

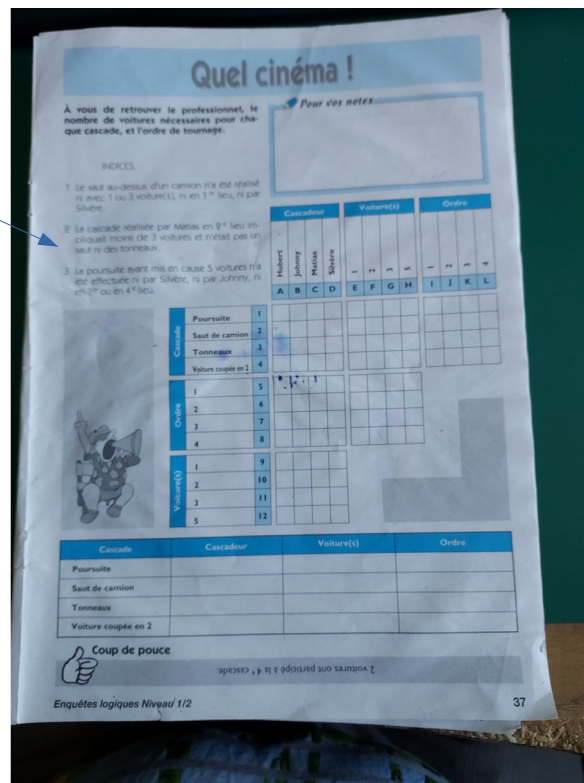
Déroulement de la fabrication du jeu des énigmes  
Y. Stroppa  
PSB 2021  
V1.0

## Introduction :

Ce document décrit les différentes étapes d'élaboration d'une application Web en partant de l'idée du jeu jusqu'à son développement. Il montre une démarche pour aborder ce type de problème et de construire une solution.

Voici la description du jeu.

Une simple photo d'une brochure expliquant l'objectif et le jeu que l'on veut construire



## Etape 1 :

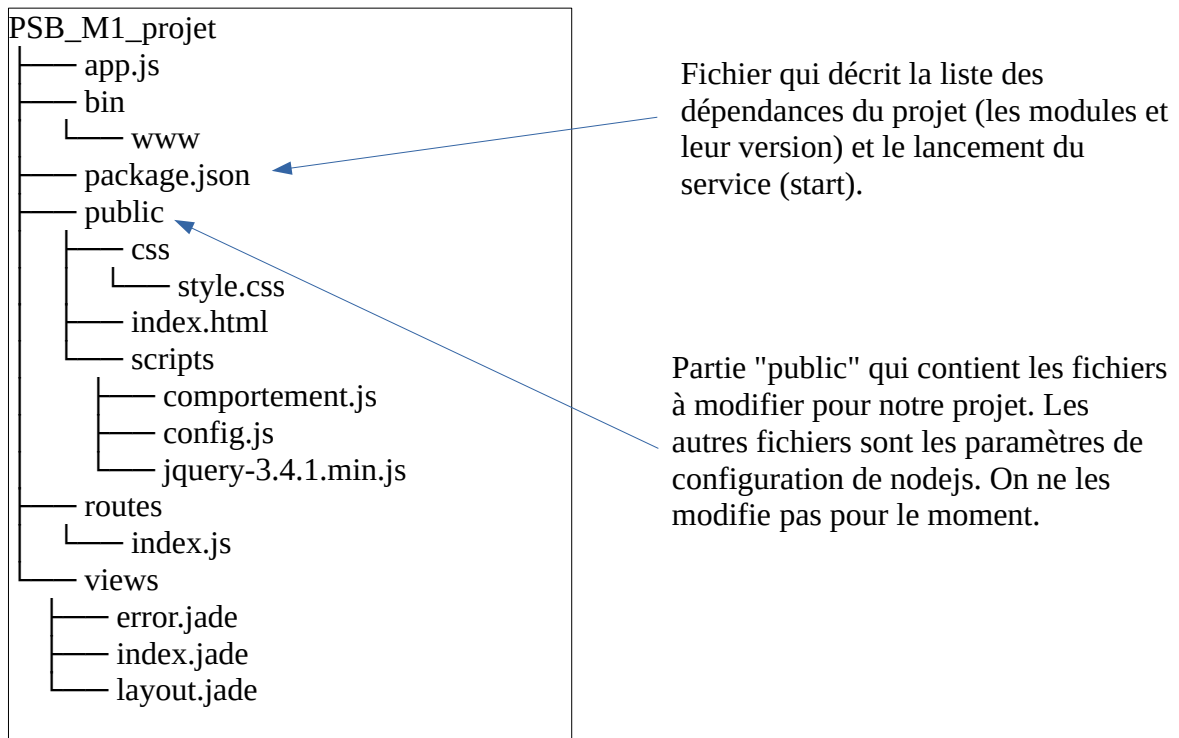
On va reprendre un prototype simple à partir de nodejs d'un squelette de site web à l'adresse suivante : [https://github.com/ystroppa/PSB\\_M1\\_projet](https://github.com/ystroppa/PSB_M1_projet)

Quelques pré-requis : installation de git et de nodejs dans vos environnements.

La commande complète : **git clone https://github.com/ystroppa/PSB\_M1\_projet**

à exécuter sous un terminal de votre système. [powershell sous Windows ou terminal sous MacOS]

A l'aide de git, on télécharge le projet en local (sous le répertoire de votre choix – commandes mkdir et cd pour créer et se déplacer) et on obtient l'arborescence suivante :



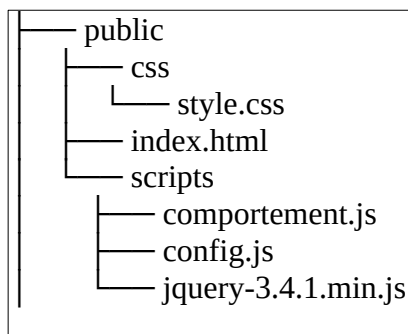
Pour installer le projet il faudra compléter l'installation par les commandes suivantes :

npm install (permet d'installer les modules de l'application)

npm start (permet d'exécuter le point d'entrée du fichier package.json)

Le service web démarre sur le port 4800

La partie sur laquelle il va falloir travailler est la suivante :



On retrouve une arborescence classique de site web que l'on pourra agrémenter en fonction de nos besoins.

Le fichier config.js va nous servir à définir les configurations de jeux ... et le fichier comportement.js va nous servir à définir le chargement de ces configurations et la gestion du jeu.

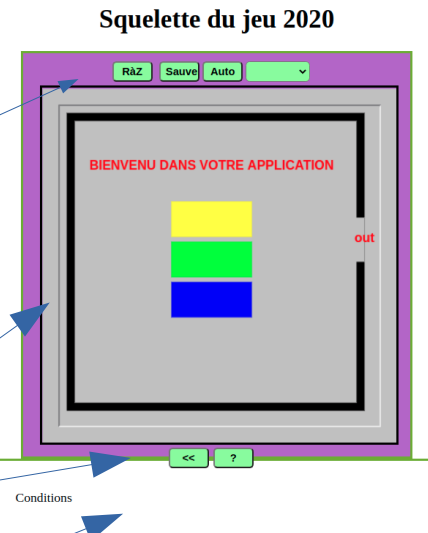
Une fois l'application démarrée on obtient l'affichage suivant : <http://localhost:4800>  
 Le fichier que l'on charge à ce moment est le fichier index.html

```
<!DOCTYPE HTML>
<html>
<head>
<title>Projet YS2020</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" type="text/css" href="/css/style.css" />
<script src="/scripts/jquery-3.4.1.min.js"></script>
</head>
<body>
<center>
<h1>Squelette du jeu 2020</h1>
</center>
<div id="conteneur" class="center">
  <div id="contenu">
    <div id="content">
      <div id="commande_top" class="commandes">
        <input type="button" id="bt3" value="RàZ" onClick="choixRaz();"/>
        <input type="button" id="bt2" value="Sauver" onClick="choixSave();"/>
        <input type="button" id="bt1" value="Auto" onClick="depart(this)"/>
        <select name="config" id="config-select" onChange="chargement_();">
        </select>
      </div>
      <div id="canv">
        <canvas id="canvas" width="440px" height="440px"></canvas>
      </div>
      <div id="commande_bottom" class="commandes">
        <input type="button" id="btReculer" value="<<" onClick="deplace('recule');"/>
        <input type="button" id="btHelp" value="?" onClick="choixRaz();"/>
      </div>
    </div>
  </div>
</div>
<script src="/scripts/config.js"></script>
<script src="/scripts/comportement.js"></script>
<div id="descriptif" class="center_texte">
  <p>
    Conditions
  </p>
</div>

<div id="piedpage">
</div>
</body>
</html>
```

Bandeau de commandes

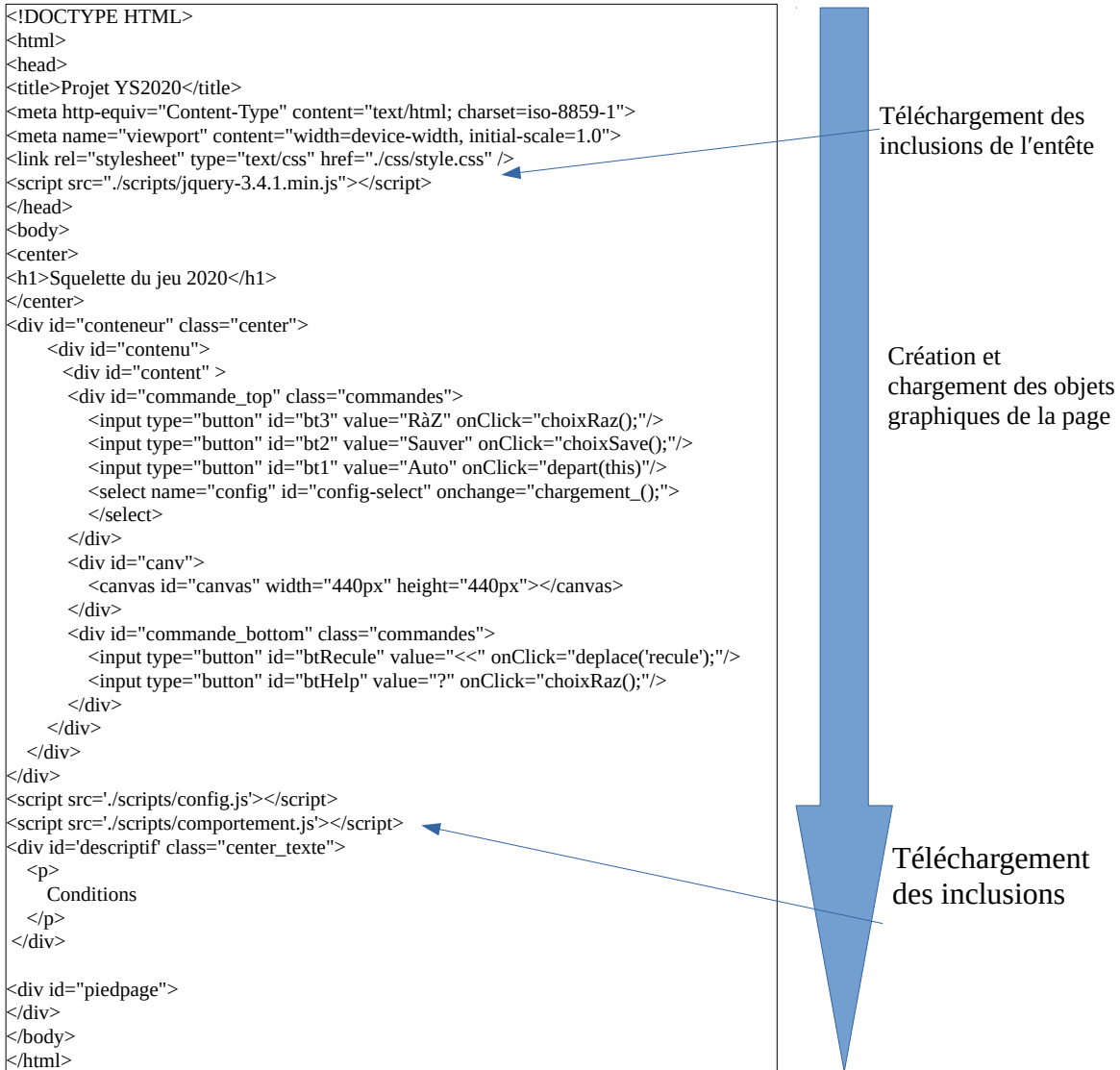
Zone de dessin



Les fonctions javascript (deplace(),choixRaz() ....) seront à définir par la suite, seul le prototype de la fonction (la signature) est défini dans le fichier comportement.js avec rien à l'intérieur.

Dérouler du chargement du fichier html dans le navigateur et interprétation par la navigateur :

Récupère le fichier html en réponse à la requête `http://localhost:4800` et une fois téléchargé, l'interprète



Modifier les fichiers suivants pour augmenter la taille de dessin et d'affichage  
Modification des fichiers html (index.html) et css (styles.css)

```
<div id="conteneur" class="center">
  <div id="contenu">
    <div id="content" >
      <div id="commande_top" class="commandes">
        <input type="button" id="bt3" value="RâZ" onClick="choixRaz();"/>
        <input type="button" id="bt1" value="Auto" onClick="depart(this)"/>
        <select name="config" id="config-select" onChange="chargement_();">
        </select>
      </div>
      <div id="canv">
        <canvas id="canvas" width="1000px" height="1000px"></canvas>
      </div>
      <div id="commande_bottom" class="commandes">
        <input type="button" id="btHelp" value="?" onClick="choixRaz();"/>
      </div>
    </div>
  </div>
</div>
```

On supprime quelques éléments (boutons) et on redimensionne la zone de dessin à 1000px et 1000px pour l'adapter à notre dimension de jeu.

Et pour la partie css

```
.center {
  margin: auto;
  width: 1040px;
  height: 1040px;
  border: 3px solid #73AD21;
  background-color: rgba(125, 10, 170, 0.623);
  padding: 10px 0px;
}

#canv{
  width: 1000px;
  height: 1000px;
  margin-left: auto;
  margin-right: auto;
  margin-bottom: 10px;
}
```

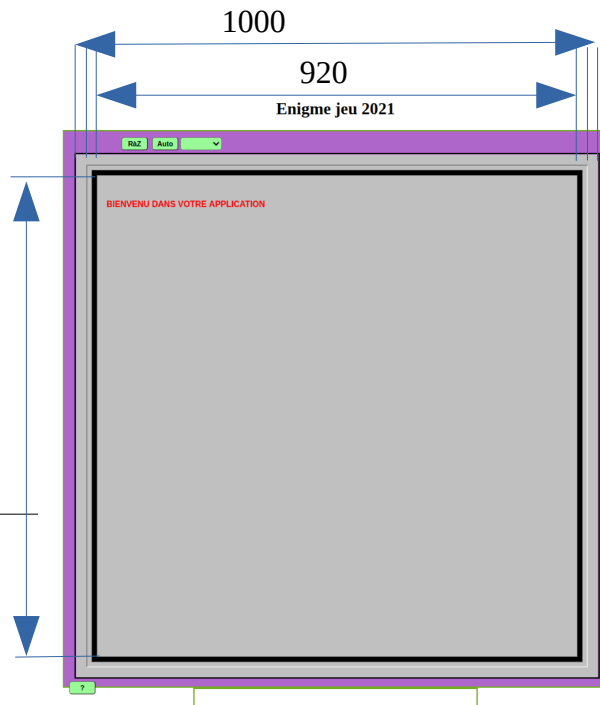
Vous pouvez bien sûr adapter le style (couleur et taille mais attention à reporter vos modifications).

Ensuite modification du fichier comportement.js

```
//il faut bien sur rajouter l'appel à la fonction trace() ;
trace() ;

function affiche(){
// exemple d'affichage d'un texte dans la zone canvas
  hh.fillStyle="red";
  hh.fillText("BIENVENU DANS VOTRE APPLICATION",210,100);
}

function trace(){
  hh.font="bold 16px Arial";
  hh.textAlign="center";
  hh.fillStyle="silver";
  hh.fillRect(0,0,1000,1000);
  hh.fillStyle="black";
  hh.fillRect(Xd+20,Yd+20,940,940);
  hh.fillStyle="silver";
  hh.fillRect(xmin,ymin,920,920);
  hh.fillStyle="silver";
  hh.fillRect(388,160,12,55);
  hh.lineWidth=1;
  cadre3D(20,20,980,980);
  affiche();
}
```



Note :

Pour éviter de fixer les valeurs on aurait pu prendre la dimension de la zone de canvas et définir les dimensions du tracé relative à ces valeurs.

Pour récupérer la dimension du canvas, `document.getElementById("canvas").width` et `height`

Note : d'où vient le hh ????

```
var fenetre = document.getElementById('canvas');
var hh=canvas.getContext('2d');
hh.canvas.style.border="3px solid #000";
hh.translate(0.5,0.5);
```

hh: représente le contexte de l'objet canvas dans lequel on va pouvoir dessiner et utiliser les instructions `fillRect`, `fillText` ....toutes les méthodes et attributs attachées à l'objet hh.

## Analyse des données :

Comment définir une structure de données pour permettre de contenir la description de nos jeux.

De quoi a t-on besoin ????

Après analyse de l'image, que peut-on constater ?

Une partie description de l'objectif du jeu

Une partie indice pour la résolution du jeu

Une partie image

La grille du jeu

Catégories colonnes

Pour chaque catégorie des items

Catégories Lignes

Pour chaque catégorie des items

Et la possibilité dans chaque case de marquer  
d'une croix ou d'un cercle la bonne réponse.

Une grille pour la solution

Une zone coup de pouce

**Quel cinéma !**

À vous de retrouver le professionnel, le nombre de voitures nécessaires pour chaque cascade, et l'ordre de tournage.

*Pour vos notes*

INDICES

- Le saut au-dessus d'un camion n'a été réalisé ni avec 1 ou 3 voitures(s), ni en 1<sup>er</sup> lieu, ni par Silvére.
- La cascade réalisée par Matras en 9<sup>th</sup> lieu implique moins de 3 voitures et n'était pas un saut ni des tonneaux.
- La poursuite avait mis en crise 5 voitures n'a été effectuée ni par Silvére, ni par Johnny, ni en 1<sup>er</sup> ou en 4<sup>th</sup> lieu.

	Cascadeur	Voiture(s)	Ordre
Halbert			
Johnny			
Matras			
Silvére			

	Cascade	Cascadeur	Voiture(s)	Ordre
Poursuite	1			
Saut de camion	2			
Tonneaux	3			
Voiture couplée en 1	4			

	Cascade	Cascadeur	Voiture(s)	Ordre
1	5			
2	6			
3	7			
4	8			

	Cascade	Cascadeur	Voiture(s)	Ordre
1	9			
2	10			
3	11			
5	12			

Cascade	Cascadeur	Voiture(s)	Ordre
Poursuite			
Saut de camion			
Tonneaux			
Voiture couplée en 2			

**Coup de pouce**

2 voitures ont participé à la 4<sup>th</sup> cascade

Enquêtes logiques Niveau 1/2

37

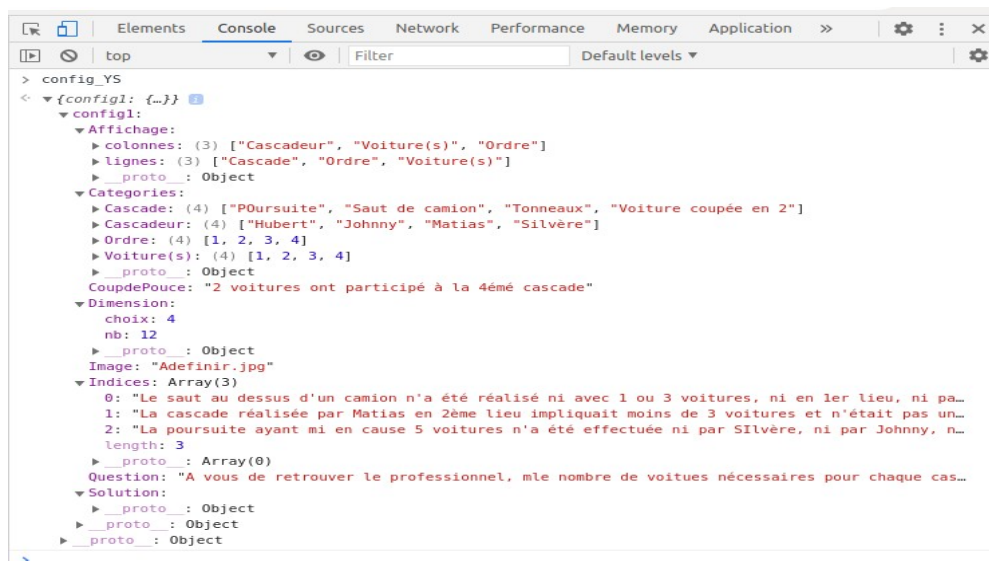
## Création de la structure de données dans le fichier config.js

Nous allons traduire notre structure dans le fichier config.js, chaque configuration de jeu devra avoir un item : Question, Indices (un tableau car plusieurs indices possibles), Image : fichier associé à la configuration du jeu, la dimension (en nombre de colonnes et de choix dans chaque groupe), catégories permettra de définir les différents choix possibles pour la grille, et l'affichage qui permettra de définir comment disposer les éléments dans les entêtes de la grille. (attention aux erreurs de syntaxe fréquentes et au respect de la structure json) si problème non visible vous pouvez utiliser un validateur en ligne de json (<https://jsonformatter.curiousconcept.com/#>)

### Proposition d'une structure

```
var a=58 ;
var config_YS={
  config1:
  {
    Question:"A vous de retrouver le professionnel, le nombre de voitures nécessaires pour chaque cascade et l'ordre de tournage.",
    Indices:[
      "1: Le saut au dessus d'un camion n'a été réalisé ni avec 1 ou 3 voitures, ni en 1er lieu, ni pas Silvère",
      "2: La cascade réalisée par Matias en 2ème lieu impliquait moins de 3 voitures et n'était pas un saut ni des tonneaux.",
      "3: La poursuite ayant mi en cause 5 voitures n'a été effectuée ni par Silvère, ni par Johnny, ni en 1er lieu ou en 4ème lieu."],
    Image:"Adefinir.jpg",
    Dimension:{nb:12,choix:4 },
    Categories:{
      "Cascadeur":["Hubert","Johnny","Matias","Silvère"],
      "Voiture(s)":[1,2,3,4],
      "Ordre":[1,2,3,4],
      "Cascade":["Poursuite","Saut de camion","Tonneaux","Voiture coupée en 2"]
    },
    Affichage:{colonnes:["Cascadeur","Voiture(s)","Ordre"],lignes:["Cascade","Ordre","Voiture(s)"]},
    CoupdePouce:"2 voitures ont participé à la 4ème cascade",
    Solution:{}
  },
}
```

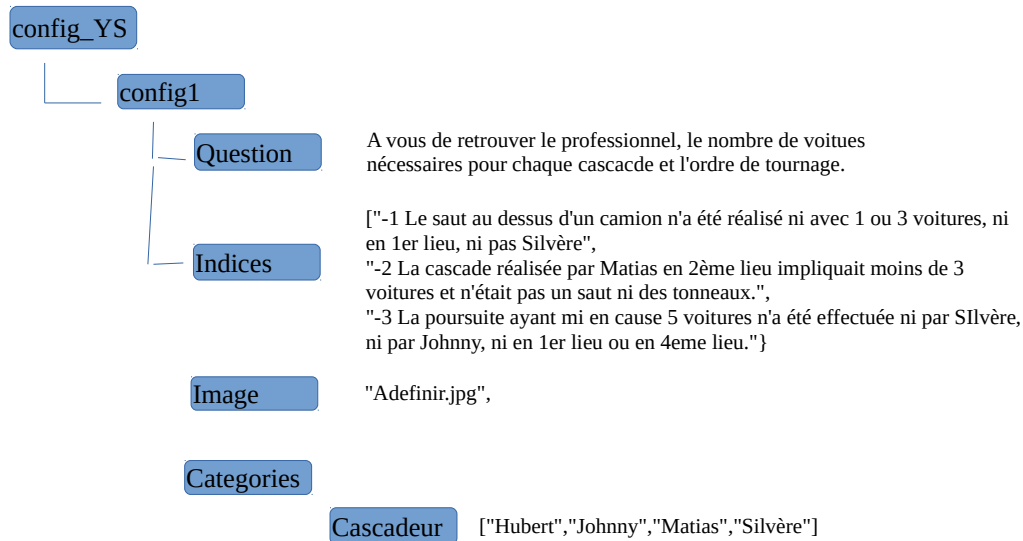
Une fois chargé dans le navigateur on doit pouvoir visualiser le contenu de cet objet...



On pourra essayer de voir comment accéder aux différentes parties de cette variable dans le navigateur  
config\_YS..... ou config\_YS[.....] Peut-on rajouter des éléments à cette structure ???? si oui comment ?



Pour vous aider dans la navigation de ce type de structure, on peut la représentation sous forme d'arbre de la manière suivante. Et pour accéder à une donnée il suffit d'indiquer le chemin pur y accéder ... on est dans une structure de type clé-valeur (key-value)



Pour atteindre les cascadeurs de la config "config1", on utilisera  
`config_YS.config1.Categories.Cascadeur`  
ou  
`config_YS["config1"]["Categories"]["Cascadeur"]`

Attention à déplacer les instructions dans le fichier index.html comme suit pour pouvoir indiquer la question dans la zone de description

```
</div>
<div id='descriptif' class="center_texte">
  <p>
    Conditions
  </p>
</div>
<script src='./scripts/config.js'></script>
<script src='./scripts/comportement.js'></script>
```

Note :  
Erreur classique, car dans le fichier comportement.js nous allons modifier au chargement de celui-ci le contenu de la div (descriptif) si cette div est définie après il y aura une petite erreur d'objet à null.

Définir les déclarations de variables et les fonctions de chargement suivantes :

```
// ajout pour adaptation
var entete_ligne=7;
var entete_colonne=7;
var configuration;

// appel des fonctions d'initialisation
charge_listejeux();
chargement_();

function charge_listejeux() {
  var sele = document.getElementById('config-select');
  for (var jeu in config_YS){
    console.log(jeu) ;
    var option = document.createElement("option");
    option.text = jeu;
    option.value = jeu;
    sele.add(option);
  }
}

function chargement_(){
  var sele = document.getElementById('config-select');
  var config=sele.value;
  indication=config_YS[config].Question;
  // lecture de la configuration
  configuration=config_YS[config];
  choixRaz();
  $("#descriptif > p").html(indication);
  console.log("chargement de la question",indication);
}

// commandes *****
function choixRaz(){
  trace();
}
```

for (var jeu in config\_YS) {}  
Elle permet de balayer l'ensemble des  
clés de la structure config\_YS

On prend la première valeur du select et on vient charger dans la variable globale configuration la config du 1<sup>er</sup> jeu. Ensuite on vient afficher dans la zone descriptif le contenu de l'attribut Question à l'aide de JQuery

En javascript natif :  
var zone=document.getElementById("descriptif")  
zone.innerHTML=indication ;

On aurait pu utiliser les instructions suivantes pour éviter cette dépendance  
Ces instructions permettent d'indiquer les commandes à exécuter lorsque le document est chargé ... à ajouter dans le fichier comportement.js

```
document.addEventListener("DOMContentLoaded", function(event) {
  // appel des fonctions d'initialisation
  charge_listejeux();
  chargement_();
});
```

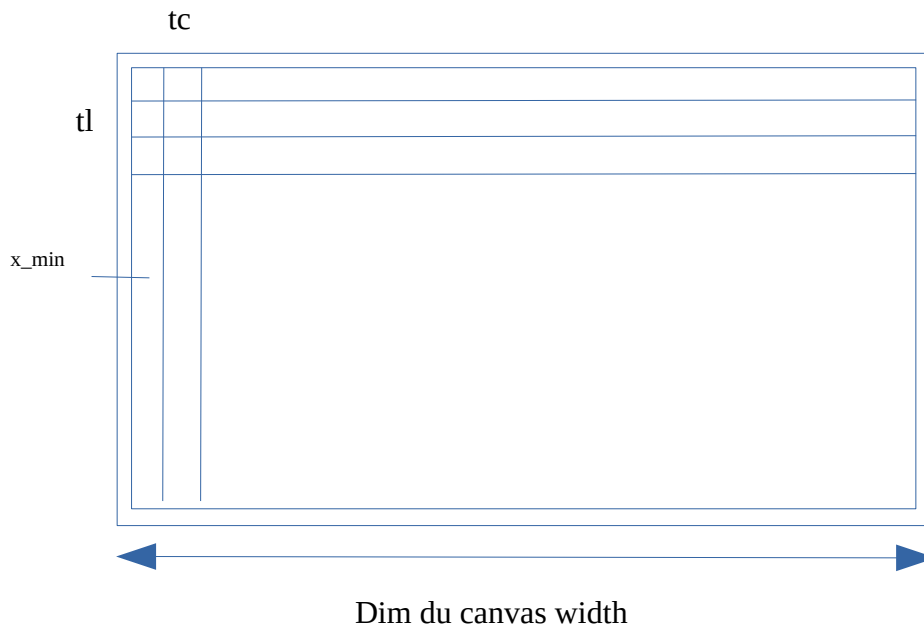
Il faut venir compléter la fonction choix\_raz() qui va nous permettre de définir quelques variables globales telles que tl et tc qui sont respectivement la taille des lignes et des colonnes en fonction de la dimension du canvas .

```
// a rajouter dans l'entete les déclarations des variables
// ajout pour adaptation
var blanc="rgb(220,220,220)",gris="rgb(70,70,70)";
var entete_=7;
var configuration;
var nb_cases; // nb de colonnes == nb ligne c'est un carré
var tl; // hauteur de la ligne
var tc; // largeur de la ligne

// commandes *****
function choixRaz() {
    zone_dessin = document.getElementById('canvas');
    nb_cases=configuration.Dimension.nb+entete_;
    tl=(zone_dessin.height-2*xmin)/(nb_cases+1);
    tc=(zone_dessin.width-2*ymin)/(nb_cases+1);
    trace();
}
```

On part du principe que l'on va dessiner une grille complète de 19 cases dont 7 seront réservées pour les entêtes de ligne et de colonne.

choix\_raz() sera appelé à chaque changement de configuration... pour recalculer les dimensions de la zone de dessin et des cases



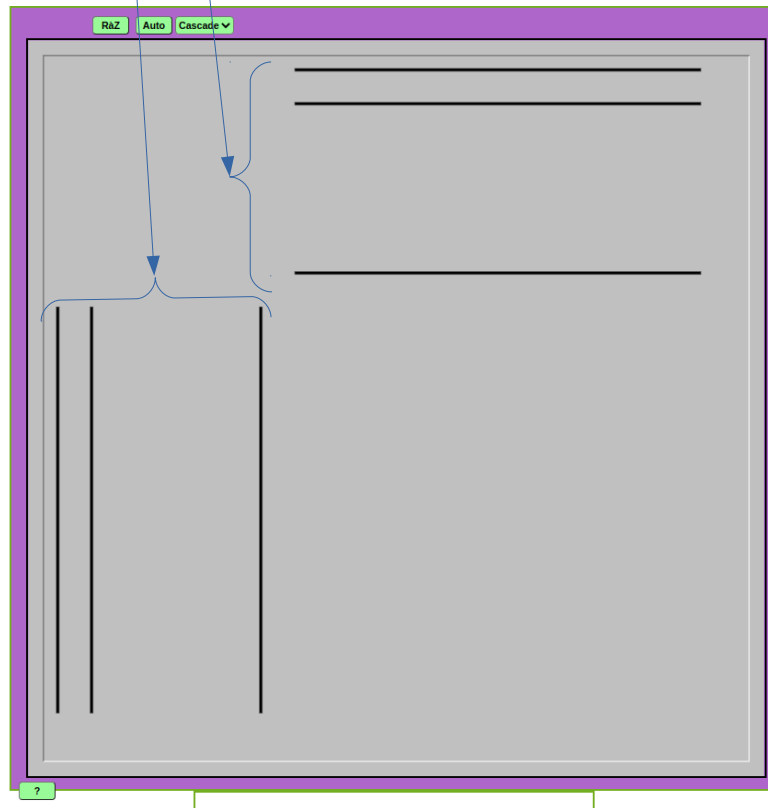
On va s'occuper de la fonction trace pour les différentes constructions du maillage de notre jeu

D'abord on s'occupe des entêtes de lignes et de colonnes pour les séparations.

```
function trace_entetes_lignes() {  
  // trace des entetes de colonnes  
  hh.strokeStyle="black";  
  hh.lineWidth = 4;  
  let y=[0+ymin,1*tl+ymin,(entete_-1)*tl+ymin];  
  for (let yi of y) {  
    drawLine(entete_*tc+xmin,yi,nb_cases*tc+xmin,yi);  
  }  
  let x=[0+xmin,1*tc+xmin,(entete_-1)*tc+xmin];  
  for (let xi of x) {  
    drawLine(xi,entete_*tl+ymin,xi,nb_cases*tl+ymin);  
  }  
}
```

On fixe la couleur et la taille des traits  
On définit deux tableaux y et x qui  
représentent les positions fixes des  
traits à tracer. Et on effectue un  
drawLine

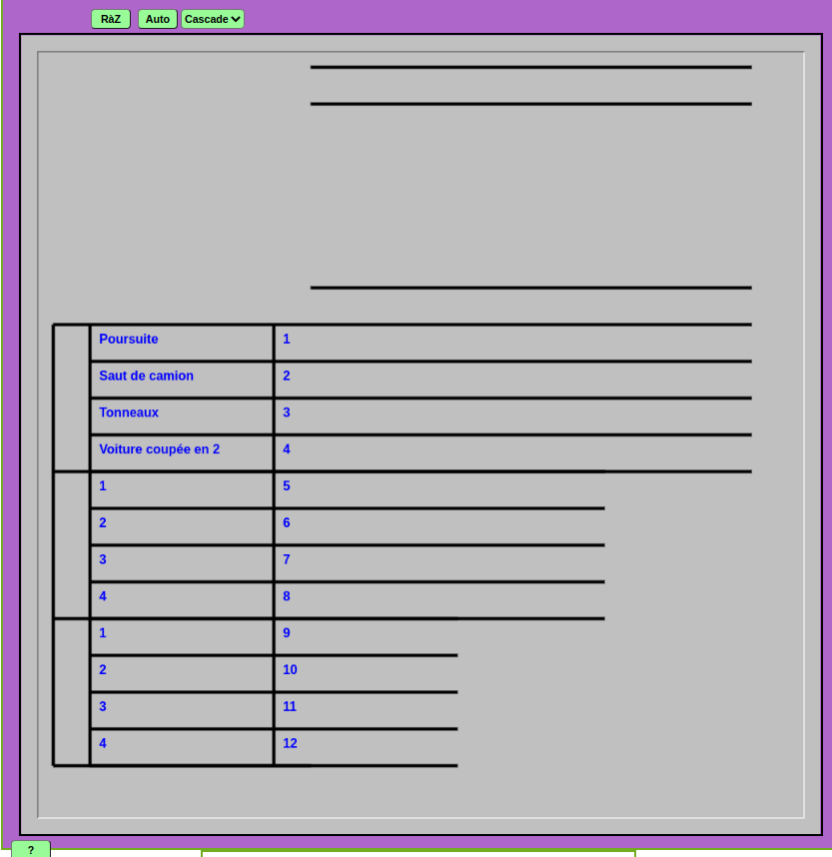
Enigme jeu 2021



On construit ensuite le tracé des autres lignes des catégories

```
function trace_lignes() {
  // trace des entetes de colonnes
  hh.strokeStyle="black";
  hh.lineWidth = 4;
  // tracé des lignes par rubriques
  let fin_ligne=nb_cases*tc+ymin, debut_ligne=0+ymin, ligne_courante=entete_*tl+ymin;
  let numero=1;
  for (let categ of configuration.Affichage.Lignes)
  {
    drawLine(debut_ligne,ligne_courante,fin_ligne,ligne_courante);
    for (let elem of configuration.Categories[categ]) {
      // ecrire l'element
      hh.fillStyle="blue";
      hh.textAlign = "left";
      hh.fillText(elem,debut_ligne+1.25*tc, ligne_courante+0.5*tl);
      hh.fillText(numero,debut_ligne+6.25*tc, ligne_courante+0.5*tl);
      // trace ligne
      ligne_courante+=tl;
      numero+=1;
      drawLine(debut_ligne+1*tl,ligne_courante,fin_ligne,ligne_courante);
    }
    fin_ligne+=configuration.Dimension.choix*tl;
  }
  drawLine(debut_ligne,ligne_courante,fin_ligne,ligne_courante);
}
```

### Enigme jeu 2021

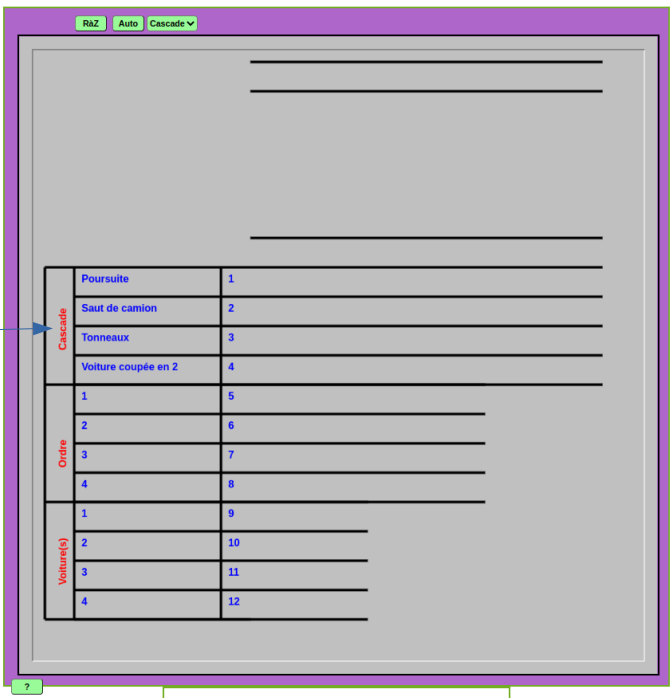


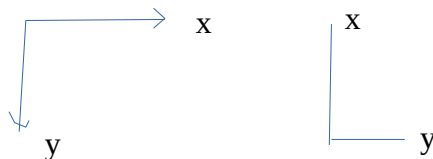
Poursuite	1		
Saut de camion	2		
Tonneaux	3		
Voiture coupée en 2	4		
1	5		
2	6		
3	7		
4	8		
1	9		
2	10		
3	11		
4	12		

On va compléter le code avec les écritures des catégories à la verticale ...

```
function trace_lignes() {
  // trace des entetes de colonnes
  hh.strokeStyle="black";
  hh.lineWidth = 4;
  // tracé des lignes par rubriques
  let fin_ligne=nb_cases*tc+ymin, debut_ligne=0+ymin, ligne_courante=en
  let numero=1;
  for (let categ of configuration.Affichage.Lignes)
  {
    drawLine(debut_ligne,ligne_courante,fin_ligne,ligne_courante);
    hh.fillStyle="red";hh.rotate(-Math.PI/2);
    hh.save();
    hh.translate(debut_ligne+0.5*tc, ligne_courante+5*tl);
    hh.rotate(-Math.PI/2);
    hh.textAlign = "left";
    hh.fillText(categ, 100, 10);
    hh.restore();
    for (let elem of configuration.Categories[categ]) {
      // ecrire l'element
      hh.fillStyle="blue";
      hh.textAlign = "left";
      hh.fillText(elem,debut_ligne+1.25*tc, ligne_courante+0.5*tl);
      hh.fillText(numero,debut_ligne+6.25*tc, ligne_courante+0.5*tl);
      // trace ligne
      ligne_courante+=tl;
      numero+=1;
      drawLine(debut_ligne+1*tl,ligne_courante,fin_ligne,ligne_courante);
    }
    fin_ligne-=configuration.Dimension.choix*tl;
  }
  drawLine(debut_ligne,ligne_courante,fin_ligne,ligne_courante);
}
```

Enigme jeu 2021

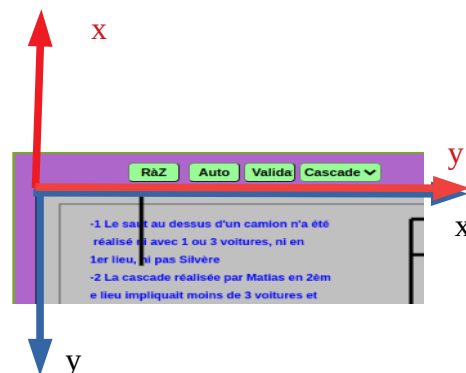




On pourra tester ce changement de repère directement dans le navigateur, en appliquant les instructions suivantes :

```
hh.rotate(-Math.PI/2);
DrawLine(0,100,-100,100)
```

```
> hh.rotate(-Math.PI/2);
< undefined
> drawLine(0,100,-100,100)
< undefined
>
```

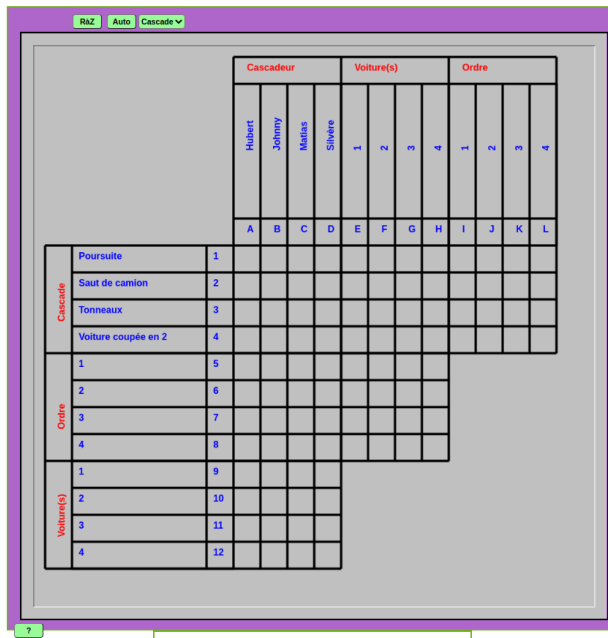


On fait la même chose pour la partie verticale

ce qui nous donne la partie à rajouter à la suite du code de la fonction trace\_ligne

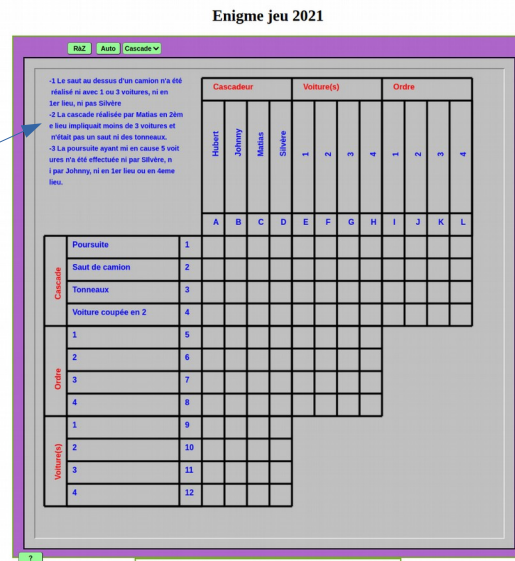
```
let valeur=65;
let etiquette=String.fromCharCode(valeur);
let fin_colonne=nb_cases*tl+xmin, debut_colonne=0+xmin, colonne_courante=entete_*tc+xmin;
for (let categ of configuration.Affichage.Colonnes)
{
  drawLine(colonne_courante,debut_colonne,colonne_courante ,fin_colonne);
  hh.fillStyle="red";
  hh.fillText(categ, colonne_courante+0.5*tc, debut_colonne+0.5*tl);
  for (let elem of configuration.Categories[categ]) {
    // ecrire l'element
    hh.fillStyle="blue";
    hh.save();
    hh.translate(colonne_courante+0.5*tc, 400);
    hh.rotate(-Math.PI/2);
    hh.textAlign = "left";
    hh.fillText(elem, 200, 10);
    hh.restore();
    // trace ligne
    hh.fillText(String.fromCharCode(valeur),colonne_courante+0.5*tc, debut_colonne+6.5*tl);
    valeur+=1;
    colonne_courante+=tl;
    drawLine(colonne_courante,debut_colonne+1*tc,colonne_courante,fin_colonne);
  }
  fin_colonne-=configuration.Dimension.choix*tc;
}
drawLine(colonne_courante,debut_colonne,colonne_courante ,fin_colonne);
```

Enigme jeu 2021



IL faut ajouter le texte avec les indices pour chaque configuration :

```
function ecriture_indices() {
  hh.fillStyle="blue";
  hh.font="bold 14px Arial";
  var ligne=0;
  let nb_caract=40;
  for (let ind of configuration.Indices) {
    // il faut découper chaque ligne de 30 caractères
    let nb_parties=ind.length/nb_caract;
    let debut=0;
    for (let i=0; i<nb_parties; i++) {
      hh.fillText(ind.substr(debut,nb_caract), 50, 50+ligne);
      ligne+=tl*.5;
      debut+=nb_caract;
    }
  }
}
```



On va pouvoir tester avec une deuxième configuration

```
Attractions: {
  Question:"Frissons garantis pour Ludivine et Frédéric qui on passé une journée au parx EffroiLand ...",
  Indices:[
    "-1 La maison du diable n'a pas été visitée apers 50min de file, ni en 2min30, ni en 3e lieu",
    "-2 L'attente au Miroir aux alouettes a été deux fois plus longue qu'à la première attraction, visitée en 2min",
    "-3 Les mines de Salomon, visitées en 1min30 après moins de 50min d'attente, 'étaient ni la 3e ni la 5e attraction.",
    "-4 Pour la 4e attraction, visitée en une minute de plus que le Tunnelde l'épouvante, Ludivine et Frédéric on tattendu 40min."],
  Image:"Adefinir.jpg",
  Dimension:{nb:15,choix:5 },
  Categories:{
    Duree:["1 min","1 min 30","2 min","2 min 30","3 min"],
    Ordre:[1,2,3,4,5],
    Attente: ["20 min","30 min","40 min","50 min","60 min"],
    Attraction:["La maison du diable","Les mines de salomon","Le miroir aux alouettes","Le temple de Sésostris","Le tunnel de l'épuouvante"]
  },
  Affichage:{Colonnes:["Duree","Ordre","Attente"],Lignes:["Attraction","Attente","Ordre"]},
  CoupdePouce:"",
  Solution:{}
}
```



Ce qui nous donne

Enigme jeu 2021

RàZAutoAttraction

-1 La maison du diable n'a pas été visitée après 50min de file, ni en 2min30, ni en 3e lieu

-2 L'attente au Miroir aux alouettes a été deux fois plus longue qu'à la première attraction, visitée en 2min

-3 Les mines de Salomon, visitées en 1min30 après moins de 50min d'attente, n'ont été visitées ni la 3e ni la 5e attraction.

-4 Pour la 4e attraction, visitée en une minute de plus que le Tunnel de l'épouvante, Ludvine et Frédéric ont attendu 40 min.

		Duree					Ordre					Attente				
		1 min	1 min 30	2 min	2 min 30	3 min	1	2	3	4	5	20 min	30 min	40 min	50 min	60 min
		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Attraction	La maison du diable	1														
	Les mines de salomon	2														
	Le miroir aux alouettes	3														
	Le temple de Sésostris	4														
	Le tunnel de l'épouvante	5														
Attente	20 min	6														
	30 min	7														
	40 min	8														
	50 min	9														
	60 min	10														
Ordre	1	11														
	2	12														
	3	13														
	4	14														
	5	15														

?

## La gestion du jeu et de l'interaction avec l'utilisateur

On va ajouter à la variable fenetre positionnée sur l'objet Canvas un module d'écoute .

```
fenetre.addEventListener('click', function(evt) {
    var x = evt.pageX;
    var y = evt.pageY;
    var node = evt.target;
    while (node) {
        x -= node.offsetLeft - node.scrollLeft;
        y -= node.offsetTop - node.scrollTop;
        node = node.offsetParent;
    }
    Xs=x-3;      Ys=y-3;
    I=1+Math.floor((Xs-xmin)/tc);
    J=Math.floor((Ys-ymin)/tl);
    console.log(X,Y);
});
```

Comment faire pour permettre la modification des cases en fonction du choix de l'utilisateur ?

On va construire un objet qui va représenter le maillage de notre jeu et chaque case sera un élément de ce maillage et aura 3 états lorsque la case est dans la partie modifiable du jeu ....

0 : pas de marquer

1 : un X pour indiquer que la cellule n'est pas retenue

2 : pour indiquer que l'on retient la cellule

sinon la case aura la valeur à -1.C'est une convention, on aura pu faire autrement ...

On ajoute l'initialisation de cette nouvelle variable pour ne remplir que les cases modifiables

```
function choixRaz() {
    maillage=[];

    zone_dessin = document.getElementById('canvas');
    nb_cases=configuration.Dimension.nb+entete_;
    let choix = configuration.Dimension.choix;
    for (let i=0;i<nb_cases;i++){
        maillage[i]=[];
    }
    tl=(zone_dessin.height-2*xmin)/(nb_cases+1);
    tc=(zone_dessin.width-2*ymin)/(nb_cases+1);
    let fin_ligne=nb_cases;
    for (let i=0;i<nb_cases;i++){
        if (i>7)
            if ((i-7)%choix==0 && i!=0){
                fin_ligne-=choix;
            }
        for (let j=0;j<nb_cases;j++){
            maillage[i][j]=0;
            if (i<7) maillage[i][j]=-1;
            if (j<7) maillage[i][j]=-1;
            if (j>=fin_ligne) maillage[i][j]=-1;
        }
    }
    trace();
}
```

Pour prendre en compte la structure particulière du jeu.

On va ajouter une nouvelle fonction pour le dessin des cases et la relier à l'événement click sur le jeu

....

Ce qui nous donne

```
function redessine_maillage() {
  hh.font="bold 20px Arial";
  for (let i=0;i<nb_cases;i++){
    for (let j=0;j<nb_cases;j++){
      let posit_x=(i)*tc+xmin;
      let posit_y=(j+1)*tl+ymin;
      switch (maillage[i][j]) {
        case 0:
          hh.fillStyle="silver";
          hh.fillRect(posit_x+5,posit_y+5-tl,tc-10,tl-10);
          hh.fillStyle="blue";
          hh.fillText("",posit_x+tc/4,posit_y-tl/4);
          break;
        case 1:
          hh.fillStyle="silver";
          hh.fillRect(posit_x+5,posit_y-tl+5,tc-10,tl-10);
          hh.fillStyle="red";
          hh.fillText("X",posit_x+tc/4,posit_y-tl/4);
          break;
        case 2:
          hh.fillStyle="silver";
          hh.fillRect(posit_x+5,posit_y+5-tl,tc-10,tl-10);
          hh.fillStyle="blue";
          hh.fillText("O",posit_x+tc/4,posit_y-tl/4);
          break;
      }
    }
  }
}
```

Et le lien avec la gestion d'événements

```
// 1 evenementiel
fenetre.addEventListener('click', function(evt) {
  var x = evt.pageX;
  var y = evt.pageY;
  var node = evt.target;
  while (node) {
    x -= node.offsetLeft - node.scrollLeft;
    y -= node.offsetTop - node.scrollTop;
    node = node.offsetParent;
  }
  Xs=x-3; Ys=y-3;
  I=Math.floor((Xs-xmin)/tc);
  J=Math.floor((Ys-ymin)/tl);
  if (maillage[I][J]!=-1) {
    // il faut changer l'état du maillage et redessiner
    console.log(maillage[I][J]);
    switch (maillage[I][J]) {
      case 0:
        maillage[I][J]=1;
        break;
      case 1:
        maillage[I][J]=2;
        break;
      case 2:
        maillage[I][J]=0;
        break;
    }
    redessine_maillage();
  }
});
```

Pour aller plus loin :

Veuillez charger les configurations de jeux suivantes

On pourra ajouter également la gestion des solutions .... sous quelle forme peut-on définir les solutions dans la partie configuration .... et comment éviter que la solution soit ramener du côté du client dans la fichier config.js, peut-on imaginer que la solution au problème se trouve du côté du serveur ou du côté d'un cache mais que l'accès soit conditionné .....

Egalement dans la saisie des résultats, comment ouvrir une boite de dialogue qui permet à l'utilisateur de saisir sa réponse ..... et de la valider auprès d'un service de vérification ... .

Mise en place de la validation de la solution ...

Pour effectuer la vérification de la solution, il faudrait pour chaque configuration de jeu la mémoriser dans la structure config\_YS dans l'attribut Solution de la manière suivante :

```
Solution:[[-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1],
          [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1],
          [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1],
          [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1],
          [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1],
          [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1],
          [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1],
          [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1],
          [-1, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
          [-1, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
          [-1, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
          [-1, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
          [-1, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, -1],
          [-1, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, -1],
          [-1, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, -1],
          [-1, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, -1],
          [-1, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, -1, -1, -1, -1, -1, -1, -1],
          [-1, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, -1, -1, -1, -1, -1, -1, -1],
          [-1, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, -1, -1, -1, -1, -1, -1, -1],
          [-1, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, -1, -1, -1, -1, -1, -1, -1]]
```

Ensuite, il faut bien sûr enregistrer la bonne solution et compléter le HTML en ajoutant un nouveau bouton Validation auquel on va associer la fonction validation() qui nous permettra de comparer le tableau maillage à la valeur de l'attribut Solution de la partie en cours ...pour ce faire nous allons avoir besoin d'une fonction qui va nous permettre de comparer deux tableaux ...

```

function array_compare(a1, a2) {
    if(a1.length != a2.length) {
        return false;
    }
    for(var i in a1) {
        if(a1[i] instanceof Array && a2[i] instanceof Array) {
            if(!array_compare(a1[i], a2[i])) {
                return false;
            }
        }
        else if(a1[i] != a2[i]) {
            return false;
        }
    }
    return true;
}

```

```

function validation() {
    // lire la solution de la partie config_YS
    // extraire les cases sélectionnées par ligne et par colonnes a partir de la
    // variable matrice
    // comparer
    // si identique : bonne solution
    // dans la case contraire ce n'est pas la bonne solution
    if (array_compare(maillage,config_YS.Attractions.Solution)) {
        alert("bonne solution");
    }
}

```