Desenvolupament d'Aplicacions Web

Projecte DAW

Nom Projecte: La Pineda Adventure
Integrants: Adrià Aubanell Cabezas

Índex

pàg

Introducció

Quan ens van demanar que féssim un joc el qual empréssim totes les tecnologies estudiades durant el curs, no tenia molt clar ni el tipus de joc ni el tipus de tecnologia que utilitzaria.

Vaig pensar a fer el joc dels escacs, però vaig sentir que un altre company tenia pensat fer-ne un. També vaig pensar a fer alguna cosa amb IA, però les idees que sem ocorrien, ja estaven fetes i d'una manera molt millor a la que tenia pensada. I així successivament.

Llavors em va venir al cap la sortida que es va fer aquest curs a un escape room i la bona reacció dels companys / professorat que hi va anar. En aquell moment ja sabia el tema del joc, però ara em faltava saber on es faria el joc, hi vaig pensar que què millor que fer-la a l'institut.

El joc és un scaperoom el qual es fa dins del mapa de La Pineda. El jugador té 5 minuts per a poder respondre les preguntes de 5 professors els quals estan repartits dins del mapa i conseguir 5 claus. Per cada professor que encerti les seves preguntes, el jugador s'emporta una clau. Si el jugador obté les 5 claus en menys de 5 minuts, ha guanyat la partida. Però si passen els 5 minuts i el jugador encara no ha obtingut les 5 claus, vindrà el Marcel i el reganyarà.

Viabilitat del projecte.

Jo crec que aquest projecte és bastant viable. Després d'una llarga recerca per internet, no he trobat cap scape room similar al que he fet.

Respecte a el cost econòmic del projecte, en utilitzar tecnologies gratuïtes, el cost és nul.

Planificació

Com ja vaig comentar en el document 'proposta_projecte_laPinedaAdventure' i el document 'lliurament_intermedi_16_de_Maig', he implementat el mètode de treball SCRUM.

Pel que fa a les dates dels lliuraments, al principi vaig poder avançar una mica més del que tenia previst. En el document de proposta de projecte vaig dir:

'... és el dia 16 de Maig. Per aquesta data tinc pensat tenir fet la view index.html i tenir el mapa del joc i el personatge que es pugui moure (però el joc no serií funcional). ...'

Pel 15 de Maig, el projecte estava: el mapa estava acabat, el jugador es podia moure pel mapa, xocava quan ha de xocar (parets, persones), podia 'parlar' amb els alumnes (que li diguessin: 'Bon dia', etc.) i estava fent que pogués tenir una conversació amb algun professor i que aquest li fes preguntes i el jugador les respongués.

L'etapa entre el lliurament intermedi i el final, ha sigut bastant problemàtica. Vaig tenir problemes amb la interacció del jugador amb el professor i tot lo rel·lacionat amb Socket.io i MongoDB. Pel que fa al multiugador, al no tenir ni el temps ni el coneixement necessari per a realitzar-ho, he preferit no fer-ho.

Arquitectura

L'arquitectura és client servidor, fet amb Node.js. Per passar informació entre el client i el servidor he utilitzat Socket.io, i per a guardar, buscar, actualitzar o esborrar partides a la BD he utilitzat MongoDB.

URL

La url:

- https://github.com/lapineda201920/Projecte_M12

Requisits

Pel que fa a requisits, s'ha d'utilitzar un S.O. Ubunto. Per la resta de requisits, estan especificats al pdf '2~manual_d'usuari', situat en la mateixa carpeta que aquest document.

Disseny del model

- La BD que he utilitzat per al joc es diu : laPinedaAdventure:

```
> use laPinedaAdventure;
switched to db laPinedaAdventure
```

- La col·lecció que he utilitzat per guardar les partides es diu: jugadors:

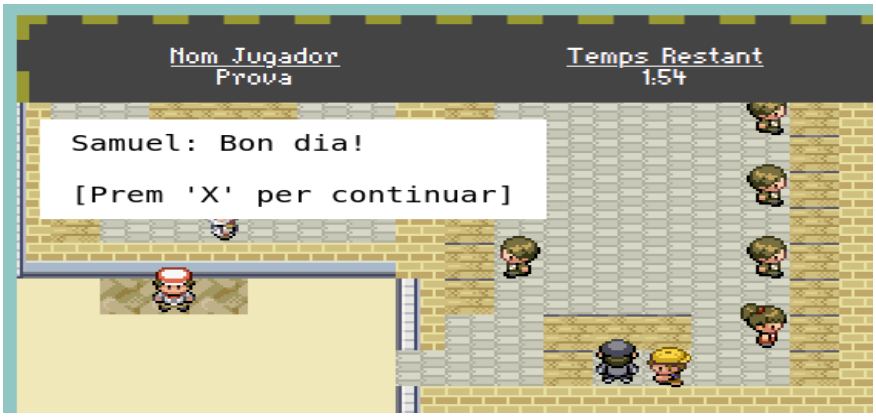
```
> db.jugadors.find().pretty();
{
        "_id" : ObjectId("5ed623b2d020812bbd03fbd3"),
        "Nom" : "CapturaPantalla",
        "Temps" : 298,
        "Claus" : 0,
        "X" : 26,
        "Y" : 7,
        "ClauSamuel" : false,
        "ClauOlga" : false,
        "ClauXavier" : false,
        "ClauSergi" : false,
        "ClauAlicia" : false
}
>
```

→ Al guardar una partida, guardo el nom del jugador, el temps restant que li queda, el total de claus obtingudes, la posició X i Y en el mapa, i de quin professor ja ha obtingut una clau. Aquest últim ho faig per a què el jugador, si guarda la partida, no pugui tornar a respondre les preguntes d'un mateix professor, per així poder obtenir un altre clau d'ell/a.

Casos d'ús / Diagrama de classes

El jugador pugui parlar amb els alumnes:

El jugador pugui parlar amb els professors:



El jugador tingui que xocar amb els elements (parets, alumnes, tutors):

Tecnologíes emprades

Pel que fa a les tecnologies emprades, he utilitzat HTML5, JS, Socket.io, MongoDB i CSS3 per la view index.html, la qual ens permet començar una nova partida o reprendre la partida prèviament guardada. Aquesta view, tant si comencem com si reprenem una partida, ens portarà a la view lapineda.html.

S'utilitza HTML5, JS, Phaser, Node.js, MongoDB i CSS3 per la view lapineda.html, la qual serà la view que ens mostrarà el mapa i la que ens deixarà jugar al joc. El Phaser l'utilitzem per a fer el joc, pel que fa a Node.js l'utilitzem per fer el servidor (localhost:8080), pel que fa a Socket.io l'utilitzem per passar informació a l'instant entre Client i Servidor, i per a poder guardar les dades de les partides utilitzem MongoDB.

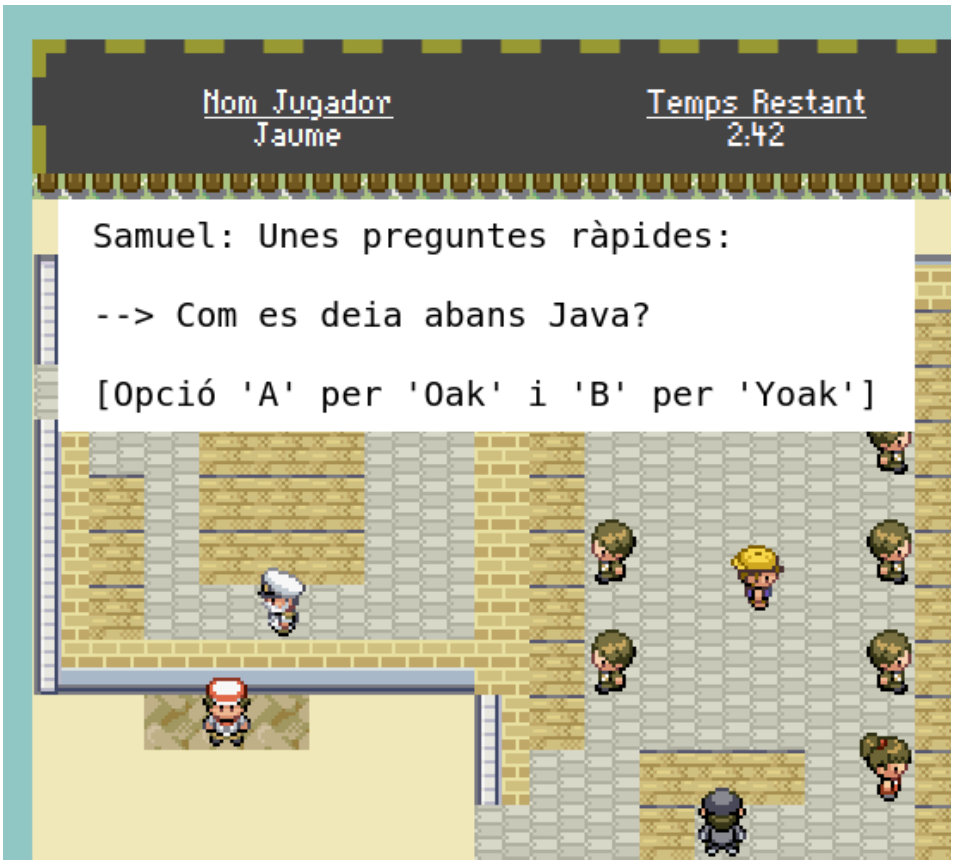
Per a fer el mapa del joc vaig utilitzar un programa anomenat Tiled i per a poder fer l'animació del jugador, vaig haver d'utilitzar un programa anomenat TexturePacker, ambdós de software lliure.

→ Tiled: https://www.mapeditor.org/

→ TexturePacker: http://free-tex-packer.com/

Errors detectats i Planificació/Millores de Futur

Pel que fa a errors, l'únic error que no he sabut com solucionar ha sigut que quan el jugador xoca amb un professor, i se'n va lluny d'ell, si prems la tecla D (com si volguessis parlar), et fa la conversació d'aquell professor:



** Com podem veure en aquesta captura, el jugador està tenint una conversació amb el professor Samuel, tot i que aquest no està al seu costat.*

Pel que fa a millores de futur seria solucionar aquest alerts els quals no afecten el joc, però que em molesten en la consola (sé que són deguts per la forma com llegeixo els fitxers en el servidor, però desconec com eliminar-los):

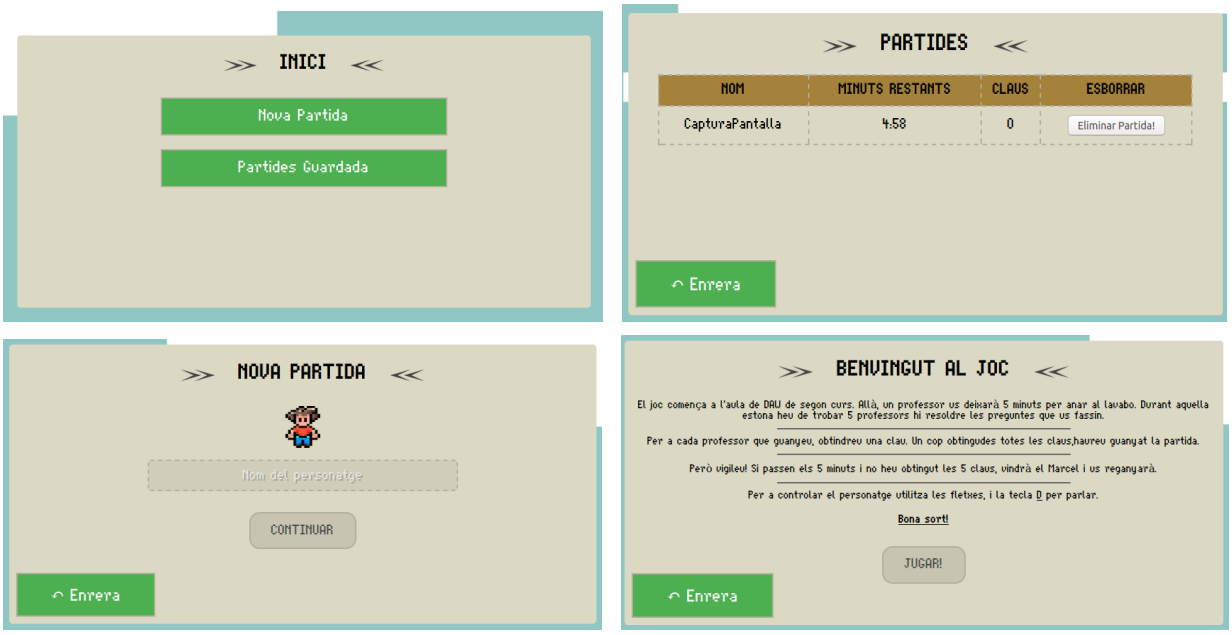
```
The script from "http://localhost:8080/js/phaser.js" was loaded even though its MIME type ("") is not a valid JavaScript MIME type. [Learn More]
The script from "http://localhost:8080/js/professors.js" was loaded even though its MIME type ("") is not a valid JavaScript MIME type. [Learn More]
The script from "http://localhost:8080/js/lapineda.js" was loaded even though its MIME type ("") is not a valid JavaScript MIME type. [Learn More]
The script from "http://localhost:8080/js/joc.js" was loaded even though its MIME type ("") is not a valid JavaScript MIME type. [Learn More]
The script from "http://localhost:8080/js/jugador.js" was loaded even though its MIME type ("") is not a valid JavaScript MIME type. [Learn More]
```

I poder tenir el codi més net, en concret el codi dins la funció update, on m'hagués agradat poder crear funcions externes per a posteriorment ser cridades, com seria el codi de caminar, l'animació de caminar, el parlar amb alumnes/professorat.

Explicació de com funciona el joc

index.html

- La view index.html es composa de 4 articles, els quals es mostren i s'amaguen depenent del que volguem fer (menuPrincipal, obrirPartida, crearPartida i informacioPartida).



Això ho controlem amb la funció mostra situada en el js index.js:

```
unction mostra(articleATapar, articleAMostrar) {

   var antic = document.getElementById(articleATapar);
   var nou = document.getElementById(articleAMostrar);

   if (nou.style.display == 'none') {

   nou.style.display = 'block';
   antic.style.display = 'none';
   }
   else {

   nou.style.display = 'none';
   antic.style.display = 'block';
   }

   // SI OBRE LES PARTIDES GUARDADES, OBRIM LA FUNCIÓ PER A CARREGAR-LES
   if(nou.style.display == "block" && articleAMostrar == "obrirPartida"){obrirPartida()};

   // SI TANCA LES PARTIDES GUARDADES, ELIMINEM ELS REGISTRES
   if(nou.style.display == "none" && articleAMostrar == "obrirPartida"){tancarPartida()};
}
```

Recuperar una Partida

Quan anem a crear una partida, en l'article informacioPartida al premer el botó 'JUGAR!', el que fem és cridar la funció agafarNom(), que agafa el nom que prèviament hem escrit i obre el html lapineda.html i li passem el nom del jugador:


```
function agafarNom(){

    var nom = document.getElementById('nomPersonatge').value;
    window.location.replace('lapineda.html?nom='+nom);
}
```

Mostrar les Partides

Pel que fa a l'article obrirPartida, que és l'article on podem veure les partides guardades en la BD, funciona de la següent manera. Quan la funció mostra() detecta que s'esta entrant a aquest article, crida a la funció obrirPartida(), i crida a la funció tancarPartida() quan detecta que s'està tancant l'article:

```
function mostra(articleATapar, articleAMostrar) {

    var antic = document.getElementById(articleATapar);
    var nou = document.getElementById(articleAMostrar);

    if (nou.style.display == 'none') {

    nou.style.display = 'block';
    antic.style.display = 'none';
    }
    else {

    nou.style.display = 'none';
    antic.style.display = 'block';
    }

    // SI OBRE LES PARTIDES GUARDADES, OBRIM LA FUNCIÓ PER A CARREGAR-LES
    if(nou.style.display == "block" && articleAMostrar == "obrirPartida"){obrirPartida()};

    // SI TANCA LES PARTIDES GUARDADES, ELIMINEM ELS REGISTRES
    if(nou.style.display == "none" && articleAMostrar == "obrirPartida"){tancarPartida()};
```

A la funció obrirPartida() lo primer quefem és cridar a la funció 'obrirPartidesGuardades', situada en el Servidor:

```
function obrirPartida(){

    // CRIDEM LA FUNCIÓ PERQUÈ ENS BUSQUI LES PARTIDES
    socket.emit('obrirPartidesGuardades', '--> Anem a buscar les partides guardades!');

    // REBEM LES PARTIDES
    socket.on('partidesGuardades', function (partides) {

        var registrePartida =   "<tr>" +
                                    "<th>NOM</th>" +
                                    "<th>MINUTS RESTANTS</th>" +
                                    "<th>CLAUS</th>" +
                                    "<th>ESBORRAR</th>" +
                                "</tr>";

        for (var i = 0; i < partides.length; i++){

            // CALCULEM EL TEMPS EN MINUTS I SEGONS
```

Imprimeix en la terminal el missatge '→ Anem a buscar les partides guardades!', i a continuació utilitza el MongoDB per a buscr totes les partides guardades. El resultat (variable 'result') el passem amb un socket.emit anomenat 'partidesGuardades':

```
----- SOCKET.IO CONNECTAT -----
--> Anem a buscar les partides guardades!
```

```
// PER A VEURE LES PARTIDES GUARDADES
socket.on('obrirPartidesGuardades', res => {

  console.log(res);

  MongoClient.connect(urlMongo, function(err, db) {
    if (err) throw err;
    var dbo = db.db("laPinedaAdventure");

    dbo.collection("jugadors").find({}).toArray(function(err, result) {
      if (err) throw err;
      socket.emit('partidesGuardades', result);
      db.close();
    });
  });
});
```

Aquest resultat el recull el socket.on situat en la funció obrirPartida(). Allà, creem l'estructura del html i amb JS DOM imprimim els registres creats a la taula situada en el html (la qual està buida):

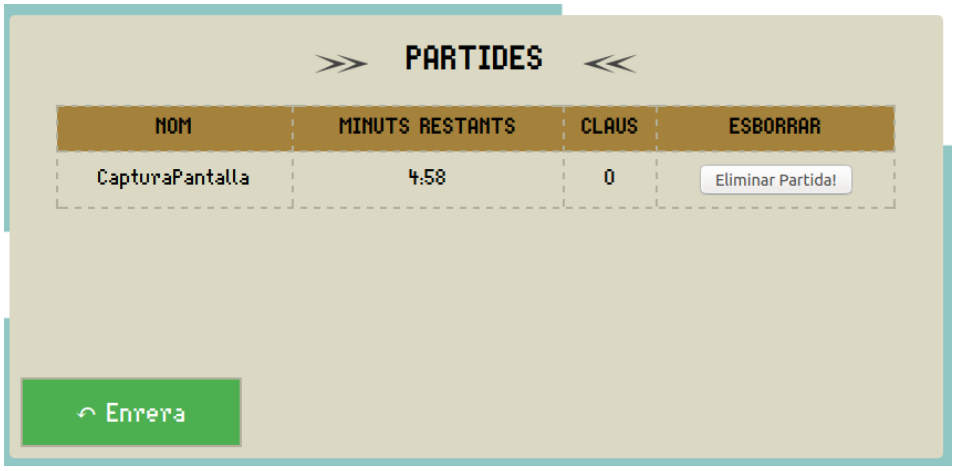
```
// REBEM LES PARTIDES
socket.on('partidesGuardades', function (partides) {

    var registrePartida =   "<tr>" +
                                "<th>NOM</th>" +
                                "<th>MINUTS RESTANTS</th>" +
                                "<th>CLAUS</th>" +
                                "<th>ESBORRAR</th>" +
                            "</tr>";

    for (var i = 0; i < partides.length; i++){

        // CALCULEM EL TEMPS EN MINUTS I SEGONS
        var minuts = Math.floor(partides[i]["Temps"] / 60);
        var segons = partides[i]["Temps"] - minuts * 60;

        // Si el segon és 0, perquè no es vegi així (5:0), li afegeixo un 0 extra (5:00)
        var extra = "";
        if (segons <= 9){

            extra = "0";
        }


        // HO GUARDEM DINS LA VARIABLE
        registrePartida +=  "<tr>" +
                                "<td onclick='recuperarPartida(this.id)' id='" + partides[i]["_id"] + "'>" + part
                                "<td onclick='recuperarPartida(this.id)' id='" + partides[i]["_id"] + "'>" + minu
                                "<td onclick='recuperarPartida(this.id)' id='" + partides[i]["_id"] + "'>" + part
                                "<td><button type='button' onclick='eliminarPartida(`" + partides[i]["_id"] + "`)
                            "</tr>";
    }

    // IMPRIMIM LA VARIABLE
    var table = document.getElementsByTagName("table")[0];
    table.innerHTML = registrePartida;
});
```

I amb la funció tancarPartida() el que fem és eliminar tots els fills que té la taula fins que no en quedi cap. Així, si de nou tornem a obrir l'article, sols mostrarà les partides actuals en la BD:

```
function tancarPartida(){

    var table = document.getElementsByTagName("table")[0];
    var child = table.lastChild;
    while (child) {
        table.removeChild(child);
        child = table.lastChild;
    }
};
```

Eliminar una Partida

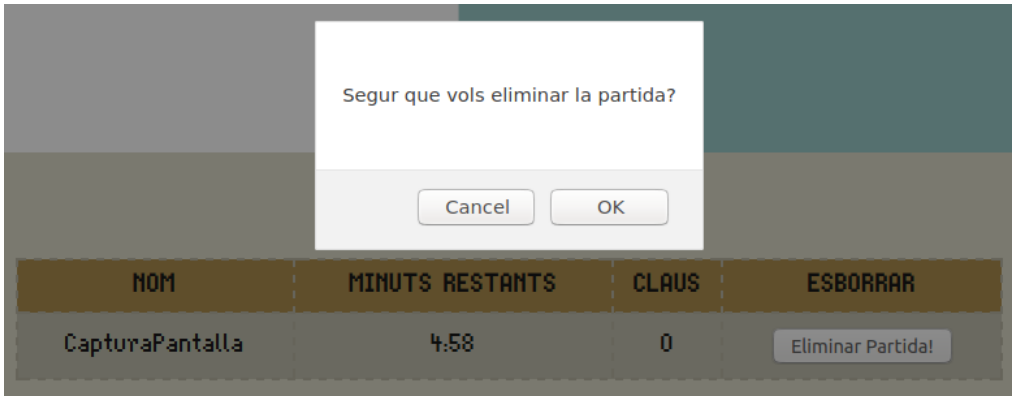
Quan volem eliminar una partida, ho podem fer des del html:



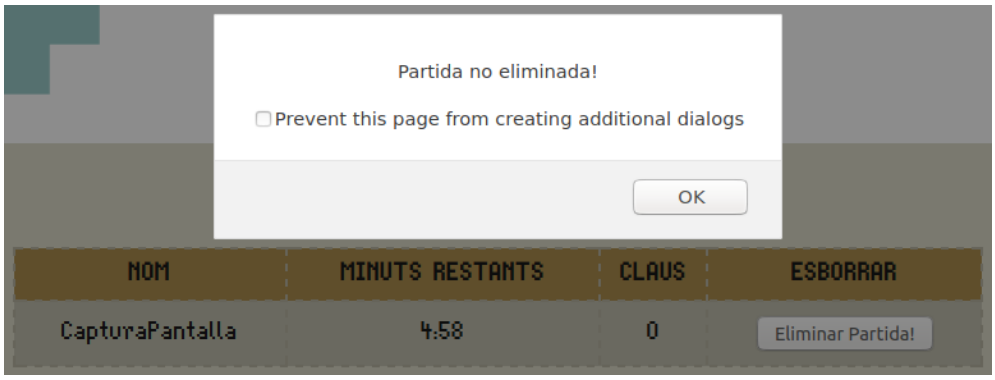
Al premer el botó 'Eliminar Partida!', el que fa és cridar a una funció anomenada eliminarPartida(), la qual abans de eliminar-la, ens fa confirmar la nostre el·lecció:

```
function eliminarPartida(idPartida){

    var confirmacio = confirm("Segur que vols eliminar la partida?");

    if (confirmacio){

        // CRIDEM LA FUNCIÓ PERQUÈ ENS ELIMINI LA PARTIDA
        socket.emit('eliminarPartidaGuardada', idPartida);

        // BORREM TOTES LES PARTIDES DEL HTML
        tancarPartida();

        // IMPRIMIM TOTES LES PARTIDES QUE ESTAN A LA BD AL HTML
        obrirPartida();
    }
    else{

        alert("Partida no eliminada!")
    }

}
```

En cas de prèmer el botó, ens mostrarà lo següent:



En cas de prèmer el botó ‘Cancel’, ens farà saber que no s’ha eliminat:



En cas de prèmer el botó ‘OK’, eliminarà la partida directament:

Recuperar una Partida

En canvi, si el que volem és recuperar una partida, el que hem de fer és premer sobre la partida en qüestió. Això fara que s'executi la funció recuperarPartida(), obrirà el html lapineda.html i li passarà en comptes del nom del jugador, la id de la partida:

```
function recuperarPartida(idPartida){

    window.location.replace('lapineda.html?id='+idPartida);
}
```

laPineda.html

- La view lapineda.html es composa d'un que fa de contenidor i en el seu interior conté 2 's. El primer és on tenim la informació de la partida (nom del jugador, temps restant i claus obtingudes). El segon solament és un div buit amb una id, el qual amb Phaser li possarem el joc.

1) Pel que fa a les dades del primer div, per a introduïr els valors a les variables, comprobem si se li passa per GET un nom o una id.

→ Si li passem una id, busquem amb la funció 'obrirPartidaGuardada' situada en el Servidor la informació de la partida (nom, temps i claus).

→ Si li passem un nom, deixa el nom buit, introduim un -1 en les claus obtingudes i li possem de temps 301 segons, lo qual equival 5 minuts i 1 segon.

```
// --> Variables
    var url = window.location.search;
    variable = url.split('=');

    // SI LO QUE PASSEM PER GET ÉS LA ID DE LA PARTIDA, RECUPERA-LA
    if (variable[0] == "?id"){

        // CRIDEM LA FUNCIÓ PERQUÈ ENS BUSQUI LA PARTIDA
        socket.emit('obrirPartidaGuardada', variable[1]);

        // REBEM LA INFORMACIÓ DE LA PARTIDA
        socket.on('partidaGuardada', function (partida) {

            nomJugador = partida[0]["Nom"];
            claus = partida[0]["Claus"] - 1;
            temps = partida[0]["Temps"] + 1;
        });
    }
    // SINÒ, CREA UNA DE NOVA
    else{

        nomJugador = "";
        claus = -1;
        temps = 301; // 301= 5:01 minuts
    }
```

A continuació, cridem la funció restarTemps(), on s'anirà cridant cada 1 segon, per així fer el compte enrrere:

```
var intervalTemps = setInterval(restarTemps,1000);
```

Lo primer que farem, és restar 1 segon (per això prèviament li hem sumat 1 segon). Convertim els segons en minuts i segons restants. Aquest, els imprimim al html. Si el temps és 0, mostrem la imatge de que s'ha perdut, netejem el interval creat per a què s'executés cada un segon, si la partida estava guardada a la BD la eliminem, parem el joc i ens parem 5 segons (temps per a que el jugador pogui llegir el text dins la imatge), i a continuació va al HTML index.html:

```
function restarTemps() {

    temps--;
    var minuts = Math.floor(temps / 60);
    var segons = temps - minuts * 60;

    // Si el segon és 0, perquè no es vegi així (5:0), li afegeixo un 0 extra (5:00)
    var extra = "";
    if (segons <= 9){

        extra = "0";
    }

    document.getElementById("temps").innerHTML = "<u>Temps Restant</u><br>"+minuts+":"+extra+segons;


    if (temps == 0) {

        // MOSTREM LA IMATGE DEL RESULTAT
        var resposta = document.createElement("img");
        resposta.setAttribute("src", "img/game_over.jpg");
        resposta.setAttribute("width", "70%");
        resposta.setAttribute("height", "670");
        resposta.setAttribute("alt", "Game Over");
        resposta.style.position = "fixed";
        resposta.style.zIndex = "5";
        resposta.style.top = "25%";
        resposta.style.margin = "0px 0px 0px 200px";
        document.body.appendChild(resposta);

        // PAREM EL TEMPS
        clearInterval(intervalTemps);

        // ESBORREM LA PARTIDA DE LA BD (SI ÉS QUE ESTAVA GUARDADA)
        if (variable[0] == "?id"){

            socket.emit('eliminarPartidaGuardada', variable[1]);
        }

        // PAREM EL JOC, PER LO QUE EL JUGADOR NO ES PODRÀ MOURE
        game.destroy();

        // I AL CAP DE 5 SEGONS, CANVIEM LA PÀGINA PER LA D'INICI
        setTimeout(function(){
            window.open("index.html", "_self");
        }, 5000);
    }
}
```

```
// PER A ESBORRAR LA PARTIDA, JA SIGUI PERQUÈ S'HAGI GUANYAT O PERDUT
socket.on('eliminarPartidaGuardada', idPartida => {

  MongoClient.connect(urlMongo, function(err, db) {
    if (err) throw err;
    var dbo = db.db("laPinedaAdventure");

    var o_id = mongo.ObjectID(idPartida);
    dbo.collection("jugadors").deleteOne({"_id": o_id}, function(err, result) {
      if (err) throw err;
      console.log("--> Partida Borrada!");
      db.close();
    });
  });
});
```

Un cop la pàgina s'hagi carregat, cridem les funcions nom() i sumarClaus():

```
// Un cop la pàgina s'hagi carregat del tot
window.onload=function (){

    // NOM JUGADOR
    nom();

    // CLAUS
    sumarClaus();
};
```

Si la variable que li hem passat per GET no és la id (per lo que serà el nom), extreiem el nom de la URL i li inserim a la variable nom (per això prèviament ho haviem deixat buit). A continuació, li haguem passat per GET una id o un nom, imprimim això el nom en el HTML.

```
function nom(){

    if (variable[0] != "?id"){

        nomJugador = decodeURI(variable[1]);
    }

    document.getElementById("nomJugador").innerHTML = "<u>Nom Jugador</u><br>"+nomJugador;
}
```

A continuació, quan cridem la funció sumarClaus(), lo primer que fem és sumar una clau a la variable 'claus' (per això previamnet li possem un valor de -1 si li passem per GET un nom). Imprimim al HTML el total de claus obtingudes. Si el número de claus és igual a 5, mostrem la imatge de que s'ha guanyat la partida, esborrem la partida de la BD si aquesta estava guardada, parem el joc i ens parem 5 segons (temps per a que el jugador pogui llegir el text dins la imatge), i a continuació va al HTML index.html:

```
function sumarClaus() {

    claus++;
    document.getElementById("claus").innerHTML = "<u>Claus</u><br>"+claus+"/5";


    if (claus == 5) {

        // MOSTREM LA IMATGE DEL RESULTAT
        var resposta = document.createElement("img");
        resposta.setAttribute("src", "img/win.jpg");
        resposta.setAttribute("width", "70%");
        resposta.setAttribute("height", "670");
        resposta.setAttribute("alt", "Game Over");
        resposta.style.position = "fixed";
        resposta.style.zIndex = "5";
        resposta.style.top = "25%";
        resposta.style.margin = "0px 0px 0px 200px";
        document.body.appendChild(resposta);

        // ESBORREM LA PARTIDA DE LA BD (SI ÉS QUE ESTAVA GUARDADA)
        if (variable[0] == "?id"){

            socket.emit('eliminarPartidaGuardada', variable[1]);
        }

        // PAREM EL JOC, PER LO QUE EL JUGADOR NO ES PODRÀ MOURE
        game.destroy();

        // I AL CAP DE 5 SEGONS, CANVIEM LA PÀGINA PER LA D'INICI
        setTimeout(function(){
            window.open("index.html", "_self");
        }, 5000);
    }
}
```

2) Pel que fa al segon div, anem al JS joc.js. En aquest JS, creem el joc. Primer de tot, creem la configuraió necessaria:

Li indiquem el width i height de la pantalla del joc, on ha d'imprimir el joc (en el nostre cas en el amb id='contenidor'), li diem que és un joc de pixels, indiquem quins passos ha de seguir per executar el codi correctament (preload, create i update), i el tipus de joc que és, en el nostre cas és un arcade amb 0 gravetat:

```
// CONFIGURACIÓ DE COM S'EXECUTARÀ PHASER
const config = {
  type: Phaser.AUTO,
  width: 844,
  height: 470,
  parent: "contenidor", // --> CONTANIDOR ON S'IMPRIMIRÀ EL JOC
  pixelArt: true,
  scene: { // --> ORDRE DE CÀRREGA DE LES FUNCIONS
    preload: preload,
    create: create,
    update: update
  },
  physics: {
    default: "arcade",
    arcade: {
      gravity: { y: 0 } // --> AL SER UN JOC EN VISTA D'OCELL, NO HI HA D'HAVER GRAVETAT, PER LO QUE POSSAREM 0
    }
  }
};
```

A continuació creem les variable globals que anirem utilitant al llarg del joc.:
→ La primera imatge podem veure variables com la creació del joc amb la configuració prèviament feta, creació de variables de tecles i jugador, contador de preguntes totals i preguntes accertades, temps de resposta, i resposta del jugador. També variables per saber si s'ha xocat amb algun alumne.

→ I en la segona imatge, podem veure variables per saber si el jugador xoca amb un professor, o si aquest ja li ha donat la clau, i també la creació de les variables de possició del jugador:

```
// DECLAREM EL JOC I LES VARIABLES
const game = new Phaser.Game(config);
let cursors;
let jugador;


var contador = contadorAccertades = 0;
var hiParla = false;
var tempsResposta = null;
var respostaJugador = 0;

/*********ALUMNE***********/
var xocaAlumne = false;
```

```
/**********SAMUEL**********/
xocaSamuel = clauSamuel = false;

/**********OLGA**********/
xocaOlga = clauOlga = false;

/**********XAVIER**********/
xocaXavier = clauXavier = false;

/**********SERGI**********/
xocaSergi = clauSergi = false;

/**********ALICIA**********/
xocaAlicia = clauAlicia = false;

/*********POSICIÓ JUGADOR**********/
x = y = 0;
```

Abans de començar a fer el joc, mirem si el que passem per GET és una id. En cas afirmatiu, busquem la informació de la partida i li donem els valors a les variables prèviament creades.

```
var url = window.location.search;
variable = url.split('=');

// SI LO QUE PASSEM PER GET ÉS LA ID DE LA PARTIDA, RECUPERA-LA
if (variable[0] == "?id"){

  // CRIDEM LA FUNCIÓ PERQUÈ ENS BUSQUI LA PARTIDA
  socket.emit('obrirPartidaGuardada', variable[1]);

  // REBEM LA INFORMACIÓ DE LA PARTIDA
  socket.on('partidaGuardada', function (partida) {

    clauSamuel = partida[0]["ClauSamuel"];
    clauOlga = partida[0]["ClauOlga"];
    clauXavier = partida[0]["ClauXavier"];
    clauSergi = partida[0]["ClauSergi"];
    clauAlicia = partida[0]["ClauAlicia"];

    x = partida[0]["X"] * 848.986 / 26;
    y = partida[0]["Y"] * 236.467 / 7;
  });
}
```

S'executa la primera funció, preload(). Aquí el que fem és carregar la imatge que el mapa utilitzarà per pintar els tiles del mapa, i carreguem la imatge i animació del jugador.

```
function preload() {

  // 1.A - CARREGUEM EL MAPA I LA IMATGE D'ON TREIEM LES 'TILES'
  this.load.image("tiles", "img/tilesets/tileset-shinygold2.png");
  this.load.tilemapTiledJSON("map", "mapes/mapa_la_pineda.json");

  // 1.B - CARREGUEM LES IMATGES DEL JUGADOR
  this.load.atlas("ninoJugador", "img/jugador.png", "img/jugador.json");
}
```

S'executa la segona funció, create(). Aquí el que fem és crear el mapa amb el mapa prèviament carregaat, afegim la imatge que ens servirà per pintar els tiles del mapa i indiquem les capes que volem que el joc mostri.

```
function create() {

  // 2 - CREEM COM UN MAPA DE TILE EL MAPA PRÈVIAMENT CARREGAT
  const map = this.make.tilemap({ key: "map" });


  // 3 - AFEGIM LES IMATGES EN ELS 'TILES' CORRESPONENTS
  const tileset = map.addTilesetImage("tileset-shinygold2", "tiles");


  // 4 - INDIQUEM QUINES CAPES VOLEM QUE MOSTRI
  const terra = map.createStaticLayer("Capa terra", tileset, 0, 0);
  const arbres = map.createStaticLayer("Capa arbres", tileset, 0, 0);
  const superficies = map.createStaticLayer("Capa superficies", tileset, 0, 0);
  const persones = map.createStaticLayer("Capa persones", tileset, 0, 0);
  const samuel = map.createStaticLayer("Capa Samuel", tileset, 0, 0);
  const olga = map.createStaticLayer("Capa Olga", tileset, 0, 0);
  const xavier = map.createStaticLayer("Capa Xavier", tileset, 0, 0);
  const sergi = map.createStaticLayer("Capa Sergi", tileset, 0, 0);
  const alicia = map.createStaticLayer("Capa Alicia", tileset, 0, 0);
  const cafeteria = map.createStaticLayer("Capa Cafeteria", tileset, 0, 0);
```

A continuació, li indiquem que agafi l'objecte 'Lloc Spawn' del mapa (creat a l'hora de fer el mapa amb Tiled). Li diem que en cas que el que es passi per GET sigui diferent a una id (per lo qual serà el nom = partida nova), li donem a les variables x i y el valor de llocspawn.x i y. Ara ja sabent la posició del jugador al començar la partida, li diem sobre quines capes té que xocar. I en el cas de les capes dels professors, activem una variable anomenada xocaPROFE que ens serà útil més tard.

```
// 5 - LOCALITZEM EL LLOC ON FARÀ SPAWN EL JUGADOR
const llocSpawn = map.findObject("Lloc Spawn", obj => obj.name === "Lloc Spawn");


// SI LO QUE PASSEM PER GET NO ÉS LA ID DE LA PARTIDA, CREEM LES POSICIONS X I Y
if (variable[0] != "?id"){

  x = llocSpawn.x;
  y = llocSpawn.y;
}

// 6 - LI AFEGIM FÍSICA AL JUGADOR (LLOC ON APAREIXARÀ-X, LLOC ON APAREIXARÀ-Y, "", "")
jugador = this.physics.add.sprite(x, y, "ninoJugador", "jugador-esquerra.png").setSize(30, 30).setOffset(0, 10);

this.physics.add.collider(jugador, arbres);
this.physics.add.collider(jugador, superficies);
this.physics.add.collider(jugador, persones, function(){ xocaAlumne = true;});
this.physics.add.collider(jugador, samuel, function(){ xocaSamuel = true;});
this.physics.add.collider(jugador, olga, function(){ xocaOlga = true;});
this.physics.add.collider(jugador, xavier, function(){ xocaXavier = true;});
this.physics.add.collider(jugador, sergi, function(){ xocaSergi = true;});
this.physics.add.collider(jugador, alicia, function(){ xocaAlicia = true;});
this.physics.add.collider(jugador, cafeteria, function(){ xocaCafeteria = true;});
```

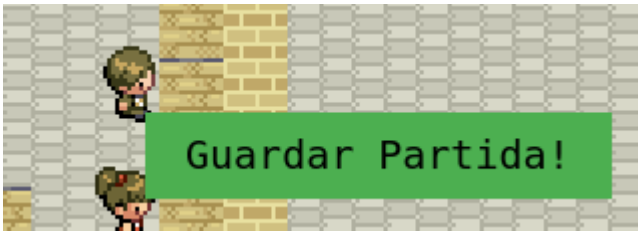
Acabem d'especificar quines capes volem que sí que tingui col·lisió, i creem els frames amb les imatges corresponents per l'animaci del jugador.

```
// 7 - LI DIEM QUINES CAPES VOLEM QUE S'EXECUTI EL 'COLLIDES'
arbres.setCollisionByProperty({ collides: true });
superficies.setCollisionByProperty({ collides: true });
persones.setCollisionByProperty({ collides: true });
samuel.setCollisionByProperty({ collides: true });
olga.setCollisionByProperty({ collides: true });
xavier.setCollisionByProperty({ collides: true });
sergi.setCollisionByProperty({ collides: true });
alicia.setCollisionByProperty({ collides: true });
cafeteria.setCollisionByProperty({ collides: true });


// 8 - CREEM ELS FRAMES AMB LES IMATGES CORRESPONENTS (PRÈVIAMENT CARREGADES AMB EL JSON)
this.anims.create({
  key: "jugador-esquerra-caminar",
  frames: this.anims.generateFrameNames("ninoJugador", {
    prefix: "jugador-esquerra-caminar.",
    start: 0,
    end: 3,
    zeroPad: 3,
    suffix: ".png"
  }),
  frameRate: 10,
  repeat: -1
});

this.anims.create({
  key: "jugador-dreta-caminar"
```

Per acabar, li indiquem a la càmera que segueixi el jugador , i escribim text a la pantalla. En el text guardar, li indiquem que és interactiu.

```
// 9 - LI INDIQUEM QUE ENS MOSTRI EL RESULTAT A PANTALLA
const camera = this.cameras.main;
camera.startFollow(jugador);
camera.setBounds(0, 0, map.widthInPixels, map.heightInPixels);
cursors = this.input.keyboard.createCursorKeys();


// 10 - ESCRIBIM TEXT A LA PANTALLA
informacioMoure = this.add.text(15, 15, "Utilitza les fletxes per moure't, i la 'D' per parla
resposta = this.add.text(15, 15, "Persona: !", { font: "20px monospace", fill: "#000000", pad
guardar = this.add.text(590, 410, "Guardar Partida!", { font: "20px monospace", fill: "#000000

guardar.setInteractive(new Phaser.Geom.Rectangle(0, 0, guardar.width, guardar.height), Phaser
```

S'executa la tercera i última funció, update(). Aquí el que fem és que quan el jugador premi el botó 'Guardar Partida!', capturem la posició actual del jugador, creem un nou jugador amb tota la informació obtenida fins ara i executem la funció de la classe jugador perquè ens guardi la partida.




```
function update(time, delta) {


  guardar.on('pointerdown', function () {

    // CAPTUREM LA POSICIÓ ACTUAL DEL JUGADOR
    let posicioX = Math.round(jugador.x/33);
    let posicioY = Math.round(jugador.y/33);

    let informacioJugador = new Jugador(nomJugador, temps, claus, posicioX, posicioY, clauSamuel, clauOlga, cl
    informacioJugador.guardarPartida(variable);
  });
```

A continuació li inidiquem la velocitat a la que el jugador s'ha de moure, possem la velocitat a 0, i mirem si el cursor (les fletxes del teclat), estàn siguent premudes, en cas afirmatiu, possem velocitat al jugador.

```
const velocitat = 180;
const velocitatPrevia = jugador.body.velocity.clone();

// 11 - PAREM QUALSEVOL MOVIMENT ABANS DEL ÚLTIM FRAME
jugador.body.setVelocity(0);


// 12 - MOVIMENT HORITZONTAL
if (cursors.left.isDown) { jugador.body.setVelocityX(-velocitat);}
else if (cursors.right.isDown) { jugador.body.setVelocityX(velocitat);}


// 13 - MOVIMENT VERTICAL
if (cursors.up.isDown) { jugador.body.setVelocityY(-velocitat);}
else if (cursors.down.isDown) { jugador.body.setVelocityY(velocitat);}


// 14 - NORMALITZAR LA VELOCITAT PERQUÈ EL JUGADOR NO ES MOGUI MÉS DEPRESSA EN DIAGONAL
jugador.body.velocity.normalize().scale(velocitat);
```

En el cas que s'estiguin prement les fletxes, imprimim la imatge corresponent del jugador. (Exemple, si s'està prement la flexta esquerra , l'imprimirem caminant a mà esquerra).

```
// 15 - ACTUALITZEM L'ANIMACIÓ SI CAMINA ...
if (cursors.left.isDown) { jugador.anims.play("jugador-esquerra-caminar", true);}
else if (cursors.right.isDown) { jugador.anims.play("jugador-dreta-caminar", true);}
else if (cursors.up.isDown) { jugador.anims.play("jugador-esquena-caminar", true);}
else if (cursors.down.isDown) { jugador.anims.play("jugador-davant-caminar", true);}
else{

  // 16 - SINÓ CAMINA, LA PAREM E POSSEM UNA IMATGE ON NO CAMINI
  jugador.anims.stop();

  if (velocitatPrevia.x < 0){ jugador.setTexture("ninoJugador", "jugador-esquerra.png");}
  else if (velocitatPrevia.x > 0){ jugador.setTexture("ninoJugador", "jugador-dreta.png");}
  else if (velocitatPrevia.y < 0) { jugador.setTexture("ninoJugador", "jugador-esquena.png");}
  else if (velocitatPrevia.y > 0) { jugador.setTexture("ninoJugador", "jugador-davant.png");}
}
```

En el cas que els cursors estàn sent premuts i que el jugador prem la telca D, no mostrem cap missatge. Aquesta és la solució que vaig trobar per a que si el jugador prem la tecla D quan no està al costat d'algúna persona, no mostri cap missatge.

```
if (cursors.left.isDown || cursors.right.isDown || cursors.up.isDown || cursors.down.isDown){

  this.input.keyboard.once("keydown_D", event => {

    informacioMoure.setVisible(true);
    resposta.setVisible(false);
    clearTimeout(tempsResposta);
    cursors = this.input.keyboard.createCursorKeys();
  });
}
```

Si el jugador a col·lisionat amb un alumne, i prem la tecla D, la informació de com moure's desapareixerà, es crearà un array de possibles respostes dels alumnes, es crea un número aleatori per a què en seleccionem una, i la imprimim a pantalla. Al cap de 2 segons, treiem la resposta del alumne i possem la informació de com moure's pel mapa.


```
// 17 - SI EL JUGADOR XOCA / ESTÀ AL COSTAT D'ALGUN --ALUMNE--
if (xocaAlumne){

  xocaAlumne = false;
  // I SI PREM LA TECLA D
  this.input.keyboard.once("keydown_D", event => {

    informacioMoure.setVisible(false);

    var respostesAlumnes = [
      "Quina calor que fa", "Bon dia", "Aquesta pràctica és molt complicada",
      "No trobo les ulleres", "Has vist al Samuel?", "Has vist a l'Olga?",
      "Has vist al Marcel?", "CORONAVIRUS!", "Saps com sol·lucionar l'error de Dockers?",
      "Ja has fet la pràctica?", "Avui hi ha entrepà de pinxos?", "Ufff..."
    ];
    var numeroRespostaAleatoria = Math.floor((Math.random() * respostesAlumnes.length));
    resposta.text = "Alumne: "+respostesAlumnes[numeroRespostaAleatoria];

    resposta.setVisible(true);

    tempsResposta = setTimeout(function(){
      informacioMoure.setVisible(true);
      resposta.setVisible(false);
      },2000);
  });
}
```

Similar és també l'estructura que li he aplicat als professors. Si ens recordem, quan el jugador col·lisiona amb un professor, la seva variable es posa a true. Quan és el cas, entra dins del if(), genera les preguntes, ajudes per a contestar i les respostes, i amb aquesta informació crea un nou professor.

```
// 18 - SI EL JUGADOR XOCA / ESTÀ AL COSTAT D'ALGUN --PROFESSOR--
// --> SAMUEL <--//
if (xocaSamuel){


  // PREGUNTES QUE FARÀ EL PROFESSOR
  var preguntes = [
    "Com es deia abans Java?", "Quantes variables hi ha a Java?",
    "Quants alumnes hi ha a la cafeteria?", "Quina és la mascota de PHP?"
  ];
  var ajudes = [
    "Opció 'A' per 'Oak' i 'B' per 'Yoak'", "Opció 'A' per '5' i 'B' per '4'",
    "Opció 'A' per '14' i 'B' per '12'", "Opció 'A' per 'Elefant Blau' i 'B' per 'Ratolí blau'"
  ];
  var respostes = [
    1, 1,
    2, 1
  ];

  var samuel = new Professor("Samuel", preguntes, ajudes, respostes, xocaSamuel, clauSamuel);
```

Si el jugador prem la tecla D, el professor li doanrà conversació. En cas que premi la X, significarà que es continua la conversació (per lo qual hi parlaràn).

```
// --> SI PREM LA TECLA D
this.input.keyboard.once("keydown_D", event => {

  samuel.conversacioProfe();
});

// --> SI PREM LA TECLA X
this.input.keyboard.once("keydown_X", event => {

  hiParla = true;
});
```

Si hi parla, continuem la conversacció.

```
if (hiParla){

  samuel.continuacioConversacio();
```

Si la clau del professor, en aquest cas del Samuel, encara no ha sigut donada, li passem el contador de preguntes realitzades per part del professor, i dins de la funció de la classe pregunta(), el professor li fa una pregunta al alumne. Si el jugador prem les tecles A o B, aquestes obtindràn un valor a 1 o 2 respectivament.

```
if (!clauSamuel){

  samuel.pregunta(contador);

  this.input.keyboard.once("keydown_A", event => {

    respostaJugador = 1;
  });
  this.input.keyboard.once("keydown_B", event => {

    respostaJugador = 2;
  });
```

Si s'ha donat una resposta per part del jugador (1 o 2), si la resposta donada és igual que la del array, sumem al contador d'accertades un +1. Si aquest contador arriba a 4, significaría que les ha respost totes correctament, per lo que el professor li faría entrega de la clau, per lo que la variable clauSamuel, en aquest cas, serà true, per lo que no podrà el jugador fer-li preguntes al professor. Si el contador arriba a 3, significaría que les reguntes s'han acabat, i que per tant el jugador ha fallat en alguna de les respostes.

```
if (respostaJugador == 1 || respostaJugador == 2){

  // SI LA RESPOSTA ÉS CORRECTE
  if (respostaJugador == respostes[contador]){

    contadorAccertades++;

    // I SI HA RESPÓS TOTES LES PREGUNTES
    if (contadorAccertades >= 4){

      cursors = this.input.keyboard.createCursorKeys();
      samuel.fiPreguntes();
      hiParla = false;
      clauSamuel = true; // POSSEM TRUE, CONFORME EL PROFESSOR JA NO POT TORNAR A DONAR UNA CLAU
      xocaSamuel = false;
      contador = -1;
      contadorAccertades = 0;

      tempsResposta = setTimeout(function(){
        informacioMoure.setVisible(true);
        resposta.setVisible(false);
        },4000);
    }
  }

  if(contador >= 3){

    samuel.error();
    cursors = this.input.keyboard.createCursorKeys();
    hiParla = xocaSamuel = false;
    contador = -1;
    contadorAccertades = 0;

    tempsResposta = setTimeout(function(){
      informacioMoure.setVisible(true);
      resposta.setVisible(false);
      },2000);
  }

  contador++;
  respostaJugador = 0;
}
```

Però, si a la pàgina 31, no és així (per lo que la clau ha sigut obtinguda), no farà més preguntes.

```
else{

  cursors = this.input.keyboard.createCursorKeys();
  hiParla = xocaSamuel = false;

  tempsResposta = setTimeout(function(){
    informacioMoure.setVisible(true);
    resposta.setVisible(false);
    },2000);
}
```

Conclusions

Aquest joc m'ha portat bastants mals de caps i frustracions, però com tota feina lluitada, al final d'aquest projecte m'he endut un regust dolç a la boca.

Suposo que per la situació que hem estat, el projecte se m'ha fet infinit. Tot i poder consultar al professorat, no és el mateix que estar en la classe, on els dubtes es poden solucionar entre companys o professorat.

Aquest projecte m'ha fet veure la cara que els estudiants que no hem fet encara pràctiques, no veiem/pensem, que és que un cop treballis en una empresa, no hi ha un Samuel o Olga a qui consultar, doncs estàs solament tu davant d'un ordinador, cara a cara, i encara que busquis i no trobis, no podràs consultar els teus dubtes a ningú, per lo que et pots passar hores o dies literalment, i al final de la batalla, no trobar una resposta/solució.

He notat que he pogut desenvolupar més el fet de poder auto aprendre noves tecnologies i mètodes de treball, necessaris per al dia a dia en una empresa.

Crec que en el meu projecte, hi ha hagut 3 moments importants:

- El primer ha sigut saber quin mètode utilitzaria per a fer el joc. Pot semblar fàcil així dit, però per arribar a escollir el mètode que utilitzaria per a treballar, em vaig passar incomptables dies provant diferents maneres de fer jocs, fracàs rere fracàs, fins que vaig trobar un exemple d'animació senzill fet amb Phaser, el qual em va fer escollir-lo per l'estructura que té a l'hora de crear el joc (preload, create i update).

- El segon moment, ha sigut fer que el jugador pugués parlar amb alumnes/professors, el qual vaig invertir moltes hores per a poder-lo fer. El problema requeia en que en Phaser no pots crear (o jo no he sabut trobar-ho) una funció dins de la funció update i modificar valors, per lo qual tot el codi (caminar, claus ...) s'ha de posar dins la funció update la qual sempre s'executa. Això em va fer perdre molt de temps, fins que veient que no podia esperar a més, vaig haver de demanar ajut a l'Olga 2 vegades, doncs no me'n sortia.

- I el tercer moment ha sigut poder guardar la partida e posteriorment obrir-la. En aquest punt vaig haver de rectificar codi important, el qual per desgràcia va fer que no acabés el joc, i perdés la primera convocatòria.