Course aux objets

DIRECTIVES ET INFORMATIONS RELATIVES AU TRAVAIL

- Ce travail compte pour 30% de la note finale.
- Ce travail se fait individuellement ou en équipe de deux (2) ou trois (3) personnes. Chaque membre de l'équipe doit contribuer et être en mesure d'expliquer l'ensemble du travail.
- Chaque équipe doit réaliser son propre concept original tout en respectant les exigences du travail.
- La remise du dossier de projet (format zip) se fait sur Léa avant la date butoir spécifiée sur Léa.
- Attention au plagiat : il est interdit de partager du code (HTML, CSS ou JavaScript) ou des ressources (ex. : sons ou images) avec d'autres équipes.
- Si vous récupérez du code d'un site Web ou de ChatGPT (ou équivalent), vous devez le comprendre et citer clairement la source dans un document Word qui sera remis avec le projet. Inclure des saisies d'écran pour chaque source. Dans le cas de ChatGPT (s'il y a lieu), les saisies d'écran doivent également comporter les questions.

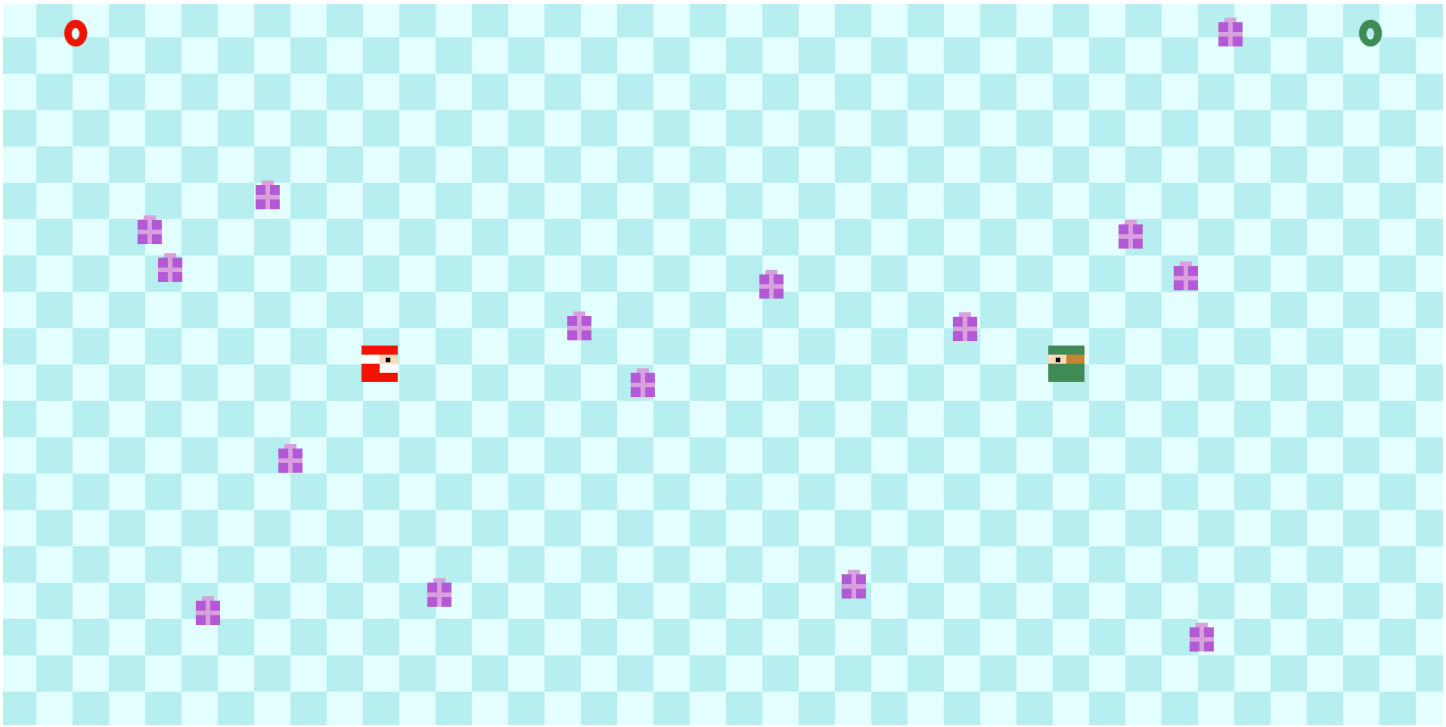
OBJECTIF

Ce travail pratique évalue l'élément de compétence « Programmer l'interactivité des interfaces » en vérifiant l'application des notions suivantes :

- Structurer un site Web dans une arborescence;
- Personnaliser les éléments d'un site Web avec les propriétés CSS;
- Intégrer des polices de caractères, des images et des sons dans un site Web;
- Utiliser des techniques de mise en page (« layout »);
- Écrire des expressions et des fonctions en JavaScript;
- Prendre en charge des événements du navigateur avec JavaScript;
- Exploiter le canevas HTML5 en JavaScript;
- Manipuler le DOM avec JavaScript.

SITE WEB À RÉALISER – ORGANISATION

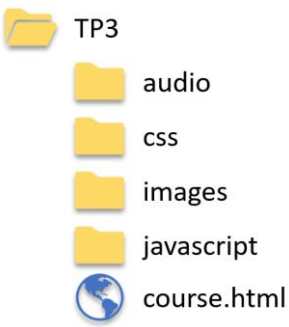
Le site Web à réaliser comporte une (1) page Web HTML5 constituée d'un canevas plein écran et une application frontale JavaScript qui affiche dans le canevas et répond aux événements du navigateur. Cette application interactive est un jeu multi-joueurs.



Le fonctionnement du jeu est tout simple :

> Chaque joueur contrôle un personnage qui récupère des objets affichés aléatoirement à l'écran. Le joueur dont le personnage ramasse le plus grand nombre d'objets gagne la partie.

Arborescence



L'arborescence du site doit prévoir un emplacement pour chaque type d'élément. Par exemple, les fichiers CSS doivent se retrouver dans un dossier prévu à cette fin, même chose pour les fichiers JavaScript, les images (s'il y a lieu) et les fichiers audio.

Toutes les feuilles de style (CSS) et les scripts (code JavaScript) doivent être externes au code HTML, c'est-à-dire contenus dans des fichiers autonomes situés dans les bons dossiers.

APPLICATION À RÉALISER – SPÉCIFICATIONS

Spécifications de la page Web

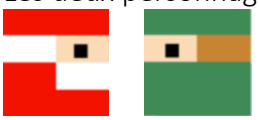
1. Le canevas doit occuper toute la fenêtre du navigateur (plein écran).
2. La page doit récupérer au moins une police de caractères Google qui sera utilisée dans le canevas.
3. Le code JavaScript doit être dans un fichier distinct (pas dans le fichier HTML).

Spécifications du jeu codé en JavaScript

1. Le texte, les images et les sons doivent être respectueux, de bon goût, socialement acceptables.

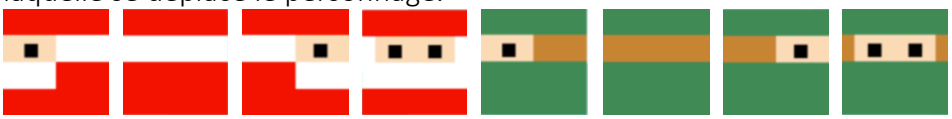
2. Le fond d'écran (un plancher de tuiles) doit être généré par code JavaScript.
3. Le plancher doit comporter au moins deux tuiles différentes.

4. Les deux personnages doivent avoir une apparence différente.



5. Le personnage de gauche doit être déplacé avec les touches « W », « A », « S » et « D ».
6. Le personnage de droite doit être déplacé avec les touches « flèche vers le haut », « flèche vers la gauche », « flèche vers le bas » et « flèche vers la droite ».
7. Les deux personnages doivent se déplacer à la même vitesse.
8. Les déplacements en diagonale doivent être permis.
9. Les personnages ne doivent pas être en mesure de sortir de l'écran.

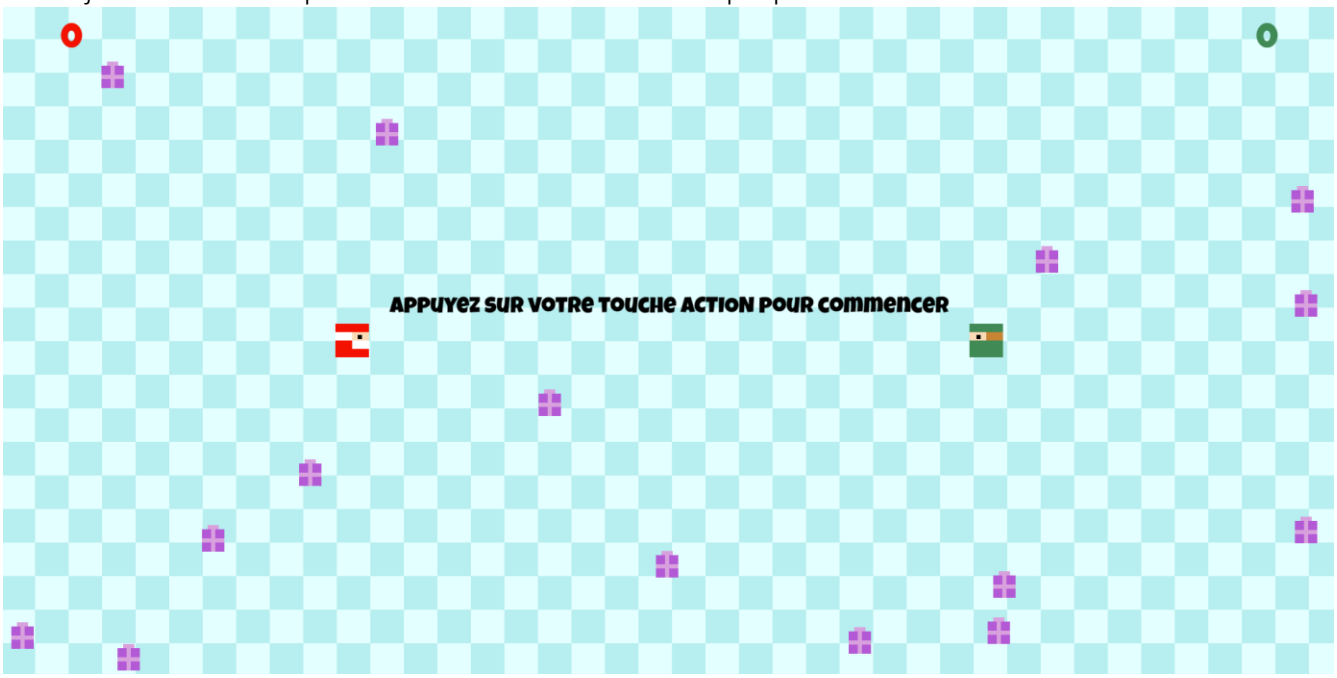
10. Chaque personnage doit avoir 4 images différentes. Chaque image doit correspondre à la direction vers laquelle se déplace le personnage.

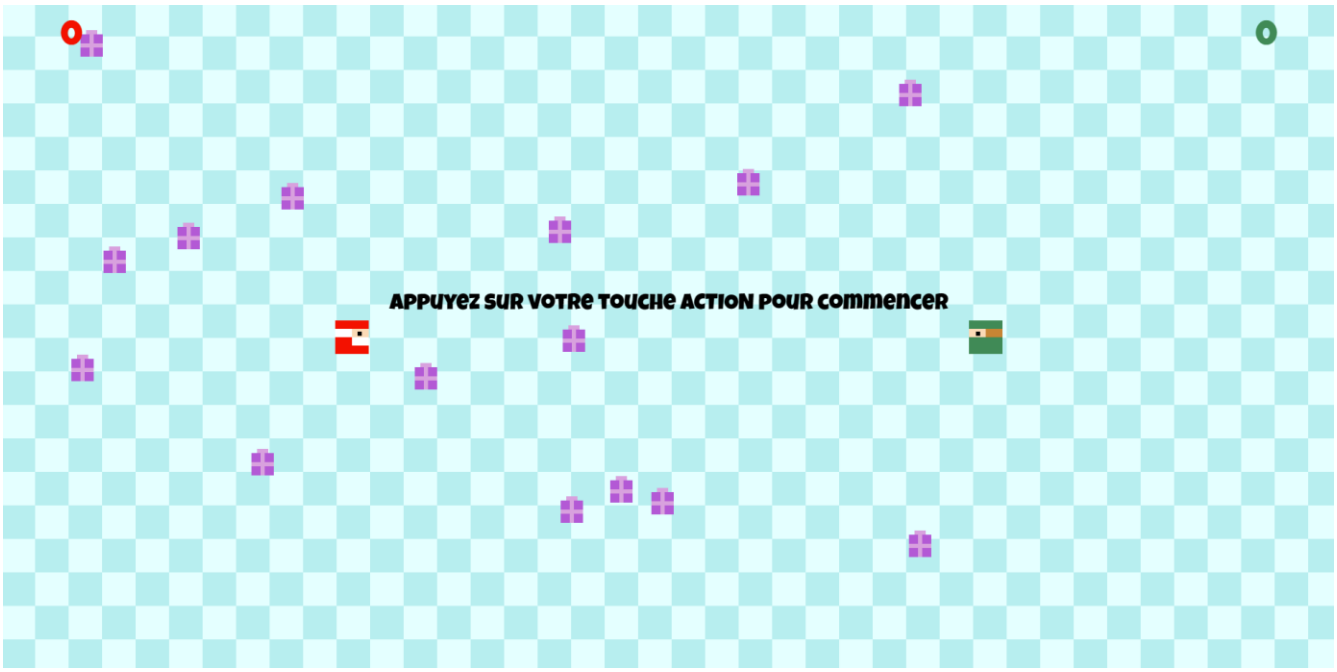


11. Le score (nombre d'objets récupérés) de chaque personnage doit être affiché dans une couleur associée à ce personnage et dans une police de caractères Google.

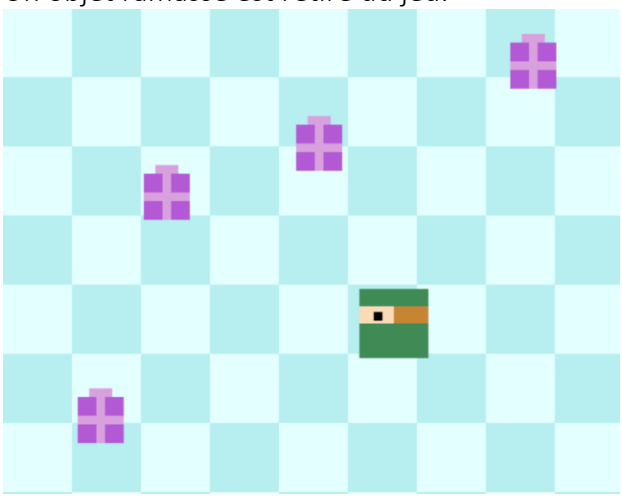
12. Le score du personnage de gauche doit être affiché dans le coin supérieur gauche du canevas, le score du personnage de droite dans le coin supérieur droit.
13. Les scores doivent être mis à jour en cours de partie (la valeur doit augmenter à chaque fois que le personnage ramasse un objet).

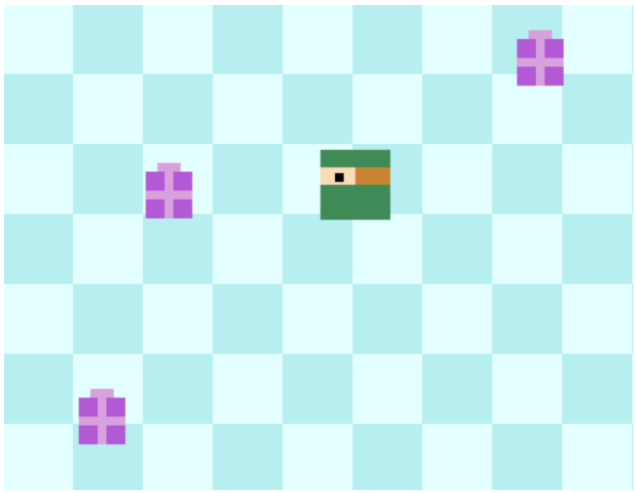
14. Les objets doivent être créés en nombre impair (pour qu'on puisse déterminer un gagnant).
15. Les objets doivent être positionnés aléatoirement à chaque partie.





16. Un objet ramassé est retiré du jeu.





17. En début de partie, les personnages ne sont pas autorisés à se déplacer avant que les deux joueurs appuient sur leur touche d'action. Lorsque les mouvements ne sont pas permis, un message doit être affiché au centre de l'écran.
18. Ce message doit être affiché dans une police de caractères Google.

19. La touche d'action du personnage de gauche doit être « 1 ». La touche d'action du personnage de droite doit être « barre d'espacement ».
20. Lorsque les deux touches d'action ont été appuyées, le message ne doit plus être affiché et la partie est en cours : les personnages peuvent se déplacer et ramasser les objets.

21. Au lancement de la partie (après les deux touches d'actions), un thème sonore doit être joué en boucle.
22. La navigateur doit jouer un son au ramassage de chaque objet, sauf le dernier (voir point suivant).
23. Un son distinctif doit être joué au ramassage du dernier objet.

PISTES DE SOLUTION

Ajout et suppression d'un élément dans un tableau

Dans ce travail, vous aurez à ajouter des objets à un tableau et à retirer (supprimer) des objet d'un tableau.

Pour ajouter un objet :

```
tableau.push(unObjet);
```

Par exemple :

```
posX = ...;
posY = ...;
cadeaux.push({x: posX, y: posY});
```

> Dans cet exemple, un cadeau est représenté par un objet ayant les propriétés **x** et **y**. Ce cadeau sera placé dans un tableau de cadeaux. Ainsi, pour obtenir la position en **x** du deuxième cadeau, on écrira **`cadeaux[1].x`**.

Pour retirer un objet, on peut utiliser la méthode **`splice`** :

```
tableau.splice(indice, 1);
```

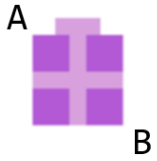
Par exemple, pour retirer un cadeau (**`indiceCadeau`** est l'indice de l'élément à retirer):

```
cadeaux.splice(indiceCadeau, 1);
```

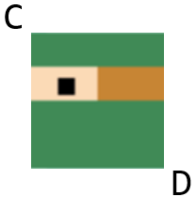
Vérifier une collision

Vous aurez également à vérifier si un personnage touche un objet. Pour y arriver, on peut vérifier si les deux rectangles s'entrecroisent.

Si on considère les coordonnées des coins des deux items pour lesquels on souhaite détecter une collision…




```
A = cadeaux[i].x, cadeaux[i].y
B = cadeaux[i].x + LARGEUR_CADEAU, cadeaux[i].y + HAUTEUR_CADEAU
```




```
C = lutin.x, lutin.y
D = lutin.x + LARGEUR_LUTIN, lutin.y + HAUTEUR_LUTIN
```

… la fonction ci-dessous retournera **`true`** s'il y a collision, **`false`** sinon :

```
function rectanglesCollision(A, B, C, D) {
    if (A.x >= D.x || C.x >= B.x){
        return false;
    }

    if (A.y >= D.y || C.y >= B.y) {
        return false;
    }

    return true;
}
```

Il faudra donc vérifier continuellement les collisions entre les deux personnages et tous les objets :

Personnage de gauche avec objet 1
Personnage de gauche avec objet 2
…
Personnage de droite avec objet 1
Personnage de droite avec objet 2
…

Une boucle sera donc très indiquée pour vérifier les collisions entre tous les objets toujours en jeu et chacun des personnages. Il faudra exécuter ces vérifications après chaque déplacement.

BARÈME

Organisation du site Web		**10**
Importation d'une police de caractères Google		/2
Code JavaScript dans un fichier distinct		/1
Canevas plein écran		/4
Arborescence		/3
Plancher		**10**
Plancher généré par code JavaScript		/5
Présence d'au moins deux (2) tuiles différentes		/5
Personnages		**30**
Deux personnages d'apparences différentes		/5
Déplacement des personnages avec les touches du clavier		/10
Images en fonction de la direction du personnage		/8
Déplacements contraints aux limites de l'écran		/7
Lancement de la partie		**10**
Affichage et effacement du message de lancement de partie		**/2**
Verrouillage et déverrouillage des déplacements		**/8**
Objets et scores		**30**
Génération et positionnement aléatoire d'un nombre impair d'objets à chaque partie		/10
Ramassage des objets (détection de collision et disparition)		/10
Affichage des deux scores aux bons endroits, dans une police Google et avec des couleurs différentes		/6
Mise à jour des scores		/4
Sons		10
Thème musical		/2
Son au ramassage d'un objet		/3
Son au ramassage du dernier objet		/5

Sous-total produit		**100**
Pénalités prévues à la PIÉA pour les travaux remis en retard	–	
NOTE		