Sophie Bousquet
Liu François

Rapport Trafic Racer

Introduction :

Notre projet consiste en l'implémentation du jeu Traffic Racer en HTML et JavaScript.

Au démarrage du jeu, on a une interface graphique avec un fond et un bouton start.

Lorsqu'on appuie sur le bouton start, une autre interface se lance avec un fond, 12 voitures, un bouton démo et une phrase avec écrit "Choisissez une voiture !". Chaque voiture est un bouton cliquable qui permet de choisir la voiture avec laquelle on peut jouer. Le bouton démo lance le jeu et joue automatiquement en choisissant une voiture aléatoirement. Par la suite, après avoir cliqué sur un des boutons, le jeu se lance.

Le jeu est constitué d'un fond, de notre voiture située au milieu en bas de l'écran, et de voitures qui apparaissent aléatoirement entre 2 temps donnés tout en haut de l'écran sur une des 8 routes. Le jeu est aussi constitué d'une ambiance sonore.

Le jeu est constitué de 4 déplacements avec les flèches du clavier, quand on appuie sur la flèche du haut, on avance, ce qui donne un effet d'accélération, un bruit d'accélération et de la fumée. Les touches gauche et droite nous font aller à gauche ou à droite en inclinant la voiture de 10 ou -10 degrés. La flèche du bas sert à reculer, mais donne un effet visuel et sonore de freinage, ainsi que des traces de freinage.

Lorsqu'on rentre en collision avec une voiture, on perd une vie, un bruit d'accident et une explosion est alors effectuée. Au total, nous avons 3 vies. Lorsqu'on n’a plus de vie, le jeu s'arrête et une interface de fin apparaît.

L'interface de fin est constituée d'un fond, d'une image de Game Over ainsi que de notre score qui représente le temps qu'on a survécu en seconde et d'un bouton permettant de relancer le jeu.

Explication du code JavaScript :

Le début de notre programme consiste en la création du canevas et en la création de 5 variables, "start", "selection_voiture"," jeu", "demo" et "fin", qui serviront plus tard à faire des vérifications afin de délimiter les différentes interfaces et à les créer.

Puis on a créé 3 autres variables, "vitesse", "vie" et "time_collision". "vitesse" sert à définir la vitesse de défilement de la route ainsi que la vitesse de déplacement des voitures, "vie" sert à définir le nombre de vies du joueur et "time_collision" sert à savoir à quel moment on a touché un civil afin de savoir quand on peut ravoir la collision après que la voiture a été invincible pendent environ 3 secondes.

Par la suite on crée 3 variables, "score", "time_debut" et "time_fin". "time_debut" sert à récupérer le temps du moment où on le lance, "time_fin" fonctionne pareil. Le score qui est en seconde est donc calculé en faisant une soustraction entre "time_fin" et "time_debut" pour récupérer le temps que l’on a tenu en jouant.

Ensuite, on a 4 variables, "demo_move" qui va servir dans la démo à aller à droite ou à gauche aléatoirement, "demo_move_up" qui va servir à avancer, reculer ou ne rien faire, avec une plus forte probabilité de ne rien faire. Puis on a les variables "rand" et "rand_up" qui prendront une valeur aléatoire entre 0 et 1 (inclus) pour "rand" et 0 et 4 (inclus) qui serviront comme indice pour les variables "demo_move" et "demo_move_up".

De plus, on a créé 4 variables, "time_civil", "cood_route", "alpha" et "nb_alpha". "time_civil" est un tableau contenant des valeurs de temps aléatoire pour chaque route afin de faire apparaître les civils et "cood_route"

est la position x de toutes les routes où les civils apparaîtront. Puis "alpha" est un tableau qui sert plus tard à faire clignoter la voiture lors d'un impact avec un civil en prenant comme indice "nb_alpha".

Maintenant, de la ligne 40 à la ligne 100, nous avons implémenté toutes les images qui nous seront utiles pour notre projet, à savoir les fonds, les boutons, les voitures du joueur, les voitures des civils et les effets visuels, en utilisant la fonctionnalité "new Image ()".

Par la suite, de la ligne 104 à la ligne 117, nous avons implémenté tous les sons utilisés dans le projet, que ce soit un son d'accélération, de freinage, d'accident de la voiture du joueur, ainsi que le son de circulation et des sirènes de police pour l'ambiance du jeu.

Maintenant, on a créé une classe "Pt" qui sert à créer les points de nos collisions. Elle prend 2 valeurs, x et y, qui correspondent à ses coordonnées. Dans cette classe, on a une seule fonction, "rotate", qui sert à faire rotationner les points afin de, plus tard, pouvoir tourner nos collisions avec les mouvements de la voiture. Cette fonction prend 2 arguments : "ref" et "angle". "angle" correspond à l'angle de rotation et "ref" au point autour duquel on va tourner.

Ensuite on a 2 fonctions : "cw" et "cwConstant". "cw" sert à calculer le produit vectoriel des vecteurs formés par les points a, b, et c.

Puis "cwConstant" permet de savoir si le point a se trouve à l'intérieur de "P". Elle parcourt chaque côté du polygone et utilise la fonction "cw" pour vérifier l'orientation des points. Si à un moment donné, l'orientation est négative, la fonction retourne false, ce qui veut dire que le point a est à l'extérieur du polygone, sinon ça renvoie true.

Maintenant, nous avons créé une classe "Rect" qui nous servira pour les collisions et la gestion des voitures civiles ainsi que celle du joueur.

Le constructeur de la classe est constitué de 6 paramètres, "xi" et "yi" qui servent à initialiser la position en haut à gauche de notre image ainsi que de notre collision, de même, "xf" et "yf" servent à initialiser la position en bas à droite. Ensuite, nous avons "im" et taille, "im" est un tableau contenant plusieurs images. "im" sert surtout pour les civils et la démo afin de sélectionner aléatoirement une image dans le tableau et "taille" est un tableau de 2 valeurs contenant la largeur et la hauteur des images.

Ensuite, on a initialisé tous les paramètres et on a créé 5 nouvelles variables, "img_id" qui est l'indice que l'on va utiliser dans "im" afin de prendre une valeur aléatoire, "img_id" prend une valeur aléatoire entre 0 et le nombre d'images contenues dans "im". Puis on a la variable "collision" qui est initialisé à false et qui sert plus tard pour le clignotement de la voiture. Puis la variable "angleInDegrees" qui est initialisé à 0 et qui sert à la rotation des images, elle contient l'angle de rotation avec lequel on va faire pivoter l'image. De plus, on a la variable "P" qui contient 4 points formés par la classe "Pt" et qui contient les 4 points de notre rectangle, ainsi que la variable "center" qui se situe au milieu du rectangle et qui sera utilisé par la rotation.

Par la suite, on va créer la fonction "AABBcollide" qui prend un paramètre "otherRect" qui correspond aux coordonnées d'une autre collision/image. Cette fonction va vérifier grâce aux coordonnées si on est en collision ou non avec une autre collision en nous renvoyant true ou false.

Nous avons par la suite la fonction "afficher" qui ne prend pas de paramètre et qui sert à afficher l'image et effectuer une rotation. Au début de notre fonction on sauvegarde le contexte actuel afin de le restaurer à la

fin de notre fonction, puis on déplace des coordonnées "xi", "xf", "yi", "yf" afin de les mettre au centre de notre rectangle pour après effectuer notre rotation en fonction de "angleInDegrees" puis on affiche l'image avec la rotation. Ensuite, on restaure le contexte sauvegardé plus tôt afin de remettre les coordonnées à leur place, puis on a un "for" qui sert à rotationner chaque point du tableau "P[]" autour de "center" de

"-angleInDegrees".

De plus, on a créé la fonction "move" qui prend 2 paramètres "dx" et "dy". Cette fonction sert à décaler "xi" et "xf" de "dx" pixel et pareil pour "yi" et "yf" de "dy" pixel. Puis on a fait pareil pour les coordonnées contenu dans "P". Puis on appelle la fonction "new_center" que l'on va expliquer plus tard.

De même, on a créé la fonction "apparait" qui prend 2 arguments "dx" et "dy" mais qui, à contrario de "move", ne décale pas mais positionne les coordonnées "xi", "xy", "yf", "yi" ainsi que celle contenu dans "P" à la position "dx" et "dy". Puis on appelle la fonction "new_center" que l'on va expliquer plus tard.

Par la suite, nous avons la fonction "bord" qui prend 2 paramètres "go_right" et "go_up", cette fonction permet de vérifier si le déplacement dépasse ou non les bords du canevas en renvoyant true ou false.

De même, nous avons créé la fonction "new_center" qui permet de recalculer le centre de nos voitures à chaque mouvement.

Enfin, nous avons créé la fonction "test_collision" qui prend un paramètre, "otherRect", qui utilise "cwConstant" afin de savoir si nos 2 rectangles sont en collision en regardant point par point dans la boucle.

Ensuite on a 2 variables, "go_up" et "go_right", qui serviront pour les déplacements.

On utilise "addEventListener" afin de vérifier si une touche est enfoncée ou si elle ne l'est pas en appelant la fonction "keydown_func(e) " et "keyup_fun(e) ".

La fonction "keydown_fun" sert à vérifier si les flèches du haut, du bas, de gauche et de droite du clavier sont enfoncées. Au début, on vérifie si "demo" est à true afin de faire un return pour ne pas pouvoir faire de mouvement.

La flèche de droite met "go_rigth" à vitesse et on pivote notre voiture ainsi que celle de police à 10 degrés, ainsi que la flèche de gauche mais c'est -vitesse et -10 degrés. La flèche du haut met "go_up" à -vitesse, lance le son d'accélération et positionne les images de fumée, la flèche du bas fait pareil mais elle met "go_up" à 10, lance le son de freinage et positionne des traces de pneu.

Puis la fonction "keyup_func" vérifie aussi si "demo" est égale à true. Dans cette fonction, si les flèches de droite et de gauche ne sont pas enfoncées, alors "go_rigth" est mis à 0 et on remet notre voiture ainsi que celle de police à l'endroit. Pareil pour les flèches du haut et du bas avec "go_up".

Ensuite, on crée grâce à la classe "Rect" la voiture du joueur, la voiture de police ainsi que 20 voitures de civils.

Par la suite on crée 4 variables, "fond_y" et "fond_y2" qui permettent de faire bouger le fond et "minTime" et "maxTime" qui servent au temps maximal et minimal d'apparition des voitures de civils.

Puis on crée le bouton "bouton". Ce bouton permet de lancer le jeu. Le bouton "bouton_refresh" permet de relancer le jeu. Le bouton "bouton_demo" ainsi que le 12 bouton correspondant à la voiture que le joueur doit sélectionner.

Ces boutons sont créés grâce à "document.createElement("img")" qui crée un nouvel élément "HTML img". Pour tous les boutons on va leur initialiser leur image, leurs coordonnées ainsi que leur taille. Ensuite ils sont ajoutés au canevas et cachés grâce à " style.display". = "none"

De plus, chaque bouton a un "addEventListener" afin de détecter si un bouton est cliqué pour qu'il fasse une action.

Lorsqu'on appuie sur le bouton "bouton", cela met la variable "start" à false et "selection_voiture" à true afin de passer à l'interface de sélection de voiture. Puis on cache le bouton.

Le bouton "bouton_refresh" met "fin" à false et "start" à true, puis cache le bouton.

Le bouton "bouton_demo" cache tous les boutons des voitures et initialise "time_debut" au temps actuel, met "selection_voiture" à false, met "demo" et "jeu" à true. Puis, on initialise "minTime" à 4 et "maxTime" à 6 et on cache le bouton.

Les boutons des voitures initialisent "car.img_id" à l'index de la voiture puis cachent les boutons de toutes les voitures ainsi que celui de "bouton_demo", puis initialise "time_debut" au temps actuel et met "selection_voiture" à false et "jeu" à true.

Puis on crée la fonction "update" qui sera appelée toutes les 20 millisecondes dans un "setInterval" à la toute fin du programme.

On commence la fonction "update" par un "if(start) " afin de lancer la première interface du jeu. Dans ce "if", on commence par afficher le fond "backgroundStart" ainsi que d'afficher le bouton "bouton". Puis, on met "time_debut", "time_fin" et "score" à 0 pour que, lorsqu'on relance le jeu, les valeurs ne s'accumulent pas.

Puis on crée un nouvel "if" à l'extérieur de l'autre "if", qui va vérifier si "selection_voiture" est égal à true. Si c'est le cas, on va réafficher le même fond. Puis on va afficher tous les boutons des voitures ainsi que le bouton "bouton_demo". Par la suite, on va créer un texte grâce à "fillText" qui affichera "Choisissez une voiture !" et on initialisera "minTime" à 3, "maxTime" à 5 et "vitesse" à 2.

Ensuite on crée un nouvel "if" en dehors de celui d'avant, qui va vérifier si "jeu" est égal à true. Si c'est le cas, on va lancer le son "circulation" puis afficher "backgroundImg" 2 fois mais un avec "fond_y" et l'autre avec "fond_y2" comme coordonnée y afin que les 2 fonds se suivent. On va par la suite multiplier par 1.5 "fond_y" et de même pour "fond_y2", puis on va vérifier si "fond_y" et "fond_y2" sont supérieur a "cnv.height" pour ensuite les déplacer afin de créer une route continue.

On va vérifier si "demo" est à true. Si c'est le cas on bouge notre voiture de -200 de coordonnée y afin de vérifier s'il y a une collision avec une autre voiture, si c'est le cas on met la variable "bouger" à true puis on remet la voiture à sa position initiale et on refait la même chose mais de -100.

Puis on va vérifier si "bouger" est à true. Si c'est le cas on va bouger notre voiture et celle de police aléatoirement de gauche à droite et possiblement de haut en bas ou rien. Puis on vérifie si notre déplacement est vers le haut ou bas. Si c'est le cas, on lance le son d'accélération et on positionne la fumée ou on lance le son de freinage et on positionne les traces de pneu.

Puis on va vérifier si notre déplacement est à gauche ou à droite afin de rotationner notre voiture ainsi que celle de police de -10 ou 10 sinon remettre notre voiture et celle de police droite et afficher notre voiture.

On va par la suite vérifier si on est en collision avec une autre voiture là où la voiture compte se déplacer, grâce à "test_collision", si c'est le cas notre voiture et celle de police iront à l'opposé, puis on réaffiche la voiture.

Enfin on vérifie si la voiture est en collision avec un bord au prochain déplacement, si c'est le cas, on décale la voiture et la voiture de police.

Si "bouger" était à false, alors on aurait mis la voiture et celle de police droite puis mis "bouger" à false et on aurait redonné une valeur aléatoire à "rand" et "rand_up".

Si "demo" était à false, on aurait bougé notre voiture ainsi que celle de police à "go_rigth" et "go_up" puis on aurait vérifié les bords comme dans le "if (demo)".

Ensuite on vérifie si "car.collision" est égal à true. Si c'est le cas on fait clignoter la voiture en utilisant "global_alpha" avec pour indice "nb_alpha", puis on affiche la voiture.

De plus, on vérifie si "vie" est égale à 1 afin d'afficher la voiture de police et de lancer le son des sirènes de police contenue dans la variable "sirene".

De même, dans le "for" on fait bouger les civils en rajoutant la valeur de "vitesse" à la coordonnée i de tous les civils, puis on les affiche.

Puis les 2 prochains "for" permettent de déplacer et d'afficher les traces de pneu et de fumée.

On vérifie si cela fait 3 secondes que la voiture clignote. Si c'est le cas elle arrête de clignoter.

Puis on vérifie si "car.collision" est à "true" et si cela fait moins de 1 seconde qu'on a eu la collision. Si c'est le cas, on positionne et on fait apparaitre l'explosion. Sinon on la fait disparaitre du canevas.

Ensuite, on refait bouger tous les civils comme dans le "for" précédent, puis on vérifie si notre voiture est en collision avec une autre voiture. Si c'est le cas on lance le son "accident", on retire une vie, on initialise "car.collison" à true, on initialise "time_collision" au temps actuel et on met "nb_alpha" à 0.

Par la suite, on vérifie le temps de chaque civil un par un afin de voir si leur temps est inférieur ou égal au temps actuel. Si c'est le cas, on vérifie si le civil est en collision avec le bord. Si c'est le cas on le fait réapparaître en haut d'une route aléatoire, puis on change le temps du civil et on fait ça pour tous les civils.

De plus, on vérifie si la variable "vie" est à 0, et si c'est le cas on arrête le son de circulation et des sirènes de police, puis on initialise "time_fin" au temps actuel et on calcule le score en seconde en soustrayant "time_debut" à "time_fin" et on divise le résultat par 1000 pour l'avoir en seconde.

Ensuite, on vérifie si "fin" est à true, et si c'est le cas on affiche le fond "backgroundfin" et l'image "imggameover" puis, on fait apparaître le bouton "bouton_refresh" qui permet de relancer le jeu. Par la suite, on va afficher un texte écrivant "Votre score : " avec notre score en seconde "seconde".

On initialise toutes les variables utilisées à leur valeur de base et on met les civils hors du canevas afin de ne plus les voir.

Enfin, on crée un nouveau "setInterval" qui se lance toutes les 20 secondes et qui lorsque "jeu" est égal à true, ajoute 1 à "vitesse" et si "minTime" est supérieur à 1 on lui enlève 1 et si "maxTime" est supérieur à 2 on lui retire 1.

Puis on appelle la fonction "update" dans un "setInterval" qui se lance toutes les 20 millisecondes.

Problème rencontré :

Le plus gros problème rencontré a été les collisions de base. Nous avions juste un rectangle avec la classe "Rect" sans la classe "Pt". Mais pour les rotations il était impossible de faire pivoter le rectangle. Donc on a dû changer la gestion des collisions en ajoutant la classe "Pt".

Répartition du travail :

Sophie Bousquet :

Elle s'est occupée des différentes interfaces, start, sélection de voiture, démo et de celui du game over.

Elle s'est occupée des déplacements et des rotations des voitures.

Elle s’est occupée de la voiture de police.

Elle s'est aussi occupée de l'implémentation des sons et de tous les boutons.

Enfin, elle s'est occupée de la création des civils ainsi que la vitesse qui augmente avec le temps et des setInterval.

Liu Francois :

Il s’est occupé des collisions des véhicules ainsi que des collisions entre les voitures.

Il s’est occupé de la création des 3 vies, du clignotement de la voiture lors de l’impact et de ne pas faire sortir les voitures du cadre.

Il s’est occupé du fait que les civils augmentent au fil du temps et qu’il y en ait plus.

Enfin, il s’est occupé du positionnement des civils sur chaque route.