItems();
		this.draw();
		this.showDescription();
	}
}

var scene1 = new Scene([]);
```

Game.js

```
var timer = setInterval(nextFrame, 17);

function nextFrame() {
	ctx.fillStyle = "#FF0000";
	ctx.font = "50px Arial";
	ctx.textAlign = 'center';
```

```
        ctx.clearRect(0, 0, canvas.width, canvas.height);
        scene1.nextFrame();
        player1.nextFrame(scene1);


        player2.nextFrame(scene1);



        if (lost) {
                ctx.clearRect(0, 0, canvas.width, canvas.height);
                renderImage('./assets/bg.jpg', 0, 0, canvas.width, canvas.height);
                ctx.fillStyle = loser.color;
                ctx.fillText(loser.name + ' has lost!', canvas.width / 2, canvas.height / 2);
                ctx.font = '30px Arial';
                ctx.fillText('game restarting...' , canvas.width / 2, canvas.height / 2 + 50);
                setTimeout(() => {
                        location.reload();
                }, 1500);
                clearInterval(timer);
        }
        iteration += 1;
}



document.addEventListener('keydown', function(event) {

        const x = event.keyCode;
        //player1 movements:
        if (x === 65) {
                player1.mvL = true;
        }else if (x === 68) {
                player1.mvR = true;
        }else if (x === 87) {
                player1.mvU = true
        }else if (x === 83) {
                player1.mvD = true;
        }else if (x === 70) {
                player1.shoot();
        }else if (x === 71) {
                player1.placeWall(scene1);
        }else if (x === 81) {
                player1.changeGun();
        }
        //player2 movements:
        else if (x === 37) {
                player2.mvL = true;
        }else if (x === 39) {
                player2.mvR = true;
        }else if (x === 38) {
                player2.mvU = true;

        }else if (x === 40) {
                player2.mvD = true;
        }else if (x === 191) {
                player2.shoot();
        }else if (x === 190) {
                player2.placeWall(scene1);
        }else if (x === 188) {
                player2.changeGun();
        }


        // else if (x === 80) {
        //      var result = '';
        //      for (var i = 0; i < wallsToAdd.length; i ++) {
        //              result += wallsToAdd[i];
        //      }
        //      console.log(result);
        // }
        // else if (x === 8) {
        //      scene1.objs.pop();
        //      wallsToAdd.pop();
        // }
});

document.addEventListener('keyup', function(event) {

        const x = event.keyCode;

        //player1 movements:
        if (x === 65) {
                player1.mvL = false;
        }else if (x === 68) {
                player1.mvR = false;
```

```
	}else if (x === 87) {
		player1.mvU = false
	}else if (x === 83) {
		player1.mvD = false;
	}

	//player2 movements:
	else if (x === 37) {
		player2.mvL = false;
	}else if (x === 39) {
		player2.mvR = false;
	}else if (x === 38) {
		player2.mvU = false;

	}else if (x === 40) {
		player2.mvD = false;
	}
});




// var wallsToAdd = [];

function addWall(event) {
	scene1.addObj(new Wall(event.clientX - 20, event.clientY - 20));
	// wallsToAdd.push(`scene1.addObj(new Wall(${event.clientX - 20}, ${event.clientY -
20}));\n`);
}
document.addEventListener("click", addWall);
```

Run.sh

```
google-chrome index.html
```

Server.js

```
//load modules
const http = require('http');
const fs = require('fs');
const path = require('path');
const url = require('url');
var express = require('express');
```

```
//Starts express
var app = express();
app.use(express.static('public'))
//GET html index file
var fileArr = [
	'index.html',
	'utils.js',
	'vector.js',
	'obj.js',
	'item.js',
	'bullet.js',
	'gun.js',
	'player.js',
	'scene.js',
	'game.js',
	'assets/bg.jpg',
	'assets/brick.png',
	'assets/health.jpg'

];

for (let i = 0; i < fileArr.length; i++) {
	app.get('/' + fileArr[i], (req, res) => {
		res.sendFile(__dirname + req.url);
		console.log("sent file: " + req.url);
	});
}

app.listen(1024, () => {
	console.log('App successfully started.');
})
```

```
README.md
# APCSP-project

controls:

player1:

'a':left

'd':right
```

'w':up

's':down

'q':change gun

'f':shoot

'g':place wall

player2:

leftArrow:left

rightArrow:right

upArrow:up

downArrow:down

'?':shoot

'>':placewall

'<':change gun

citation for images:
https://images.ecosia.org/uF6myYghIsYE8VF_8Y3p4-P2rHU=/0x390/smart/http%3A%2F%2Fstatic.planetminecraft.com%2Ffiles%2Fresource_media%2Fscreenshot%2F1228%2Farena4_2916525.jpg\

the image of the brick was made by a friend of mine

The healthpack image was made by me