

```

1 // weapons
2
3 class Weapon {
4     constructor(name, damage, className, image) {
5         this.name = name;
6         this.damage = damage;
7         this.className = className;
8         this.image = image;
9     }
10 }
11
12 var quarter = new Weapon("quarter note", 10, "weapon-1", "quarter.png");
13 var eighth = new Weapon("eighth note", 5, "weapon-2", "eighth.png");
14 var half = new Weapon("half note", 20, "weapon-3", "half.png");
15 var whole = new Weapon("whole note", 40, "weapon-4", "whole.png");
16 var keyblack = new Weapon("keyblack", 15, "block", "black.png");
17
18
19 // players
20 const player1 = {
21     position: {
22         // x: 0,
23         //y: 0
24     },
25     //health: 100,
26     currentWeapon: quarter,
27     isDefending: false
28 };
29
30 const player2 = {
31     position: {
32         //x: 0,
33         //y: 0
34     },
35     //health: 100,
36     currentWeapon: quarter,
37     isDefending: false
38 };
39
40 /*===== Build The Game
41 ===== */
42 // Generate random numbers
43 const generateRandomNum = () => Math.floor(Math.random() * 10);
44
45 // Generate grid with blocks
46 for (let i = 0; i < 10; i++) {
47     for (let j = 0; j < 10; j++) {
48         $('grid-container').append('<div class="grid-item" data-y=' + i + ' data-x=' + j +
49             '></div>');
50     }
51     $('#boardContainer').append('<div class="canvas"></div>');
52 }

```

```

51 }
52
53 // Iterates different elements to display them on the board
54 function generate(func, times) {
55     for (let i = 0; i < Number(times); i++) {
56         func();
57     }
58 }
59 // This functions helps blocks, weapons and players find an available aquare in the board
60 function placeElements(className) {
61     const random_x = generateRandomNum();
62     const random_y = generateRandomNum();
63     $('.grid-item').each(function () {
64         const element = $(this);
65         if (this.dataset['x'] == random_x && this.dataset['y'] == random_y) {
66             if (!(this.classList.contains("unavailable"))) {
67                 element.addClass(className);
68                 element.addClass("unavailable");
69                 // updates the position values to the player objects
70                 if (className === "player-1") {
71                     player1.position.x = this.dataset['x'];
72                     player1.position.y = this.dataset['y'];
73                 } else if (className === "player-2") {
74                     player2.position.x = this.dataset['x'];
75                     player2.position.y = this.dataset['y'];
76                 } else if (className === "weapon-1" ||
77                     className === "weapon-2" ||
78                     className === "weapon-3" ||
79                     className === "weapon-4") {
80                     element.addClass("weapon");
81                 }
82                 if (playerEncounter()) {
83                     console.log("Early Encounter");
84                     playerReset(className);
85                     placeElements(className);
86                 }
87             } else {
88                 // Function calls itself recursively to find an open space
89                 placeElements(className);
90             }
91         }
92     });
93 }
94
95
96 function generateGame() {
97     //reset();
98     // Anonymous functions
99     generate(function () {
100         placeElements("block");
101     }, 20);
102     generate(function () {

```

```

103     placeElements("weapon-1")
104 }, 1);
105 generate(function () {
106     placeElements("weapon-2");
107 }, 1);
108 generate(function () {
109     placeElements("weapon-3");
110 }, 1);
111 generate(function () {
112     placeElements("weapon-4");
113 }, 1);
114 generate(function () {
115     placeElements("keyblack");
116 }, 8);
117 generate(function () {
118     placeElements("player-1");
119 }, 1);
120 generate(function () {
121     placeElements("player-2");
122 }, 1);
123 movePlayer(player1);
124 movePlayer(player2);
125 pathHighlight();
126 weaponDisplay(player1);
127 weaponDisplay(player2);
128
129 }
130
131
132 function playerOneHighlight() {
133     document.getElementById('playerOneTurn').style.color = "green";
134     document.getElementById('playerTwoTurn').style.color = "white";
135 }
136
137 function playerTwoHighlight() {
138     document.getElementById('playerOneTurn').style.color = "white";
139     document.getElementById('playerTwoTurn').style.color = "green";
140
141 }
142
143
144 /*===== Player Movements
===== */
145 // Variable to check movements
146 let playerTurn = true;
147 //document.getElementById('playerOne').style.color = "green";
148
149 function pathHighlight() {
150     //if (!fightMode) {
151     if (playerTurn) {
152         possiblePath(player1);
153         $('.player-1').click(function () {

```

```

154     $('#possible').css("background-color", "rgba(0, 255, 0, 0.6)");
155 });
156
157
158     playerOneHighlight();
159 } else {
160     possiblePath(player2);
161     $('.player-2').click(function () {
162         $('#possible').css("background-color", "rgba(0, 255, 0, 0.6)");
163     });
164     playerTwoHighlight();
165 }
166
167 }
168
169 function movePlayer(player) {
170     //fightMode = false;
171
172     $('.grid-item').click(function () {
173         pathHighlight();
174         const element = $(this);
175         const block = this;
176         // Make sure is within distance
177         if (element.hasClass("possible")) {
178
179             if (player === player2) {
180                 playerTwoHighlight();
181
182                 $('.player-2').click(function () {
183                     $('#possible').css("background-color", "rgba(0, 255, 0, 0.6)");
184                 });
185
186                 document.getElementById("rest").style.display = "none";
187                 if (!playerTurn) {
188                     playerOneHighlight();
189
190                     weaponChecker(block, player);
191                     handleWeapon(element, player);
192                     playerReset("player-2");
193                     element.addClass("player-2");
194                     handleFight();
195                     playerTurn = !playerTurn;
196                 }
197             }
198
199             if (player === player1) {
200                 playerOneHighlight();
201                 $('.player-1').click(function () {
202                     $('#possible').css("background-color", "rgba(0, 255, 0, 0.6)");
203                 });
204             }
205

```

```

206         document.getElementById("rest2").style.display = "none";
207         if (playerTurn) {
208
209             weaponChecker(block, player);
210             handleWeapon(element, player);
211             playerReset("player-1");
212             element.addClass("player-1");
213             handleFight();
214             playerTurn = !playerTurn;
215
216         }
217     }
218
219 }
220
221 });
222 }
223
224 function getPlayerPosition() {
225     $('.grid-item').each(function () {
226         const element = $(this);
227         // I take the coordinates of the
228         if (element.hasClass("player-1")) {
229             player1.position.x = this.dataset['x'];
230             player1.position.y = this.dataset['y'];
231         }
232         if (element.hasClass("player-2")) {
233             player2.position.x = this.dataset['x'];
234             player2.position.y = this.dataset['y'];
235         }
236     });
237 }
238
239 /*===== Handle Weapons =====*/
240 function playerReset(player) {
241     $('.grid-item').each(function () {
242         const element = $(this);
243         element.removeClass(player);
244         element.removeClass("possible");
245         $('#aKey').attr("disabled", false);
246         $('#dKey').attr("disabled", false);
247         $('#left').attr("disabled", false);
248         $('#right').attr("disabled", false);
249     });
250 }
251 }
252
253 function squareOccupied(element) {
254     return (
255         element.hasClass("block") ||
256         element.hasClass("player-1") ||

```

```

257     element.hasClass("player-2")
258 );
259 }
260
261 function possiblePath(player) {
262     $('.grid-item').each(function () {
263         const element = $(this);
264         const block = this;
265         if (isInDistance(player, block) && !squareOccupied(element)) {
266             element.addClass("possible");
267         }
268     });
269     $('.grid-item').each(function () {
270         const element = $(this);
271         const block = this;
272         // value larger with larger X
273         if (isInDistance(player, block) && (block.dataset['x'] > player.position.x)) {
274
275
276             //removing inDistance allows for diagonal movement
277             //if((block.dataset['x'] > player.position.x)){
278             if (squareOccupied(element)) {
279                 occupiedObject = this;
280                 $('.possible').each(function () {
281                     const element = $(this);
282                     const block = this;
283
284                     if (block.dataset['x'] > occupiedObject.dataset['x']) {
285                         // console.log("block x is greater than player x");
286                         element.removeClass("possible");
287                     }
288                 });
289             }
290         }
291         // value with lower X
292         if (isInDistance(player, block) && (block.dataset['x'] < player.position.x)) {
293
294             if (squareOccupied(element)) {
295                 occupiedObject = this;
296                 $('.possible').each(function () {
297                     const element = $(this);
298                     const block = this;
299
300                     if (block.dataset['x'] < occupiedObject.dataset['x']) {
301                         //console.log("block x is less than player x");
302                         element.removeClass("possible");
303                     }
304                 });
305             }
306         }
307         // value with higher y
308         if (isInDistance(player, block) && (block.dataset['y'] > player.position.y)) {

```

```

309
310     if (squareOccupied(element)) {
311         occupiedObject = this;
312         $('.possible').each(function () {
313             const element = $(this);
314             const block = this;
315
316             if (block.dataset['y'] > occupiedObject.dataset['y']) {
317                 //console.log("block y is greater than player y");
318                 element.removeClass("possible");
319             }
320         });
321     }
322 }
323 // value with lower y
324 if (isInDistance(player, block) && (block.dataset['y'] < player.position.y)) {
325
326     if (squareOccupied(element)) {
327         occupiedObject = this;
328         $('.possible').each(function () {
329             const element = $(this);
330             const block = this;
331
332             if (block.dataset['y'] < occupiedObject.dataset['y']) {
333                 // console.log("block y is less than player y");
334                 //alert("up");
335                 element.removeClass("possible");
336             }
337         })
338     }
339 }
340 })
341
342
343 $('.grid-item').each(function () {
344     $('.possible').css("background-color", "");
345     //document.getElementById('playerTwo').style.color = "white";
346 });
347
348
349 }
350 // I need to reference all the elements that are "inDistance"
351 // To determine if they have block or player class, if they do
352 // movement is not possible
353 function isInDistance(player, block) {
354     const firstCondition = (Math.abs(block.dataset['x'] - player.position.x) < 4) &&
355         (block.dataset['y'] === player.position.y);
356     const secondCondition = (Math.abs(block.dataset['y'] - player.position.y) < 4) &&
357         (block.dataset['x'] === player.position.x);
358     return (firstCondition || secondCondition);
359     // $('.grid-item').each(pathHighlight);
360 }

```

```

361
362 // Check if there are any weapons in the path
363 function weaponChecker(block, player) {
364     checkSmallerX(block, player);
365     checkSmallerY(block, player);
366     checkLargerX(block, player);
367     checkLargerY(block, player);
368 }
369
370 // Check if there is a weapon left of the player
371 function checkSmallerX(block, player) {
372     if (block.dataset['x'] < player.position.x) {
373         $('.possible').each(function () {
374             const element = $(this);
375             const innerBlock = this;
376             if ((innerBlock.dataset['x'] < player.position.x) &&
377                 (innerBlock.dataset['y'] == player.position.y) &&
378                 innerBlock.dataset['x'] > block.dataset['x']) {
379                 if (element.hasClass("weapon") ||
380
381                     element.hasClass("keyblack")) {
382                     weaponChange(element, player);
383                 }
384             }
385         })
386     }
387 }
388
389 // Check if there is a weapon right of the player
390 function checkLargerX(block, player) {
391     if (block.dataset['x'] > player.position.x) {
392         $('.possible').each(function () {
393             const element = $(this);
394             const innerBlock = this;
395             if ((innerBlock.dataset['x'] > player.position.x) &&
396                 (innerBlock.dataset['y'] == player.position.y) &&
397                 (innerBlock.dataset['x'] < block.dataset['x'])) {
398                 if (element.hasClass("weapon") ||
399
400                     element.hasClass("keyblack")) {
401                     weaponChange(element, player);
402                 }
403             }
404         })
405     }
406 }
407
408 // Check if there is a weapon under the player
409 function checkSmallerY(block, player) {
410     if (block.dataset['y'] < player.position.y) {
411         $('.possible').each(function () {
412             const element = $(this);

```



```

413     const innerBlock = this;
414     if ((innerBlock.dataset['y'] < player.position.y) &&
415         (innerBlock.dataset['x'] == player.position.x) &&
416         innerBlock.dataset['y'] > block.dataset['y']) {
417         if (element.hasClass("weapon") ||
418
419             element.hasClass("keyblack")) {
420             weaponChange(element, player);
421         }
422     }
423 })
424 }
425 }
426
427 // Check if there is a weapon over the player
428 function checkLargerY(block, player) {
429     if (block.dataset['y'] > player.position.y) {
430         $('.possible').each(function () {
431             const element = $(this);
432             const innerBlock = this;
433             if ((innerBlock.dataset['y'] > player.position.y) &&
434                 (innerBlock.dataset['x'] == player.position.x) &&
435                 innerBlock.dataset['y'] < block.dataset['y']) {
436                 if (element.hasClass("weapon") ||
437
438                     element.hasClass("keyblack")) {
439                     weaponChange(element, player);
440                 }
441             }
442         })
443     }
444 }
445
446 var eighthNote = document.getElementById("eighth");
447 var quarterNote = document.getElementById("quarter");
448 var halfNote = document.getElementById("half");
449 var wholeNote = document.getElementById("whole");
450 var eNote = document.getElementById("enote");
451 var qNote = document.getElementById("qnote");
452 var hNote = document.getElementById("hnote");
453 var wNote = document.getElementById("wnote");
454
455
456 // If there is a weapon, handle it by removing the class and replacing it
457 // with the current one of the player
458 function weaponChange(element, player) {
459     let playerWeapon = player.currentWeapon;
460     let damage = this.damage;
461     if (element.hasClass("weapon-1")) {
462         element.removeClass("weapon-1");
463         element.addClass(playerWeapon.className);
464         player.currentWeapon = quarter;

```

```

465     weaponDisplay(player);
466     quarter.damage = 10;
467
468     } else if (element.hasClass("weapon-2")) {
469         element.removeClass("weapon-2");
470         element.addClass(playerWeapon.className);
471         player.currentWeapon = eighth;
472         weaponDisplay(player);
473         eighth.damage = 5;
474
475     } else if (element.hasClass("weapon-3")) {
476         element.removeClass("weapon-3");
477         element.addClass(playerWeapon.className);
478         player.currentWeapon = half;
479         weaponDisplay(player);
480         half.damage = 20;
481
482     } else if (element.hasClass("weapon-4")) {
483         element.removeClass("weapon-4");
484         element.addClass(playerWeapon.className);
485         player.currentWeapon = whole;
486         weaponDisplay(player);
487         whole.damage = 40;
488
489     } else if (element.hasClass("keyblack")) {
490         //keyblack.damage = 15;
491         if (player === player1) {
492             document.getElementById("rest").style.display = "block"
493             document.getElementById("myProgress").value -= 15;
494             progressMeter();
495             meterHealth();
496         } else if (player === player2) {
497             document.getElementById("rest2").style.display = "block"
498             document.getElementById("myPro").value -= 15;
499             proMeter();
500             meterHealth();
501         }
502         element.removeClass("keyblack");
503
504     }
505
506 }
507
508
509 function weaponDisplay(player) {
510     if (player === player1) {
511
512         if (player.currentWeapon === eighth) {
513             eighthNote.style.display = "block";
514         } else {
515             eighthNote.style.display = "none";
516         }

```

```

517
518     if (player.currentWeapon == quarter) {
519         quarterNote.style.display = "block";
520     } else {
521         quarterNote.style.display = "none";
522     }
523     if (player.currentWeapon == half) {
524         halfNote.style.display = "block";
525     } else {
526         halfNote.style.display = "none";
527     }
528     if (player.currentWeapon == whole) {
529         wholeNote.style.display = "block";
530     } else {
531         wholeNote.style.display = "none";
532     }
533
534
535 }
536 if (player === player2) {
537     if (player.currentWeapon == eighth) {
538         eNote.style.display = "block";
539     } else {
540         eNote.style.display = "none";
541     }
542
543     if (player.currentWeapon == quarter) {
544         qNote.style.display = "block";
545     } else {
546         qNote.style.display = "none";
547     }
548     if (player.currentWeapon == half) {
549         hNote.style.display = "block";
550     } else {
551         hNote.style.display = "none";
552     }
553     if (player.currentWeapon == whole) {
554         wNote.style.display = "block";
555     } else {
556         wNote.style.display = "none";
557     }
558
559 }
560 }
561
562 /*===== Player Encounters and fighting
===== */
563 //let fightMode = false;
564
565 function handleWeapon(element, player) {
566     weaponChange(element, player);
567 }

```

```

568 // What happens when players encounter each other
569 function playerEncounter() {
570     getPlayerPosition();
571     const xPosition = Math.abs(Number(player1.position.x) - Number(player2.position.x));
572     const yPosition = Math.abs(Number(player2.position.y) - Number(player1.position.y));
573     return (((xPosition == 0) && (yPosition == 1)) ||
574         ((yPosition == 0) && (xPosition == 1))
575     )
576 }
577 }
578
579 // Logic to take care of the turns
580 function handleFight() {
581     if (playerEncounter()) {
582         //fightMode = true;
583         //movePlayer(player1) = false;
584         //movePlayer(player2) = false;
585
586         $("#toggleButton").css("color", "green");
587         $("#WASDmodal").css("display", "block");
588         $("#arrowKeysmodal").css("display", "block");
589         $("#block").css("display", "none");
590         $("#block2").css("display", "none");
591
592         const oneAttacks = document.getElementById('aKey');
593         const twoAttacks = document.getElementById('left');
594         const oneDefends = document.getElementById('dKey');
595         const twoDefends = document.getElementById('right');
596
597         if (!playerTurn) {
598             //}
599             $('#aKey').attr("disabled", false).css("backgroundColor", "rgba(255, 19, 24, 0.55)");
600             $('#dKey').attr("disabled", false).css("backgroundColor", "rgba(255, 19, 24, 0.55)");
601             $('#left').css("color", "");
602             $('#right').css("color", "");
603             //$('#left').attr("disabled", true);
604             //$('#right').attr("disabled", true);
605         }
606         oneAttacks.onclick = function () {
607             playerReset(player1);
608             //$('#aKey').on('click', function(){
609             meterHealth();
610             proMeter();
611             proBar.value -= player1.currentWeapon.damage;
612             $('#aKey').attr("disabled", true).css("backgroundColor", "");
613             $('#dKey').attr("disabled", true).css("backgroundColor", "");
614
615             playerTwoHighlight();

```

```

618     $('#left').css("backgroundColor", "rgba(0, 118, 255, 0.57)");
619     $('#right').css("backgroundColor", "rgba(0, 118, 255, 0.57)");
620 }
621
622 //$('#aKey').attr("disabled", false);
623 oneDefends.onclick = function () {
624     playerReset(player1);
625     //$('#dKey').on('click', function(){
626     //defendFunc();
627     meterHealth();
628     //player1.isDefending = true;
629     progressMeter();
630
631     backBar.value -= player2.currentWeapon.damage / 2;
632     $('#dKey').attr("disabled", true).css("backgroundColor", "");
633     $('#aKey').attr("disabled", true).css("backgroundColor", "");
634
635     playerTwoHighlight();
636     $('#left').css("backgroundColor", "rgba(0, 118, 255, 0.57)");
637     $('#right').css("backgroundColor", "rgba(0, 118, 255, 0.57)");
638     //}
639 }
640 //$('#dKey').attr("disabled", false);
641 if (playerTurn) {
642     //if(movePlayer(player2)){
643     //fightMode = false;
644
645     //}
646     $('#left').attr("disabled", false).css("backgroundColor", "rgba(0, 118, 255, 0.57)"
647 );
648     $('#right').attr("disabled", false).css("backgroundColor", "rgba(0, 118, 255,
649 0.57)");
650     $('#aKey').css("color", "");
651     $('#dKey').css("color", "");
652     //$('#aKey').attr("disabled", true);
653     //$('#dKey').attr("disabled", true);
654 }
655 twoAttacks.onclick = function () {
656     playerReset(player2);
657     //$('#left').on('click', function(){
658     meterHealth();
659     progressMeter();
660     backBar.value -= player2.currentWeapon.damage;
661     $('#left').attr("disabled", true).css("backgroundColor", "");
662     $('#right').attr("disabled", true).css("backgroundColor", "");
663
664     playerOneHighlight();
665     $('#aKey').css("backgroundColor", "rgba(255, 19, 24, 0.55)");
666     $('#dKey').css("backgroundColor", "rgba(255, 19, 24, 0.55)");
667 }
668 //$('#left').attr("disabled", false);
669 twoDefends.onclick = function () {

```

```

668     playerReset(player2);
669     //$('#right').on('click', function(){
670     //defendFunc();
671     meterHealth();
672     //player2.isDefending = true;
673     proMeter();
674     proBar.value -= player1.currentWeapon.damage / 2;
675     $('#right').attr("disabled", true).css("backgroundColor", "");
676     $('#left').attr("disabled", true).css("backgroundColor", "");
677
678     playerOneHighlight();
679     $('#aKey').css("backgroundColor", "rgba(255, 19, 24, 0.55)");
680     $('#dKey').css("backgroundColor", "rgba(255, 19, 24, 0.55)");
681 }
682 //getPlayerPosition();
683
684
685     //$('#right').attr("disabled", false);
686 }
687 /**/
688 //playerReset(player1);
689 //playerReset(player2);
690 else if (!playerEncounter()) {
691     $("#toggleButton").css("color", "");
692     $("#WASDmodal").css("display", "none");
693     $("#arrowKeysmodal").css("display", "none");
694
695     $("#block").css("display", "block");
696     $("#block2").css("display", "block");
697     //fightMode = false;
698
699     $('#right').attr("disabled", true);
700     $('#left').attr("disabled", true);
701     $('#dKey').attr("disabled", true);
702     $('#aKey').attr("disabled", true);
703 }
704
705 }
706
707
708 // Logic to take care of the attacks
709 function meterHealth() {
710
711     handleFight();
712
713     if (backBar.value <= 0) {
714         backBar.value = 0;
715
716         window.open('beethoven.html', "_self");
717         //beetAudio();
718
719

```

```
720     }
721     //if (player2.health <= 0 ) {
722     if (proBar.value <= 0) {
723         proBar.value = 0;
724
725         window.open('mozart.html', "_self");
726         //ma.play();
727     }
728 }
```