1. Introducció

La pràctica consta de fer un joc amb l'ús de HTML, CSS i JavaScript.

És un joc de lluita on tu ets un personatge i has d'intentar mantenir-te amb vida fins al final del joc. El joc tracta de sobreviure respecte altres jugadors que intentaran matar-te.

Cada personatge és controlat per cada un dels jugadors que es troben situats dins d'un mapa tancat.

Cada jugador pot fer el següent dins del joc:

- Moure´s pel mapa de forma que pot tirar endavant, endarrere o moure la càmera cap als costats.
- Atacar als enemics.
- Tindre un visor de navegació en el que pot visualitzar el mapa sencer.
- Tindre un mini mapa que mostra la posició del jugador respecte els enemics i la direcció cap a la que es mouen. També podem trobar localitzats els objectes que hi ha pel el mapa.
- Fer ús d'una brúixola.
- Veure les seves estadístiques en un costat de la pantalla.
- Nou jugador, reviure jugador o eliminar jugador.

2. Desenvolupament i resultats

El funcionament del joc conté molts aspectes rellevants, un dels més importants son les apis que hem creat.

La primera api es la que ens proporciona el token i el code del jugador, es a dir, la que hem de cridar al crear el jugador **SpawnPlayer.**

```
//Llamada a la api para spawnear al jugador
function spawnPlayer(nombre) {
  let url = "http://battlearena.danielamo.info/api/spawn/" + group_token + "/" + nombre;
  fetch(url, {
    method: 'GET'
  })
    .then(function (response) {
      return response.json();
    })
    .then(function (json) {
      //guardamos la informacion del jugador
      player.token = json.token;
      player.code = json.code;
      player.nombre = nombre;
      console.log(json);
      //creamos el refresco
      interval = setInterval(update, 2000);
    })
    .then(function () {
      //buscamos la informacion del jugador que acabamos de crear
      getPlayer(player.token);
    })
    .catch((error) => {
      console.log(error);
    });
}
```

Tot seguit, executem **GetPlayer** per obtenir totes les característiques del jugador, i guardar-les a la classe player.

```
function getPlayer(token) {
  let url = "http://battlearena.danielamo.info/api/player/" + group_token + "/" + token;
  fetch(url, {
    method: 'GET'
  })
    .then(function (response) {
      return response.json();
    })
    .then(function (json) {
      //guardamos la informacion del jugador
      player.attack = json.attack;
      player.defense = json.defense;
      player.direction = json.direction;
      player.image = json.image;
      player.name = json.name;
      player.object = json.object;
      player.vitalpoints = json.vitalpoints;
      player.x = json.x;
      player.y = json.y;
      player.kills = 0;
      console.log(json);
      //actualizamos los datos del jugador
      document.getElementById('vitalpoints').innerHTML = player.vitalpoints;
      document.getElementById('attack').innerHTML = player.attack;
      document.getElementById('defense').innerHTML = player.defense;
      document.getElementById('kills').innerHTML = player.kills;
    })
    .then(function () {
      //si es la primera vez que la llamamos, guardamos en que posicion esta la brujula
      if (first) {
        setDirections();
        first = false;
      }
    })
    .catch((error) => {
      console.log(error);
    });
}
```

La api **getMap** ens retorna la posició de tots els enemics que hi ha al mapa.

```
//Llamada a la api para ver las posiciones de todos los enemigos en el mapa
function getMap(token) {
  let url = "http://battlearena.danielamo.info/api/map/b89f96ae/" + token;
  fetch(url, {
    method: 'GET'
  })
    .then(function (response) {
      return response.json();
    })
    .then(function (json) {
      console.log(json);
      //Eliminamos los enemigos que teniamos en el array
      for (let i = 0; i < enemies_map.length; i++) {
        enemies_map[i].emptyEnemy();
      }
      //Guardamos los nuevos enemigos.
      for (let i = 0; i < json.enemies.length; i++) {
        var new_enemy = new Enemy(json.enemies[i].direction, json.enemies[i].x, json.enemies[i].y);
        enemies_map.push(new_enemy);
      }
    })
    .then(function () {
      //Llamamos a la otra api para asegurarnos de que tenga la informacion al crear el mapa
      getPlayersAndObjects(player.token);
    })
    .catch((error) => {
      console.log(error);
    });
}
```

Després cridem **getPlayersAndObjects** es una api que cridem per a obtenir més informació sobre aquests enemics que estan a la mateixa o a caselles adjacents al nostre jugador.

```
//Llamada a la api para ver la informacion mas detallada de los enemigos alrededor del jugador
function getPlayersAndObjects(token) {
  let url = "http://battlearena.danielamo.info/api/playersobjects/" + group_token + "/" + token;
  fetch(url, {
    method: 'GET'
  })
    .then(function (response) {
      return response.json();
    })
    .then(function (json) {
      console.log(json);
      //Eliminamos los enemigos que teniamos en el array
      for (let i = 0; i < enemies_close.length; i++) {
        enemies_close[i].emptyEnemy();
      }
      //Guardamos los nuevos enemigos.
      for (let i = 0; i < json.enemies.length; i++) {
        enemies_close[i] = new Enemy;
        enemies_close[i].vitalpoints = json.enemies[i].vitalpoints;
        enemies_close[i].image = json.enemies[i].image;
        enemies_close[i].direction = json.enemies[i].direction;
        enemies_close[i].x = json.enemies[i].x;
        enemies_close[i].y = json.enemies[i].y;
      }
    })
    .then(function () {
      //Cuando tengamos la informacion actualizamos el mapa
      makeMap();
    })
    .catch((error) => {
      console.log(error);
    });
}
```

La qual esta lligada, amb la funció **makeMap**, que utilitzem per fer un mini mapa on es veuran els enemics i on ens podrem veure en tot moment.

```
function makeMap() {
  var table = document.getElementById("mapa");
  var tbody = table.createTBody();
  var old_tbody = document.getElementById("tbody");
  //eliminamos el mapa anterior
  table.replaceChild(tbody, old_tbody);
  tbody.setAttribute("id", "tbody");
  //nuestro mapa es de 5x5
  for (let i = 0; i < 5; i++) {
    var new_row = tbody.insertRow(0);
    for (let j = 0; j < 5; j++) {
      var casella = new_row.insertCell(-1);
      casella.setAttribute("id", "x" + j + "y" + i);
      casella.style.width = "30px";
      casella.style.height = "30px";

      //creamos una casilla con el suelo
      var empty = document.createElement("img");
      empty.src = "minimapa.png";
      empty.style.width = "30px";
      empty.style.height = "30px";
      empty.style.marginBottom = "none";
      casella.appendChild(empty);

      //si es la casilla central añadimos el sprite del jugador mirando a la direccion apropiada
      if (i == 2 && j == 2) {
        var player_img = document.createElement("img");
        switch (directions[0]) {···
        }
        player_img.style.width = "30px";
        player_img.style.height = "30px";
        player_img.style.position = "absolute";
        player_img.style.top = "50%";
        player_img.style.left = "50%";
        player_img.style.transform = "translate(-50%, -50%)";
        casella.appendChild(player_img);
      }
      //creamos los enemigos
      for (let n = 0; n < enemies_map.length; n++) {
        if (j == enemies_map[n].x - player.x + 2 && i == enemies_map[n].y - player.y + 2) {
          var enemy = document.createElement("img");
          var enemy_cont = document.createElement("div");
          //si el enemigo esta en el centro, utilizamos una imagen distinta para que no solape con el jugador
          if (j == 2 && i == 2) {
            enemy.src = "enemy_tiny.png";

          } else {
            enemy.src = "enemy.png";
          }
          enemy.style.width = "30px";
          enemy.style.height = "30px";
          enemy.style.position = "absolute";
          if (full) {
            enemy.style.transform = "translate(0%, -110%)";
          }
          enemy_cont.style.position = "absolute";
          enemy_cont.style.margin = "none";
          enemy_cont.style.width = "30px";
          enemy_cont.style.height = "30px";
          enemy_cont.appendChild(enemy);
          casella.appendChild(enemy_cont);
        }
      }
    }
  }
```

displayEnemy revisa la posició on es el jugador i cap a on esta mirant per veure si hi ha algun enemic.

```
//Comprueba si hay enemigo en frente del usuario y si lo hay lo muestra.
function displayEnemy() {
  for (let i = 0; i < enemies_close.length; i++) {
    if (((enemies_close[i].x == player.x + 1 && enemies_close[i].y == player.y && directions[0] == "E") ||
      (enemies_close[i].x == player.x - 1 && enemies_close[i].y == player.y && directions[0] == "O") ||
      (enemies_close[i].x == player.x && enemies_close[i].y == player.y + 1 && directions[0] == "N") ||
      (enemies_close[i].x == player.x && enemies_close[i].y == player.y - 1 && directions[0] == "S"))) {
      if (enemies_close[i].vitalpoints > 0) {···
      } else if (!enemies_close[i].equals(matar_enemy)) {
        console.log("aquest enemic ja esta mort ¯\\_(ツ)_/¯");
        //Guardamos el enemigo para saber que ya lo hemos visto.
        matar_enemy = { ...enemies_close[i] };
        //Enseñamos un mensaje informando al usuario de que el enemigo esta muerto.
        var text = document.createElement("p");
        text.style.position = "absolute";
        text.style.bottom = "500px";
        text.style.textAlign = "center";
        text.style.color = "wheat";
        text.style.fontFamily = "Garamond";
        text.innerHTML = "Aquest enemic ja esta mort";
        text.style.width = "100%";
        text.style.fontSize = "xx-large";
        text.style.fontWeight = "bolder";
        document.body.appendChild(text);
        fade(text);
      }
    }
  }
}
```

A més, per poder fer que el jugador interactiu amb el joc hem fet que cada fletxa tingui les seves funcions: la fletxa de dalt et permet moure el jugador cap endavant gracies a la api **movePlayer**, a no ser que hi hagi una paret en el cas de les fletxes dreta i esquerra hem implementat una opció que permet girar la visió del jugador i a demes hem fet que la tecla espai ens permeti atacar als enemics utilitzant la api de **attackEnemy**.

```
//Controla lo que sucede al darle a una tecla.
window.addEventListener("keydown", function (event) {
  let bruixula = document.getElementById("agulla");
  var enemy = document.getElementById("enemy");
  //el usuario se puede mover aunque este en modo "combate" si esta muerto
  if (!combat || dead) {
    switch (event.code) {
      case "ArrowLeft":···
      case "ArrowRight":···
      case "ArrowUp":···
      case "ArrowDown":···
    }

  } else {
    if (matar_enemy.vitalpoints <= 0) {···
    }

    //Se ataca con el espacio.
    if (event.code == "Space") {
      attackEnemy(player.token, directions[0]);

    }
  }
}, true);
```

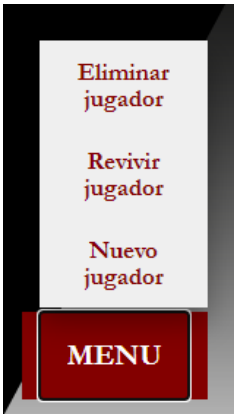
```
//Llamada a la api para mover al jugador
function movePlayer(token, direccion) {
  let url = "http://battlearena.danielamo.info/api/move/b89f96ae/" + token + "/" + direccion;
  fetch(url, {
    method: 'GET'
  })
    .then(function () {
      //llamamos a las apis una vez se haya acabado la llamada a esta api para actualizar la informacion local.
      getPlayersAndObjects(player.token);
      getPlayer(player.token);
    })
    .catch((error) => {
      console.log(error);
    });
}
```

```
//Llamada a la api para atacar al enemigo
function attackEnemy(token, direccion) {
  let url = "http://battlearena.danielamo.info/api/attack/b89f96ae/" + token + "/" + direccion;
  fetch(url, {
    method: 'GET'
  })
    .then(function (response) {
      console.log(response.status);
      return response.json();
    })
    .then(function (json) {···
    })
    .catch((error) => {
      console.log(error);
    });
}
```

Per lo que fa referencia al menú, hem pensat que en qualsevol moment l'usuari pot voler eliminar la partida o en cas que hagi mort, haurà de reviure al seu jugador. Per tan amb els botons del menú hem fet funcions que executen les apis. En el cas de eliminar jugador executem la api de **removePlayer,** i en el altre cas de voler reviure un jugador activem el **respawn** i després tornem a activar el getPlayer per actualitzar els estats**.**

```
//Llamada a la api para eliminar al jugador
function removePlayer(token, code) {
  var ajaxSYNC = function (url) {
    var request = function request(url) {
      var xhr = new XMLHttpRequest();
      xhr.open("DELETE", url, false);
      return xhr.responseText;
    }

    return {
      request: request
    }
  }();
  ajaxSYNC.request("http://battlearena.danielamo.info/api/remove/b89f96ae/" + token + "/" + code);
  console.log("removed.");
}
```



3. Tecnologies i eines

Per fer la practica Hem fet us d'elements HTML com botons, div, imgs... I els hem estilejat amb el CSS fent ús de flexbox i grid-layout per una correcta distribució dels elements dins dels seus contenidors o arreu de la pàgina.

Hem utilitzat fetch, per poder realitzar correctament les crides a les apis de forma asíncrona, i un XMLHTTPRequest síncron per a fer el remove, ja que vam intentar fer-ho de manera asíncrona però ens donava problemes.

Hem creat classes per a guardar el jugador i l'enemic,

També hem utilitzat el play per a reproduir sons.

El setInterval l'hem utilitzat per anar actualitzant el mapa cada 2 segons.

Hem utilitzat un listener per a permetre la interacció amb el teclat. Hem utilitzat event.code per a llegir de quina tecla es tractava.

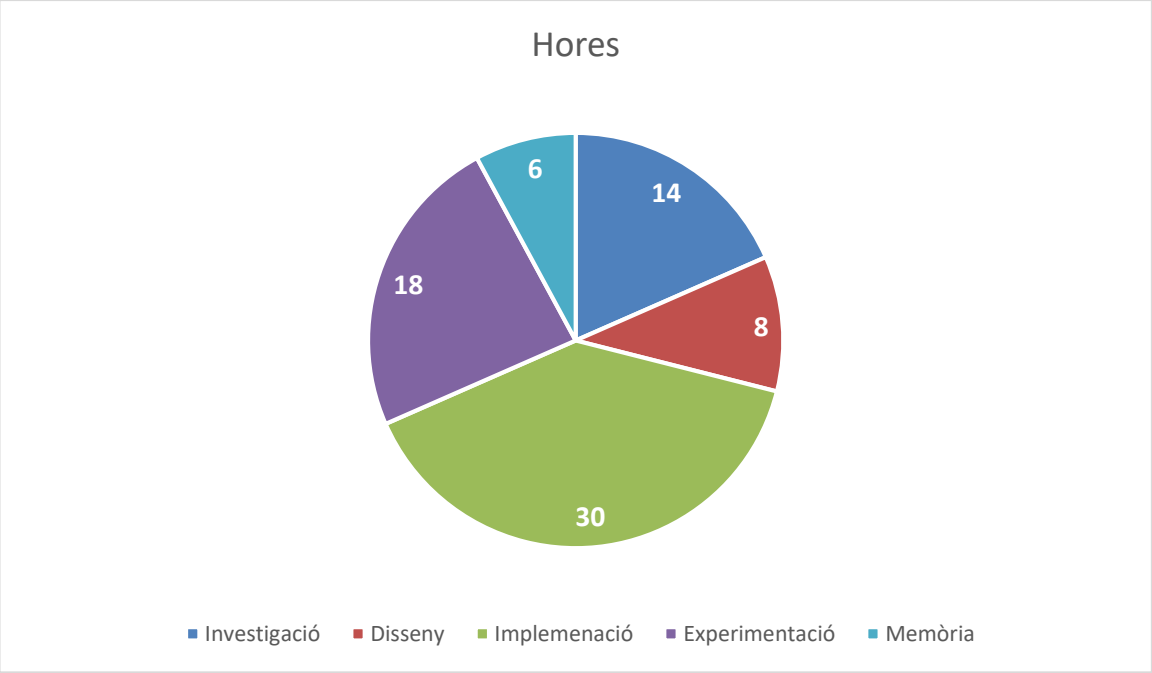
A més, hem creat molts elements HTML o els hem editat de manera dinàmica al JavaScript.

4. Costos

En quan als costos del projecte, hi ha diverses categories:

1. Investigació: temps de cerca online, revisió d'exercicis anteriors fets a classe i cerques a la informació proporcionada pels professors.
2. Disseny: tots els elements de la interfície com el menú, la barra dels estats, les pantalles emergents, el mini mapa...
3. Implementació: creació de les apis, afegir classes, confeccionar les funcions i altres.
4. Experimentació: comprovació de totes les apis, creació de jugadors per validar cada element creat: borar jugador, reviure jugador, atacar enemic...
5. Memòria: registre del treball realitzat.

Evidentment és difícil estimar el temps que hem dedicat a cada apartar, ja que casi les categories es feien simultàniament durant la realització de la practica, a demes del factor que el treball ha sigut de forma remota.



5. Conclusions

Durant el transcurs d'aquesta practica hem pogut assolir els coneixements que hem anat adquirint durant el semestre, ja que ha sigut una oportunitat per programar un joc online que té molts elements diferents. Com són el disseny del joc que hem pogut adaptar al nostre gust gràcies al CSS, la creació d'elements com el mini mapa que té molta informació i molts elements que valorar....

A més a més, es podria dir que aquesta pràctica ens ha ajudat a donar-nos compte del potencial que té el JavaScript respecte altres llenguatges de la programació per el disseny de pagines web. També hem pogut comprovar la gran utilitat d'ajuntar els tres llenguatges que hem après (HTML, CSS i JavaScript).

6. Competències

Competències generals

CG01	Capacitat d'anàlisi i síntesi.
CG06	Presa de decisions.

Competències específiques

CE2	Dissenyar, implementar i mantenir sistemes informàtics a través de l'aplicació apropiada dels paradigmes, entorns i llenguatges de programació, així com de les eines de suport afins.
CE4	Desenvolupar i utilitzar components software intercanviables, reutilitzables i definits per interfícies que permetin la composició i cooperació de sistemes informàtics.
CE6	Aplicar fonaments del disseny gràfic, la usabilitat i accessibilitat en el desenvolupament de solucions informàtiques que incrementin el grau de satisfacció i l'experiència d'ús de l'usuari.