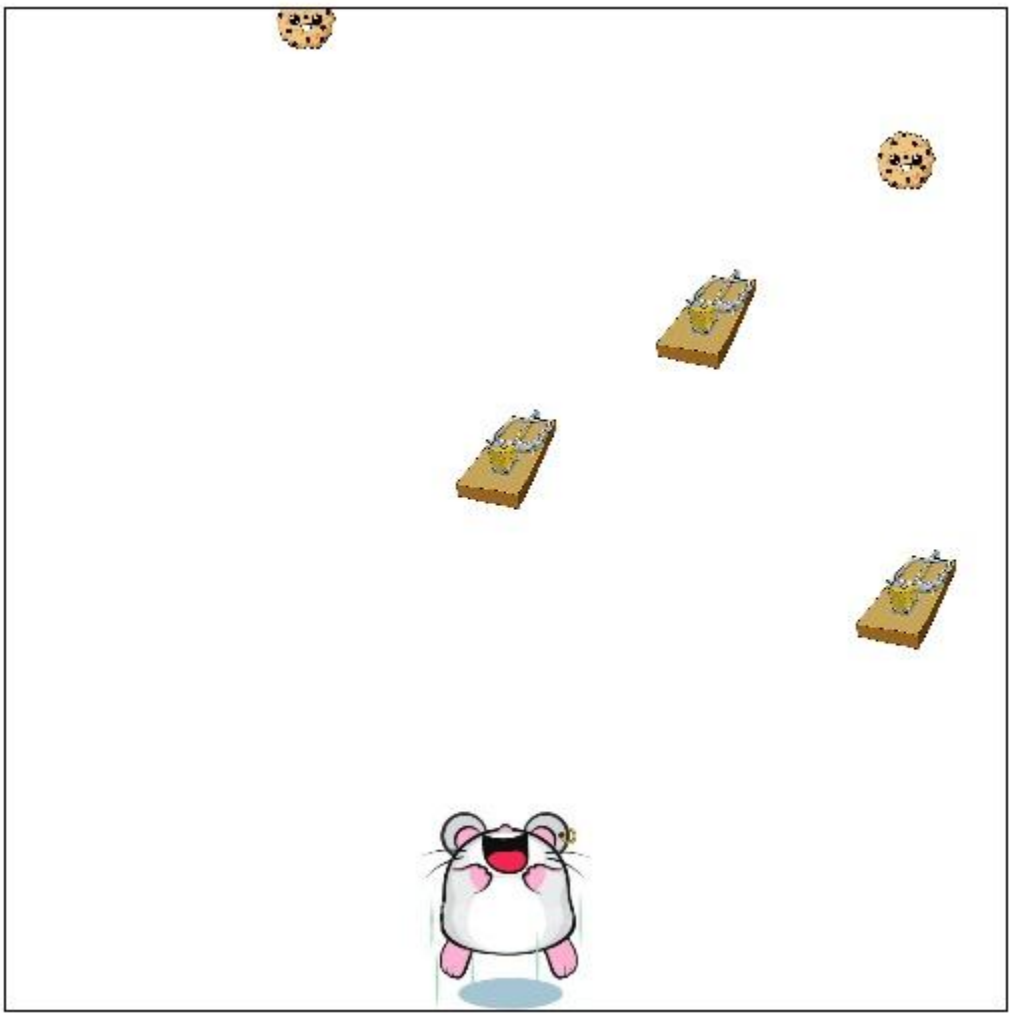
PROJET WEB – HAMSTER



RESUME

Le projet qui nous a été donné de faire, en utilisant HTML, CSS , Javascript ainsi que les Cookies est un mini-jeu dans lequel nous contrôlons un hamster qui est entouré d'un cadre dont il ne peut pas sortir. Ne pouvant se déplacer que latéralement, le but est d'éviter les pièges apparaissant aléatoirement et de rammasser le plus de cookies possible, permettant de monter un score.

CHOIX PERSONNELS

CANVAS

Nous avons choisi d'utiliser le canvas, il s'agit d'un espace de pixels initialement transparents, armés de JavaScript pour réaliser un bon nombre de fonctions graphiques. Il y a de nombreux avantages à l'utilisation, il est performant et accéléré matériellement sur la plupart des navigateurs et systèmes.

Il est aussi bien plus pratique pour gérer tous les objets crées dans le mini jeu afin de les supprimer lorsqu'ils sortent de la zone de jeu par exemple. Et ça nous permet d'appréhender quelque chose qui est obligatoire pour créer un jeu web plus poussé car il est destiné à terme à remplacer flash car mieux pris en charge sur les mobiles.

LA SUPPRESSION DES OBJETS (COOKIES, GROS COOKIES, PIEGES)

Nous avons choisis de supprimer ces éléments avant qu'ils ne descendent complètement en bas de la zone de jeu, au niveau supérieur du hamster. Cela nous permet d'éviter a avoir à gérer une collision sur l'axe des abscisses.

TAUX D'APPARITION DES GROS COOKIES

Nous avons choisis de faire un taux d'apparition de 4% pour les gros cookies, soit 48% pour les cookies normaux ainsi que les pièges. Ce nombre est assez faible pour ne pas avoir une apparition trop fréquente et ainsi augmenter la difficulté.

CONCEPTION DE L'INTERFACE GRAPHIQUE

Le menu de navigation entre les différentes pages du site web a été fait dans le but de rester simple et sobre. De façon à garder une visibilité sur les différents contenu de la page tels que le canvas, les règles etc . Les boutons une fois survolés peuvent avec le cadre noir, faire penser à la bordure de la zone de jeu.

1. Un hamster est contrôlé par le joueur à partir du clavier et se décale vers la gauche ou vers la droite (touches flèche gauche et droite du clavier).

2. Des cookies et des pièges sont générés aléatoirement, apparaissent en haut de la zone du jeu et bougent du haut vers le bas.

3. Le hamster doit manger les cookies et éviter de toucher les pièges.

Ci-dessus, un exemple de l'affichage des règles.

Les règles se trouvent sur une autre page du site web, dans l'onglet de navigation « règles », la police utilisée est assez claire et ne surcharge pas la page, nous avons utilisés des couleurs différentes pour souligner les objets se trouvant dans le jeu tels que le hamster (en vert) , les cookies (en brun) et les pièges (en rouge)

DOCUMENTATION DU CODE SOURCE

HTML

```
<html>
    <head>
        <meta name="author" content="Rémy Rodolphe Arnaud"charset="utf-8">
        <link rel="stylesheet" type="text/css" href="style.css"/>
        <link href='http://fonts.googleapis.com/css?family=Raleway:200,400,800'
rel='stylesheet' type='text/css'>
        <title> jeu hamster </title>
    </head>
    <body>
        <nav class="menu">
                <a  href="index.html">Accueil</a>
                <a class="page" href="hamster.html">Jeu</a>
                <a  href="regles.html">Règles</a>
                <a  href="rapport.html">Rapport</a>
        </nav>
        <div id="vie"></div>
        <div id="score"></div>
        <div id="temps"></div>
        <div id="piege"></div>
        <canvas id="canvas"></canvas>
        <input class="btn"type="button" value="commencer" onclick=commencer(),chrono()>
</input>
        </div>
        <div id="temps"></div>

        <script type="text/javascript" src="canvas.js"></script>
    </body>
</html>
```

```
<nav class="menu">
        <a  href="index.html">Accueil</a>
        <a class="page" href="hamster.html">Jeu</a>
        <a  href="regles.html">Règles</a>
        <a  href="rapport.html">Rapport</a>
</nav>
```

La balise nav permet de faire un menu de navigation comportant les 4 pages suivantes : Accueil, Jeu , Règles , Rapport. On peut remarquer que seule la page jeu a la classe « page », cela permet au bouton de navigation « page » de rester comme s'il était sous le curseur, on sait ainsi dans quelle page on se trouve

```
        <div id="vie"></div>
        <div id="score"></div>
        <div id="temps"></div>
        <div id="piege"></div>
```

Permet de récupérer les informations de vie, score, temps et pièges évités sur la page de jeu

```
         <input class="btn"type="button" value="commencer" onclick=commencer(),chrono()>
</input>
        </div>
```

Le bouton commencer permet de relancer le jeu ainsi que le chronomètre une fois que l'on a été en contact avec un piège

JAVASCRIPT

```
function update(){
```

La fonction update() est le cœur du jeu, en effet c'est ici que l'on met à jour la zone de jeu.

Tout d'abord, on remplit le canvas de pixels vides.

```
context.fillRect(0, 0, canvas.width, canvas.height);
```

Ensuite on crée un objet aléatoirement à l'aide de la méthode Math.random, selon le nombre généré on obtient donc soit un cookie, soit un piège ou soit un gros cookie.

```
if (nbre % modulo==0){

    var newObjet = new Object();
    var aleatoire = Math.random();

    if (aleatoire < 0.48){
        newObjet.x = (Math.floor (Math.random() *5)*100+35);
        newObjet.cookieOuPiege = 0;
        newObjet.taille=tailleCookie;
    }

    if (aleatoire >= 0.48 && aleatoire < 0.96) {
        newObjet.x = (Math.floor (Math.random() *5)*100+25);
        newObjet.cookieOuPiege = 1;
        newObjet.taille=taillePiege;
    };

    if (aleatoire >= 0.96){
        newObjet.x = (Math.floor (Math.random() *5)*100+15);
        newObjet.cookieOuPiege = 2;
        newObjet.taille=tailleGrosCookie;
    }
```

On ajoute ensuite le nouvel objet crée en fin du tableau CookiesArray.

```
    for (var i=CookiesArray.length - 1; i >= 0; i--){

        CookiesArray [i].y++;

        if (CookiesArray[i].y + CookiesArray[i].taille > canvas.height - tailleHamster +
5){
            if (CookiesArray[i].x+CookiesArray[i].taille/2==x+tailleHamster/2){
                collision(CookiesArray[i]);
            }
```

Si la taille de l'objet figurant dans le tableau CookiesArray à la position i divisé par deux + la coordonnée x de ce même objet est égale à la coordonnée x du hamster + sa taille divisée par deux alors on a collision et on appelle la fonction collision.

```
function collision(objet){
```

La fonction collision permet de vérifier avec quel objet le hamster est rentré en collision. Si l'on entre dans un cookie, on incrémente le score de 1 . Si l'on entre dans un piège, on décrémente la vie de 1

```
if (objet.cookieOuPiege == 0){
    score++;
    alert2.innerHTML = "SCORE : "+score;
    var audio = new Audio('cookie.mp3');
    audio.play();
}
if (objet.cookieOuPiege == 1){
    var audio2 = new Audio('tapette.mp3');
    audio2.play();
    vie--;
    alert1.innerHTML = "VIES : "+vie;
    clearTimeout(compte);

    if (vie == 0){
```

Si le nombre de points de vie passe à 0, on regarde si le score du joueur est supérieur au score sauvegardé, si c'est le cas on remplace l'ancien par le nouveau et on réinitialise le jeu et les variables telles que le score, les pièges évités etc. Ceci est rendu possible grâce aux fonctions setCookie et getCookie fourni par notre enseignante.

```
function setCookie(cname,cvalue,exdays) {
    var d = new Date();
    d.setTime(d.getTime() + (exdays*24*60*60*1000));
    var expires = "expires=" + d.toGMTString();
    document.cookie = cname+"="+cvalue+"; "+expires;
}

function getCookie(cname) {
    var name = cname + "=";
    var ca = document.cookie.split(';');
    for(var i=0; i<ca.length; i++) {
           var c = ca[i];
           while (c.charAt(0)==' ') c = c.substring(1);
                if (c.indexOf(name) != -1) {
                    return c.substring(name.length, c.length);
                }
            }
    return "";
}
```

```
function deplacerHamster(e){
```

Cette fonction, comme son nom l'indique permet de déplacer le hamster soit vers la droite gauche, soit vers la droite à l'aide des flèches gauche et droite.

```
switch (e.keyCode) {

    case 37:

    if (x - mouvement > -tailleHamster){
        clear();
        x = x-mouvement;
        placerHamster(x,y,tailleHamster,tailleHamster);
    }

    break;

    case 39:

    if (x + mouvement < widthCanvas){
        clear();
        x = x+mouvement;
        placerHamster(x,y,tailleHamster,tailleHamster);
    }
    break;
}
```

CHANGEMENT DE STYLE

A chaque fois qu'il y a collision avec un objet en fonction de sa nature (cookie ou piège) il y a changement de style. Ainsi le nombre de vie devient rouge au contact d'un piège et vert au contact d'un petit cookie.

```
alert1.style.color="#383a3c";
alert2.style.color="green";
```

CONCLUSION

Nous avons eu des difficultés pour la mise en place des cookies et donc la sauvegarde des scores notamment sur les navigateurs Google Chrome et Opera car il y a une sécurité pour ne pas utiliser les cookies de façon locale.

Au niveau des améliorations possibles on pourrait essayer d'enlever tout clignotement mais aussi rajouter un système de niveaux pour augmenter la difficulté au fur et à mesure de la partie, limiter le nombre de points de vie maximum. Et même rajouter un classement en ligne des meilleurs joueurs.