



Le jeu du pirate

RÉALISÉ PAR MATHIAS BRADICEANU

Sommaire



Étape 1 : génération de la carte

- Générer aléatoirement la carte du jeu.
- Insérer un nombre limité d'armes, placées aléatoirement
- Insérer les deux joueurs de façon aléatoire sur la carte au chargement de la partie. Non côte à côte

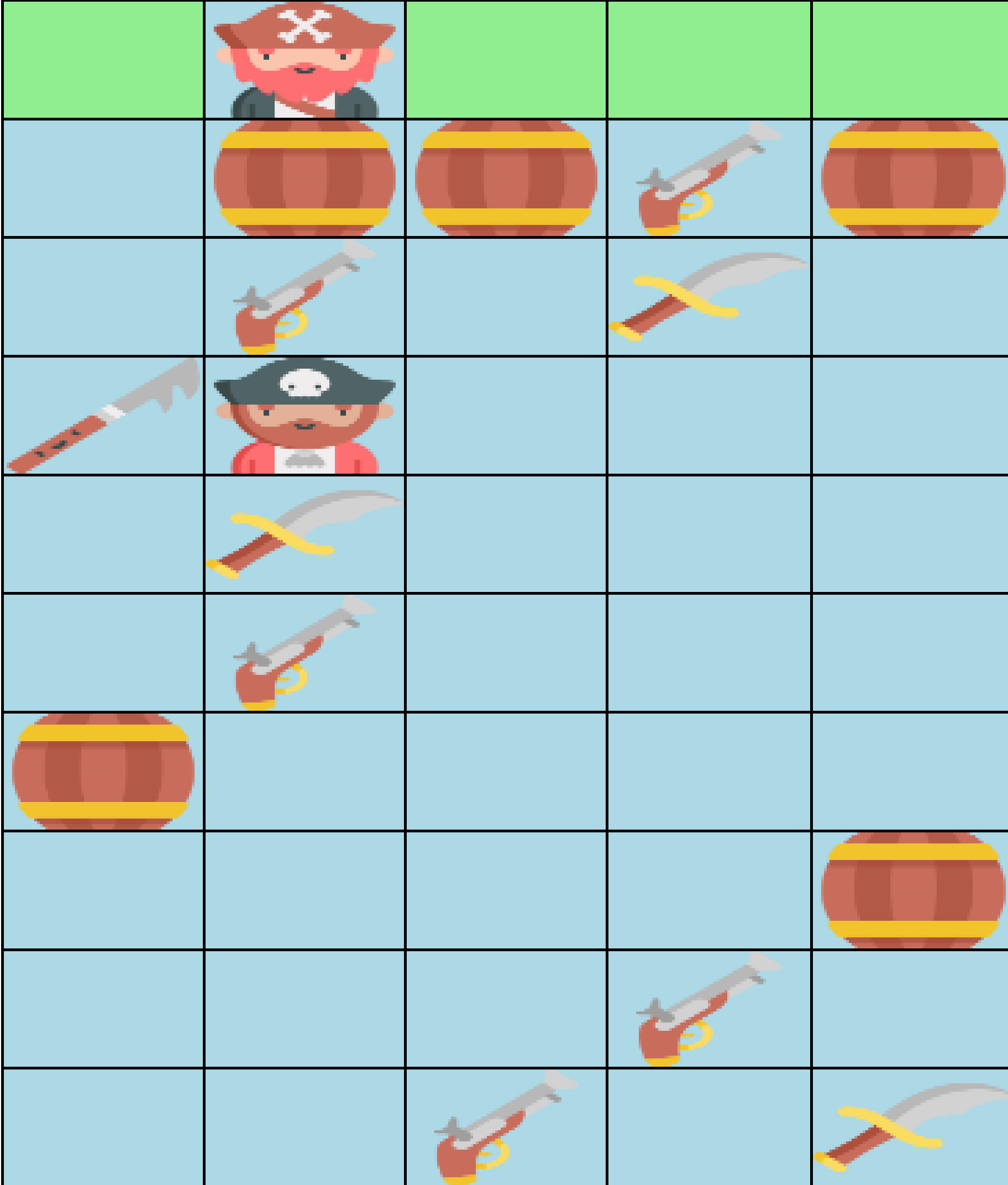
Étape 2 : les mouvements

- Déplacement possible de 3 cases horizontalement et verticalement puis changement de personnage.
- En cas de déplacement sur une case contenant une arme, le joueur laisse son arme actuelle sur place et la remplace par la nouvelle

Étape 3 : le combat !

- Le combat se déroule au tour par tour.
- Un joueur peut attaqué : il subit les dégâts de l'arme qu'il a actuellement
- Le joueur peut se défendre : il encaisse alors 50% de dégâts en moins qu'en temps normal.
- Fin de partie quand les points de vie d'un des deux joueurs tombent à 0.





CHOIX TECHNIQUE



SweetAlert



- GÉNÉRATION DE LA CARTE (Map.js)

-> génération de la structure du tableau

```
//create the lines
for(j = 0; j < this.nbOfLines; j++) {
    const trElt = document.createElement('tr');
    trElt.id = `line-${j}`;
    $('table').append(trElt);
}

//create the cells
for (i=0; i < numberOfBoxes; i++) {
    const tdElt = document.createElement('td');
    tdElt.id = `${x}-${y}`; //each td as a unique id → his x/y position
    // tdElt.innerHTML = i; //just for info
    $('#line-${indexOfTheLine}`).append(tdElt); //pushing into the tr element
    x++;

    //if there are 10 colmuns
    if ($('#line-${indexOfTheLine}').children().length === 10) {
        indexOfTheLine++; //go to the next line
        x = 0; //for the X position of the cell
        y++; //for the Y position of the cell
    }
}
```



<td id="x-y">

- GÉNÉRATION DE LA CARTE (Map.js)

-> génération des murs

```
generateWalls() {  
  const tdElts = $("td");  
  // the number of the walls will be in this interval  
  const min = 10;  
  const max = 15;  
  const randomNumber = random(min, max); // number of walls  
  
  for (let i = 0; i < randomNumber; i++) {  
    // selecting a random <td> element  
    let index = random(0, tdElts.length);  
    let randomTdElt = tdElts[index];  
  
    //while the <td> element is not free  
    while (this.getCellContent(randomTdElt.id) !== 0) {  
      //reassign a new <td>  
      index = random(0, tdElts.length);  
      randomTdElt = tdElts[index];  
    }  
  
    // managing the cell for the wall  
    $(randomTdElt).removeClass("free");  
    $(randomTdElt).addClass("greyed");  
  }  
}
```

index.js -> random()



- GÉNÉRATION DE LA CARTE (Map.js)
- > génération des armes



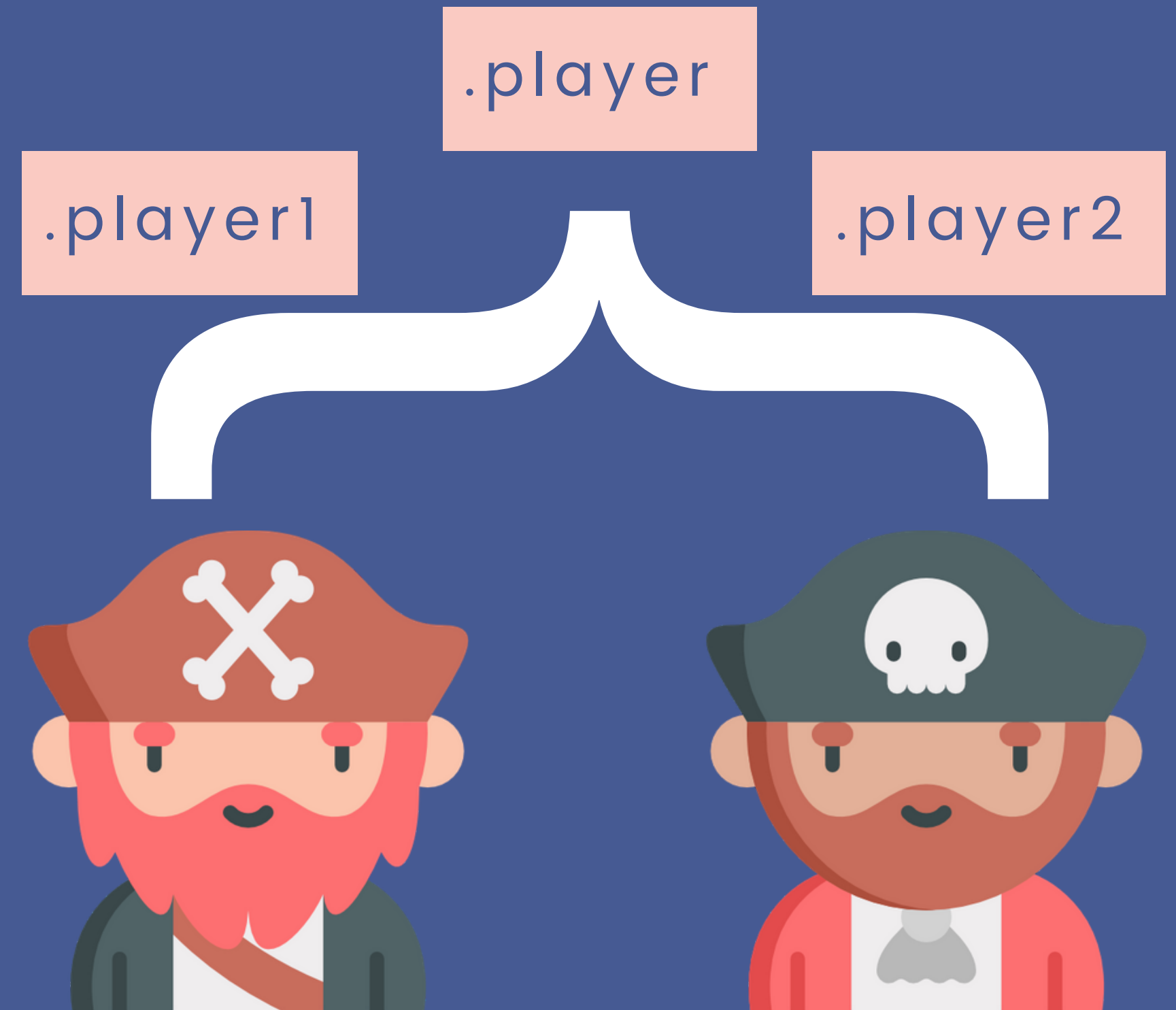
boucle qui évite les conflits :

```
while (this.getCellContent(elPos) !== 0) {  
  //picking a new pos  
  let newXpos = random(0, this.nbOfLines); // the maximum x(width) grid  
  let newYpos = random(0, this.nbOfColumns); // the maximum y(height) grid  
  
  //assignate the new pos to the weapon  
  randomWeapon.x = newXpos;  
  randomWeapon.y = newYpos;  
  elPos = newXpos + "-" + newYpos;  
  randomWeapon.pos = elPos;  
  
  //selecting a new cell to the weapon  
  tdEltId = `#${elPos}`;  
}
```

- GÉNÉRATION DE LA CARTE (Map.js)

-> génération des joueurs

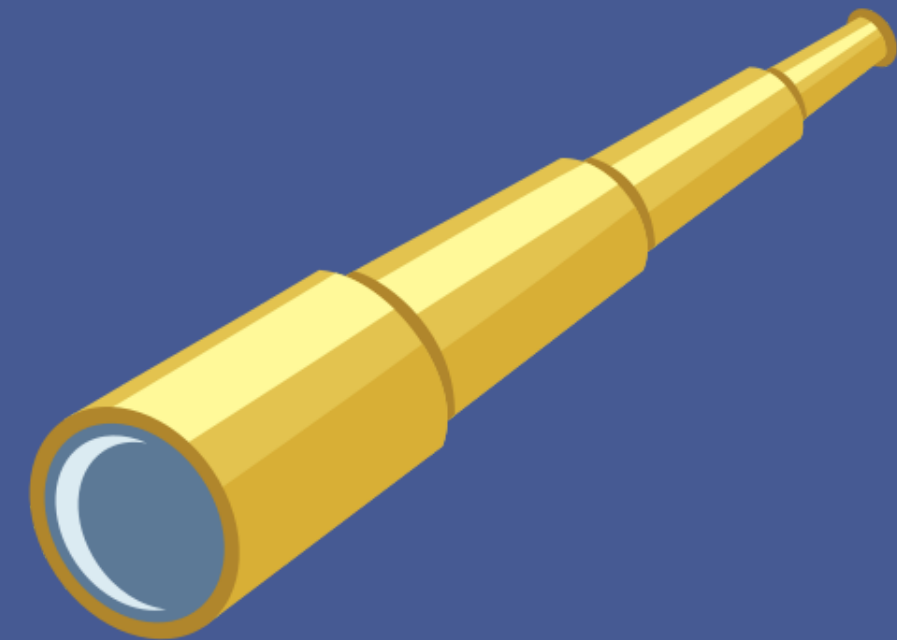
```
createPlayers() {  
  // get and select the position of each players  
  this.players.forEach(el => {  
    let elPos = `${el.x}-${el.y}`;  
    el.pos = elPos;  
    let tdEltId = `${'#${el.pos}'}`;  
  
    // while the pos is not free OR the pos is free but  
    while (  
      this.getCellContent(elPos) !== 0 ||  
      this.lookAround(elPos, 1) === 1  
    ) {  
      let newXpos = random(0, 10);  
      let newYpos = random(0, 10);  
      el.x = newXpos;  
      el.y = newYpos;  
      elPos = newXpos + "-" + newYpos;  
      el.pos = elPos;  
      tdEltId = `${'#${elPos}'}`;  
    }  
  
    // managing the cell for player  
    tdEltId.removeClass("free");  
    tdEltId.addClass("player");  
    tdEltId.attr("data-player", `player${el.number}`);  
    tdEltId.addClass(tdEltId.attr("data-player"));  
  });  
}
```





Étape 2 :

LES MOUVEMENTS



- LES MOUVEMENTS (Map.js & Game.js)

-> Déplacement possible de 3 cases horizontalement & verticalement

```
// displaying the possibilites
let posArrayKeys = Object.keys(posArray);
for (let i = 0; i < posArrayKeys.length; i++) {
  const actualKey = posArrayKeys[i];
  posArray[actualKey].forEach(el => {
    // if it's free
    if (this.getCellContent(el) !== 1 && this.getCellContent(el) !== 3) {
      $('#'+ el).addClass('green');
    }
    // if it's an obstacle
    else {
      // get index of the obstacle
      const obstacleIndex = posArray[actualKey].indexOf(el);
      //reduce the table of possibilites for this line, so we pass to the
      next direction. (=> avoiding to cross the obstacle)
      posArray[actualKey].splice(obstacleIndex, posArrayKeys.length - 1);
    }
  })
}
```

Afficher les
mouvements
possibles

Map.js =>
displayMoves()

Babord, Tribord,
Proue, Poupe,
nous vaquerons
!



- LES MOUVEMENTS (Player.js)

-> Déplacement possible de 3 cases
horizontalement & verticalement



Changer la position
des joueurs

Game.js =>
movePlayers()

Player.js =>
updatePlayerPosition

```
updatePlayerPosition(player, newPos) { //7-1
  let x = parseInt(newPos.charAt(0));
  let y = parseInt(newPos.charAt(newPos.length - 1));
  player.x = x;
  player.y = y;
}
```

LES MOUVEMENTS (Map.js)

-> Gérer le cas de ramassage d'arme

Le joueur ramasse une arme :

```
if ($(e.currentTarget).hasClass("weapon")) {  
    let oldWeapon = this.user.weaponName;  
  
    //if it's the first weapon  
    if (this.user.weaponName === "default") {  
        this.user.weaponName = $(e.currentTarget).attr("data-weapon");  
        $(e.currentTarget)  
            .removeClass($(e.currentTarget).attr("data-weapon"))  
            .removeClass("weapon")  
            .removeAttr("data-weapon");  
    } else {  
        this.user.weaponName = $(e.currentTarget).attr("data-weapon");  
        $(e.currentTarget)  
            .removeClass($(e.currentTarget).attr("data-weapon"))  
            .addClass(oldWeapon)  
            .attr("data-weapon", oldWeapon);  
    }  
}
```

Étape 3 :

LE COMBAT




- LE COMBAT (Game.js)
- > Le combat se déroule au tour par tour

this.tour++



Récupérer le player joueur actuel



```
constructor() {  
  this.map;  
  this.tour = 0;  
  this.user;  
  this.textToPrompt = "";  
}
```

```
getCurrentPlayer() {  
  if (this.tour % 2 === 0) {  
    //pair  
    return this.players[0];  
  } else {  
    //not pair  
    return this.players[1];  
  }  
  // exemple : 2%2 = 0 but 3%2 = 1, etc ...  
}
```



- LE COMBAT (Game.js)

-> Le joueur peut attaqué et inflige les dégâts de son arme



```
//looking if they are next to each others
if (this.map.lookAround(idOftheWantedPosition, 1) === 1) {
    this.fight();
};
```

movePlayers ->
Map.lookAround()



fight()

```
//Displaying an alert
swalWithBootstrapButtons.fire({
  title: "COMBAT !",
  text: `Combattez-vous jusqu'à la mort ! Joueur ${this.user.number}, c'est
à vous d'attaquer avec votre ${this.map.weapons[this.user.weaponName]
.nameFR}`,
  icon: 'warning',
  showCancelButton: true,
  confirmButtonText: "Je veux l'attaquer !",
  cancelButtonText: 'Je préfère me défendre ... ',
  reverseButtons: true,
  allowOutsideClick: false
});
```

SweetAlert

- LE COMBAT (Game.js)

-> Le joueur peut attaqué et inflige les dégâts de son arme

Le joueur veut attaquer :

Player.attack()

```
.then((result) => {  
  if (result.value) { //if he clicked on "Je veux l'attaquer"  
    //Checking if one of them is defending  
    this.user.attack();  
    //Display a new (confirmation) alert  
    swalWithBootstrapButtons.fire(  
      `Coup porté !`,  
      this.textToPrompt, //to display exactly the damages according to  
      the user isDefending attribute  
      `success`  
    ).then(() => { //on click on the button " OK "  
      //check if a player is dead or not  
      this.isOneOfThemDead();  
      //if no one is dead, we pass to the other player and fight again  
      this.tour += 1;  
      this.fight();  
    })  
  }  
})
```


- LE COMBAT (Game.js)

-> Le joueur peut se défendre

Le joueur veut se défendre



Player.defense()



```
else if (result.dismiss === Swal.DismissReason.cancel) {  
  //Displaying an alert  
  swalWithBootstrapButtons.fire(  
    'Défense activée !',  
    'Les dégats reçus sont réduits.',  
    'error'  
  ).then(() => { //on click on " OK "  
    //changing the state of the player  
    this.user.defense();  
    //pass to the next player and fight again  
    this.tour += 1;  
    this.fight();  
  })  
}
```

- LE COMBAT (Player.js)

-> La méthode attack()



```
if(this.number === 2) { //P2
  const player1 = this.game.players[0];
  if(player1.isDefending) {
    player1.life -= (this.damages / 2);
    player1.isDefending = !player1.isDefending;
    this.game.textToPrompt = `Votre attaque grâce à votre $
    {userWeaponNameInFrench} fait <b>${this.damages/2}</b> de dégâts sur
    l'adversaire`;
  } else {
    player1.life -= this.damages;
  }
} else { //P1
  const player2 = this.game.players[1];
  if(player2.isDefending) {
    player2.life -= (this.damages / 2);
    player2.isDefending = !player2.isDefending;
    this.game.textToPrompt = `Votre attaque grâce à votre $
    {userWeaponNameInFrench} fait <b>${this.damages/2}</b> de dégâts sur
    l'adversaire`;
  } else {
    player2.life -= this.damages;
  }
}
```

- LE COMBAT (Player.js)
- > La méthode defense()

```
//change the state of the player  
defense() {  
    this.isDefending = true;  
}
```



PROJET 6 OPENCLASSROOMS

CRÉER UN JEU DE PLATEAU TOUR PAR TOUR EN JAVASCRIPT



RÉALISÉ PAR MATHIAS BRADICEANU