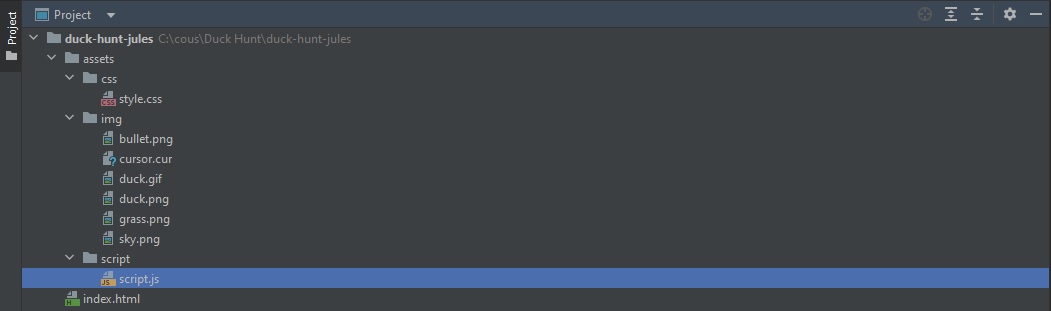
**Réalisation d’un Duck Hunt (HTML, CSS, Javascript)**

1 / Recherche d’informations sur le jeu Duck Hunt.

* Quel est le principe du jeu Duck Hunt ?
* Définir le jeu, quel est l’objectif ?
* Histoire de Duck Hunt ?
  + Langage utilisé
  + Gameplay
  + Accessoire
  + Année
  + Développeur
* Caractéristiques
  + Paramètre de victoire
  + Paramètre de défaite
  + Que signifie UHD ? A quoi ça sert Concrètement ?
* En regardant du gameplay du jeu sur Internet :
  + Quelles sont les ressources dont nous allons avoir besoin ?
* Quels seront les fonctionnalités pour la première version du jeu que nous allons créer ? Elle doit être simple mais fonctionnelle. (Imaginez-vous en train d’y jouer et décrivez tout ce qui va se passer à l’écran avec vos interactions)
* Comment sera calculé le score sur le Duck Hunt que nous allons créer ?
* Pour le futur quelles améliorations voyez-vous pour le jeu ?

2 / Création de l’interface de base

* Récupérer les images que le formateur vous fournis
* Créer un nouveau projet ou vous voulez sur votre ordinateur, ce projet doit comprendre :
  + Un fichier html (index.htm)
  + Un dossier assets, qui comprend :
    - Un dossier css, qui comprend :
      * Un fichier style.css
    - Un dossier img, qui comprend :
      * Toutes les images fournies par le formateur
    - Un dossier script qui comprend :
      * Un fichier script.js



3 / Mise en place du jeu de base

* Générer le code HTML de base nécessaire au bon fonctionnement de votre page
* Relié votre css et votre script à votre page HTML
  + Afin d’être sûr que votre css et js sont bien relié :
    - Faites un console.log dans votre js et regardez s’il apparait bien dans le navigateur
    - Mettez du style sur un paragraphe simple et regardez s’il apparaît bien dans le navigateur
* Ensuite créer le début du jeu, le début du jeu consiste à faire apparaître cet écran : 
* L’élément est une div avec un background gris de 800px \* 450px qui doit être centré sur le body, à l’intérieur de celle-ci se trouve :
  + Une autre div qui va faire 100% de largeur et 100% de la hauteur de son parent (la div grise) qui va comprendre :
  + Un bouton qui va être centré
* Ajouter le CSS nécessaire sur le bouton afin qu’il ressemble au bouton d’exemple.
* Ajout du premier morceau de code en Javascript :
  + Au clique sur le bouton, la deuxième div (celle qui comprend le bouton doit disparaitre).
  + Pensez à bien commenter tout ce que vous faîtes dans votre JS pour ne pas être perdu par la suite
* Après avoir fait disparaitre cette div, faite apparaitre une autre div qui était de base cachée dans votre code au même niveau qui la div disparue. Cette div va contenir tous les éléments de notre jeu :
  + Placez les deux images (sky et grass) en absolute dans votre div afin d’arriver à ce résultat :



* Toujours dans votre nouvelle div, afin de tester si tout fonctionne bien, ajoutez une autre image (duck.gif), toujours en absolute à la fin de votre div, si tout ce passe bien le duck doit passer entre le sky et la grass, de cette façon :



* Si votre duck passe au-dessus de la grass, il va vous falloir utilisez la propriété z-index. Recherchez sur internet comment ça fonctionne.
* Si tout se passe bien vous pouvez maintenant supprimer votre image du duck.

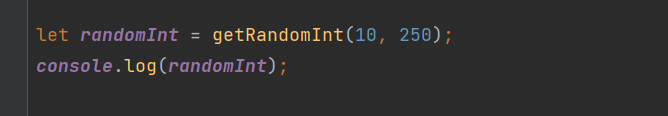
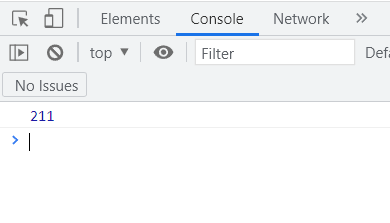
De retour un peu en arrière, au clique sur le bouton Commencer, nous avons donc fait apparaître une nouvelle div qui correspond à notre plan de jeu. Dans le même processus faite appel à une fonction startGame() (qui pour l’instant ne fait que console.log cette phrase : « Début du jeu ! »).

Notre futur code Javascript se passera dans cette fonction startGame.

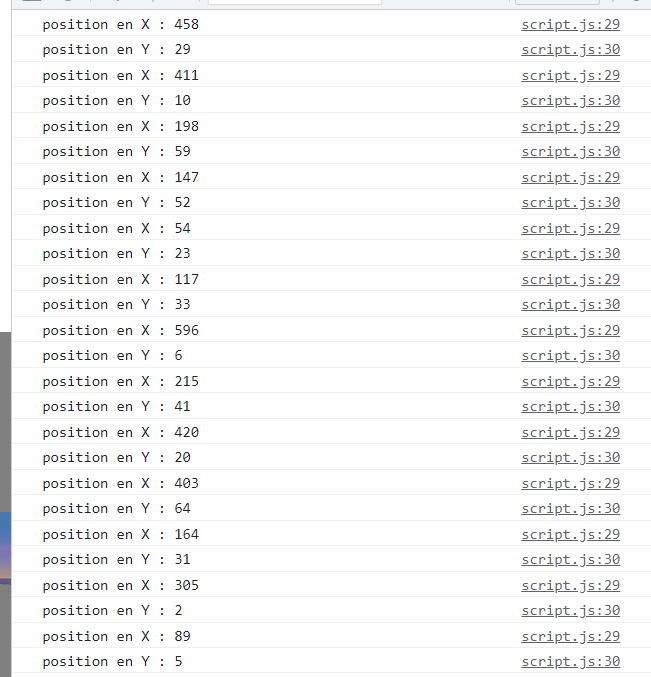
4/ Faire apparaître des canards qui vont de bas en haut dans des directions aléatoires

*Premier vrai algo Javascript*

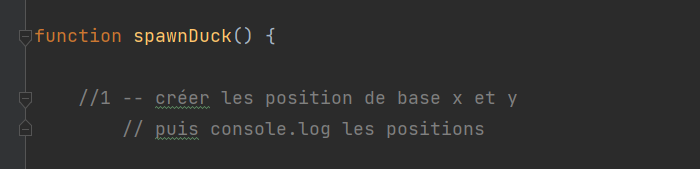
* Renseignez vous sur la fonction setInterval de Javascript, elle va vous être utile.
* En utilisant setInterval vous allez pouvoir effectuer des actions toutes les secondes
* Le premier but dans votre setInterval est de définir la position d’un duck (en X et en Y) sur votre plateau de jeu.
  + Créer une fonction dans votre code « getRandomInt » qui va vous permettre de vous donner un nombre aléatoire entre 2 nombres, exemple :

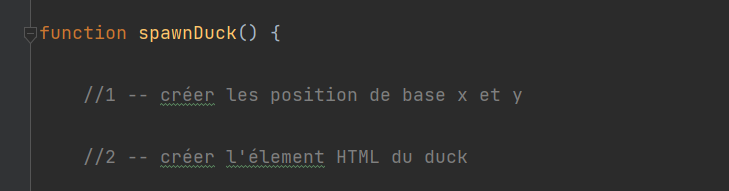
* Une fois votre fonction getRandomInt fonctionnelle, tester la dans votre setInterval, le but est :
  + Toutes les secondes voir apparaitre dans la console une position en X comprise entre **50 et 725** et une position en Y comprise entre **0 et 75**
  + Résultat de la console :



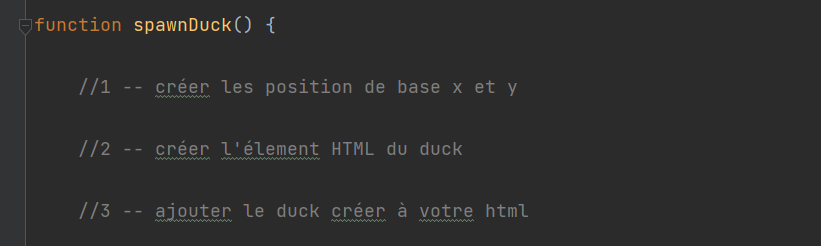
* Très bien ! Notre algo fonctionne, on va pouvoir commencer à faire apparaître des canards.
* Dans votre setInterval, au lieu de directement écrire du code à l’intérieur, faites appel à une fonction spawDuck(), qui elle va se charger de faire apparaître les ducks.
* Dans spawnDuck(), ajouter le calcul de vos position X et Y, ainsi que le console.log, puis vérifiez que tout fonctionne toujours comme avant.



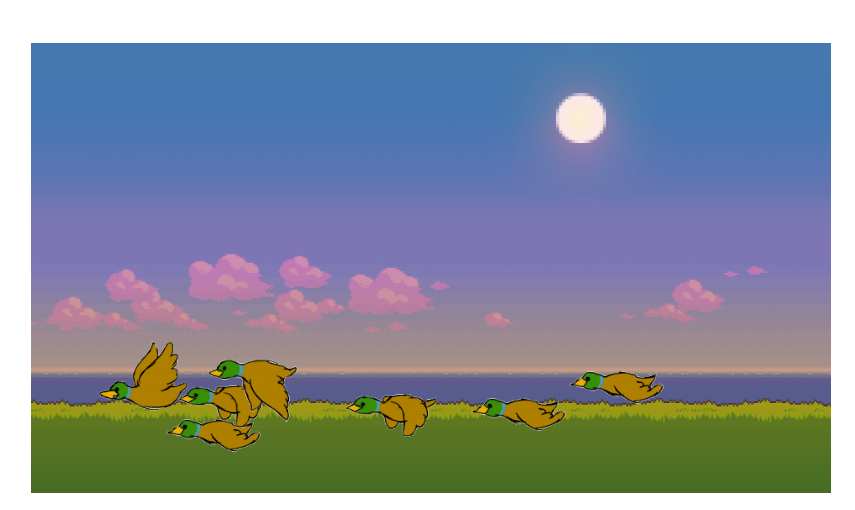
* Maintenant, dans votre fonction spawnDuck après avoir calculé les positions, créer un élément image en javascript (referez-vous à la fonction document.createElement en javascript), vous allez devoir lui rajouter les éléments :
  + Un attribut src qui est relié à l’image du duck.gif
  + Un attribut draggable à false
  + Un style.bottom qui représente la position en Y (oubliez pas de rajouter px !)
  + Un style.left qui représente la position en X (oubliez pas de rajouter px !)
  + Une class css qu’on va appeler « duck »
* EN CSS, customisé votre class duck :
  + Donner lui une width de 100px
  + Une position absolute
  + Et pour l’instant un z-index de 99999



* Votre élément HTML duck est maintenant généré ! Vous n’avez plus qu’a l’ajouter à votre plateau de jeu.
  + Dans votre fonction spawnDuck, après avoir créé l’élément HTML(Je parle bien de la div qui comprend l’image grass et l’image sky), utilisez la fonction insertAdjacentElement qui va vous permettre d’ajouter le duck à votre div.



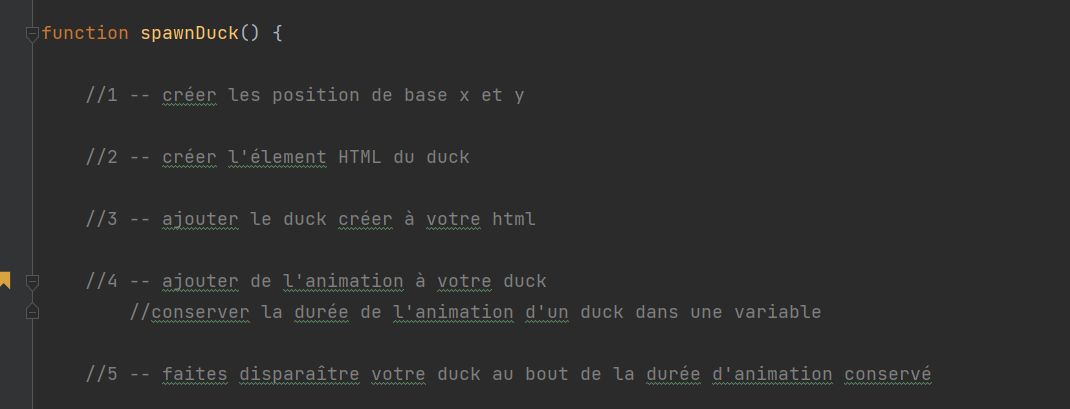
* + Tester le tout, vous devriez voir des ducks qui spawns toutes les secondes en bas de votre écran, comme ceci :



* Si tout se passe bien, modifier dans le css la class duck et retiré le z-index. Tous vos Duck devraient maintenant apparaître derrière la grass ! (Testez)

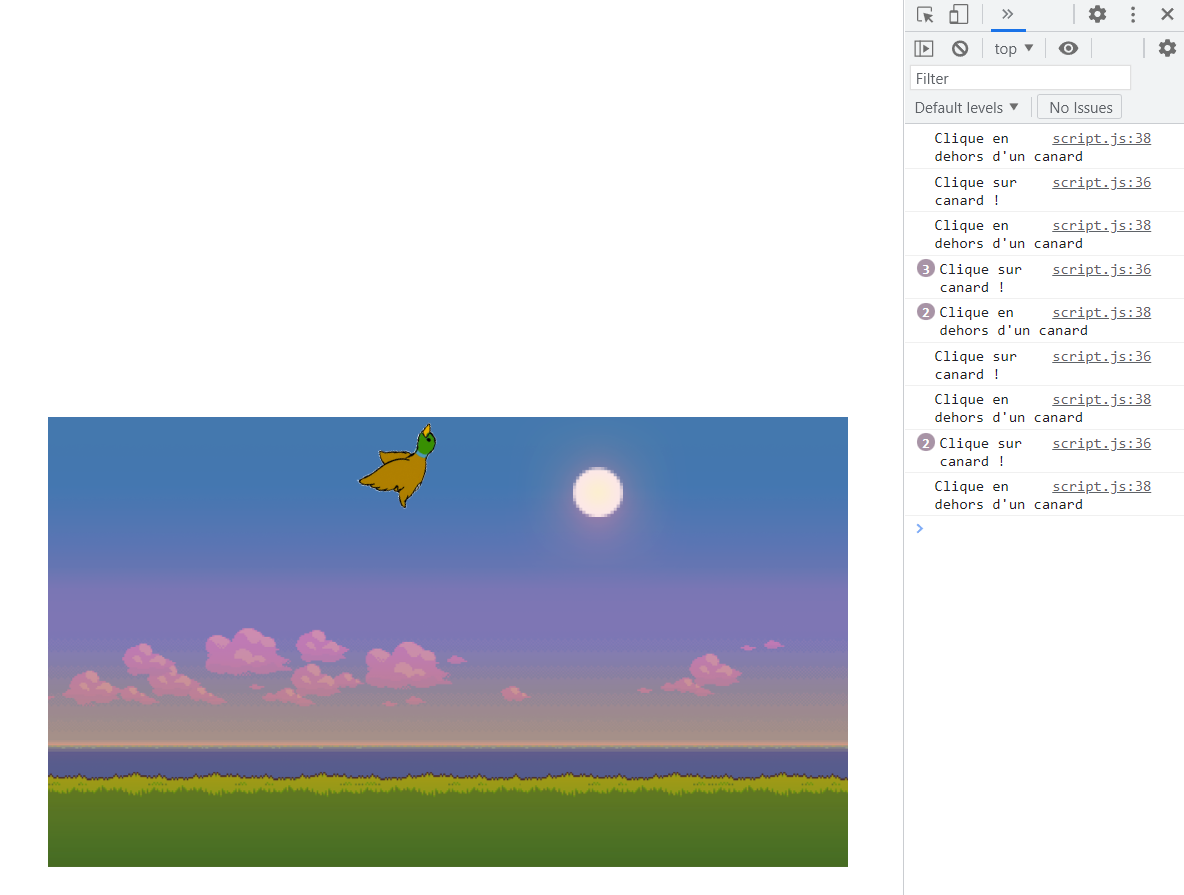


* Super ! On a maintenant des ducks qui spawn toutes les secondes à l’infini en bas de notre écran avec des positions aléatoires. C’était surement la partie la plus **difficile**.
* Faites-en sortent en CSS que vos canards soit orienté vers le haut quand il spawn.
* Maintenant, le but va être de les animer ! Les ducks doivent :
  + Sortir de l’écran en allant vers le haut
  + Se décaler un peu à gauche ou à droite quand ils se déplacent vers le haut, avec une vitesse aléatoire.
  + Faites un peu de recherche sur comment est ce que vous allez arriver à faire ça. (Regardez la fonction animate en Javascript), finalement faîtes disparaître chaque duck au bout de 4s de vie afin de ne pas surcharger votre page HTML. *Voir ci-dessous un pattern de votre function spawnDuck*

**

5 / Intéraction avec la souris sur les ducks

* Sans indications, en CSS trouver comment, uniquement sur le plateau de jeu, changer le curseur de la souris par l’image cursor.cur.
* Créer une fonction addGloabListener() qui va permettre d’ajouter un évènement au clique sur tout votre plateau de jeu, cette fonction doit être appelé après votre fonction startGame(). Au clique dans votre div de jeu, récupérer la target (qui va correspondre à l’élément HTML sur lequel vous avez cliqué), trouver comment identifier si c’est un canard.
* Une fois que vous avez trouver comment identifié si c’est un canard, je veux voir apparaître dans la console :
  + Clique sur canard ! à chaque clique sur un canard
  + Clique en dehors d’un canard, si ce n’est pas un canard

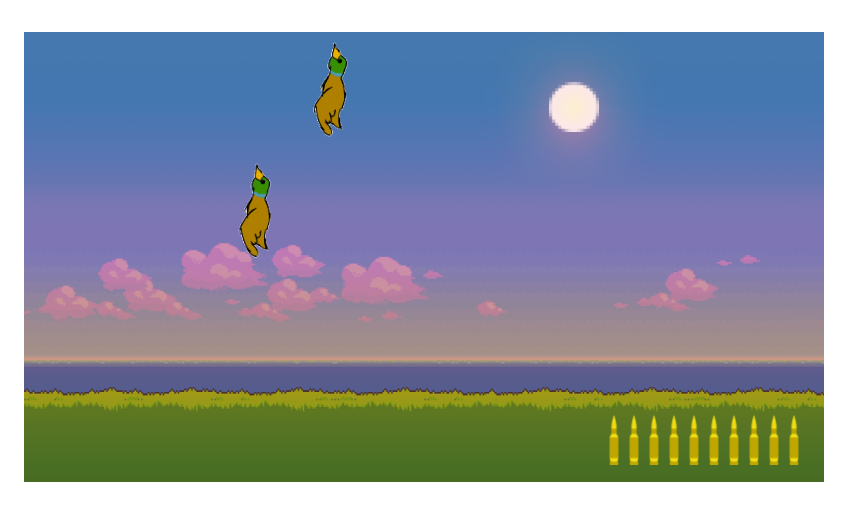


* Ceci étant fait, supprimer les console.log, puis si le clique sur un canard est effectué, supprimer l’élément HTML.

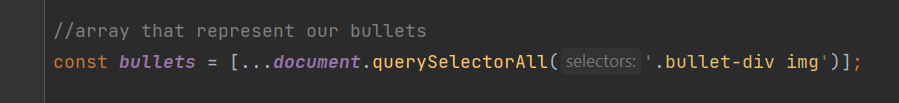
*Nous pouvons maintenant tirer en continu sur les canards.*

6 / Gestion des balles restantes

* Ajouter dans votre GameScreen (directement en HTML), les balles que le joueur possède (10 au début), comme ceci :



* De retour dans le code Javascript, faîtes en sorte que, au clique en dehors d’un canard, une des balles disparaisse (en partant de la gauche), pour se faire :
  + Ajouter une variable dans votre javascript, ça va être un tableau qui va contenir tous les éléments de balles HTML (avec un querySelectorAll). Voici un exemple (appelé le formateur si vous ne comprenez pas l’exemple) :



* + Au clique en dehors d’un canard, supprimer une des balles de votre tableau HTML. (L’affichage ET le stockage, **vous avez bien deux opérations à faire ici**)
  + Après avoir retirer cette balle, regardez si la longueur de votre tableau est de 0, si c’est le cas le joueur a épuisé toutes ces balles et donc pour l’instant, affichez dans la console : GAME OVER.

7 / Gestion du GAME OVER

Pour plus de simplicité et avoir un peu plus de clarté dans le code, nous allons créer un deuxième fichier de script, qui va s’appeler : gameOver.js

* Créer se fichier de script et incorporé le dans votre HTML
* Créer simplement une fonction gameOver() à l’intérieur. Débrouillez-vous pour :
  + Arrêter le jeu et affiché un écran rouge avec écrit GAME OVER en blanc, comme ceci :



* Appelé votre function gameOver lorsque les balles du joueur sont à 0.

8 / Gestion du score

* Ajouter en haut à droite de votre plateau de jeu, un score. *(un peu plus d’autonomie ce coup ci)*
  + Score de base à 0
  + A chaque fois qu’un Duck est touché récupérer le score marqué et ajouter + 2
  + Trouver une solution pour gérer et affichez les combos !
  + Si le joueur fait un enchainement sans perdre de balle le multiplicateur prend + 1
  + Donc le score ne fait pas + 2 à chaque duck mais bien + (2 \* multiplicateur)
  + Bien sur si un duck est manqué le multiplicateur retombe à 1.
* Le multiplcateur est a affiché en haut à gauche de votre écran, voici un exemple complet :

Au démarrage du jeu :



Puis, plus on avance dans le jeu :



* Finalement, dans votre écran de gameOver, affichez aussi le score du joueur.



9 / Fonctionnalités supplémentaires

1. Récupérer un gif d’explosion sur internet et quand un canard meurt, affichez le gif d’explosion à l’endroit où le canard est mort puis le supprimer.
2. Récupérer sur internet un son de pistolet, à chaque clique de la souris jouer le son.
3. Ajouter une musique d’ambiance générale quand le jeu se lance
4. Avant de cliquer sur le bouton démarrer le jeu, l’utilisateur doit rentrer son nom dans un champ.
   1. Vous devez réussir à stocker les scores de chaque personne dans le localStorage du navigateur. (Venez me voir si vous ne voyez pas de quoi je parle)
   2. Sur l’écran d’accueil, ajouter un lien vers une autre page HTML dans laquelle vous allez afficher les scores de tout les joueurs, du plus grand au plus petit.
5. **Difficile**, Ajouter un Super Duck !
   1. Un super Duck à 5% de chance de spawn à la place d’un duck normal.
   2. Il lui faut 2 tirs pour mourir et non 1 comme les ducks
   3. Le score qu’il rapporte est + 20 et non + 2
   4. L’image du super duck est présente dans les images sous duck.png

*//le dernier exercice est très difficile car vous allez devoir adapter tous votre code afin de faire fonctionner le super duck.*