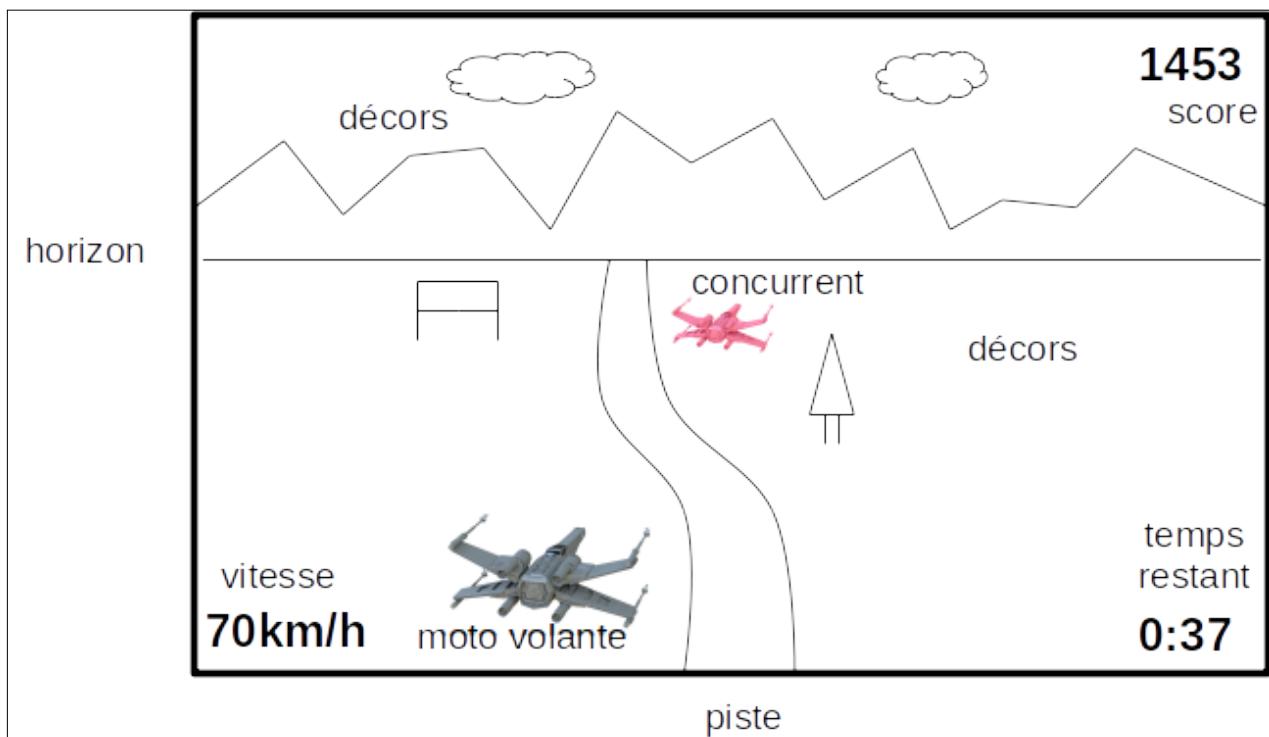

PROGRAMMATION CONCURRENTE ET INTERFACES INTERACTIVES



Rapport de projet

Préparé pour : Nicolas SABOURET

Préparé par : FERREIRA ALVES Patricia, BORISSOV Piotr

3 mai 2020

PROGRAMMATION CONCURRENTE ET INTERFACES INTERACTIVES

INTRODUCTION : PRÉSENTATION DE L'OBJECTIF DU PROJET

Objectif (rappel)

Ce projet vise à réaliser un jeu vidéo simulant une course de vaisseaux avec caméra arrière. Le joueur doit atteindre les points de contrôles apparaissant sur la piste en un certain temps, tout en évitant les obstacles et les adversaires éventuels.

Buts

Réaliser un programme répondant au cahier des charges implémentant une expérience utilisateur fluide. Réaliser un code soigné, documenté et compréhensible pour d'autres programmeurs.

Solution

Le programme doit donc générer une piste, ainsi que des obstacles et des adversaires de manière aléatoire. Il crée également un vaisseau, dont les mouvements sont contrôlables par le joueur. La vitesse de l'appareil augmente au plus proche de la piste. En cas de collision avec un obstacle ou un adversaire, l'engin volant perd de la vitesse.

Dans le cas où le temps imparti, augmentant à chaque point de contrôle, atteint la valeur nulle : la partie est perdue.

L'interface doit afficher : la piste (avec décors), le véhicule, les obstacles et les adversaires, ainsi que le score du joueur, la vitesse du véhicule, le temps restant avant la fin de la partie. Les différents éléments devront être représentatifs de la situation réelle.

Contour du projet

Le projet s'organise en une dizaine de séances de travail. Les premières effectuées dans les locaux de l'université Paris-Saclay, les suivantes à domicile pour cause de la situation exceptionnelle liée au COVID 19.

Le travail de chaque séance est divisé en trois temps :

- Coder
- Remplir le diagramme de Gantt
- Rédiger le rapport ci-présent

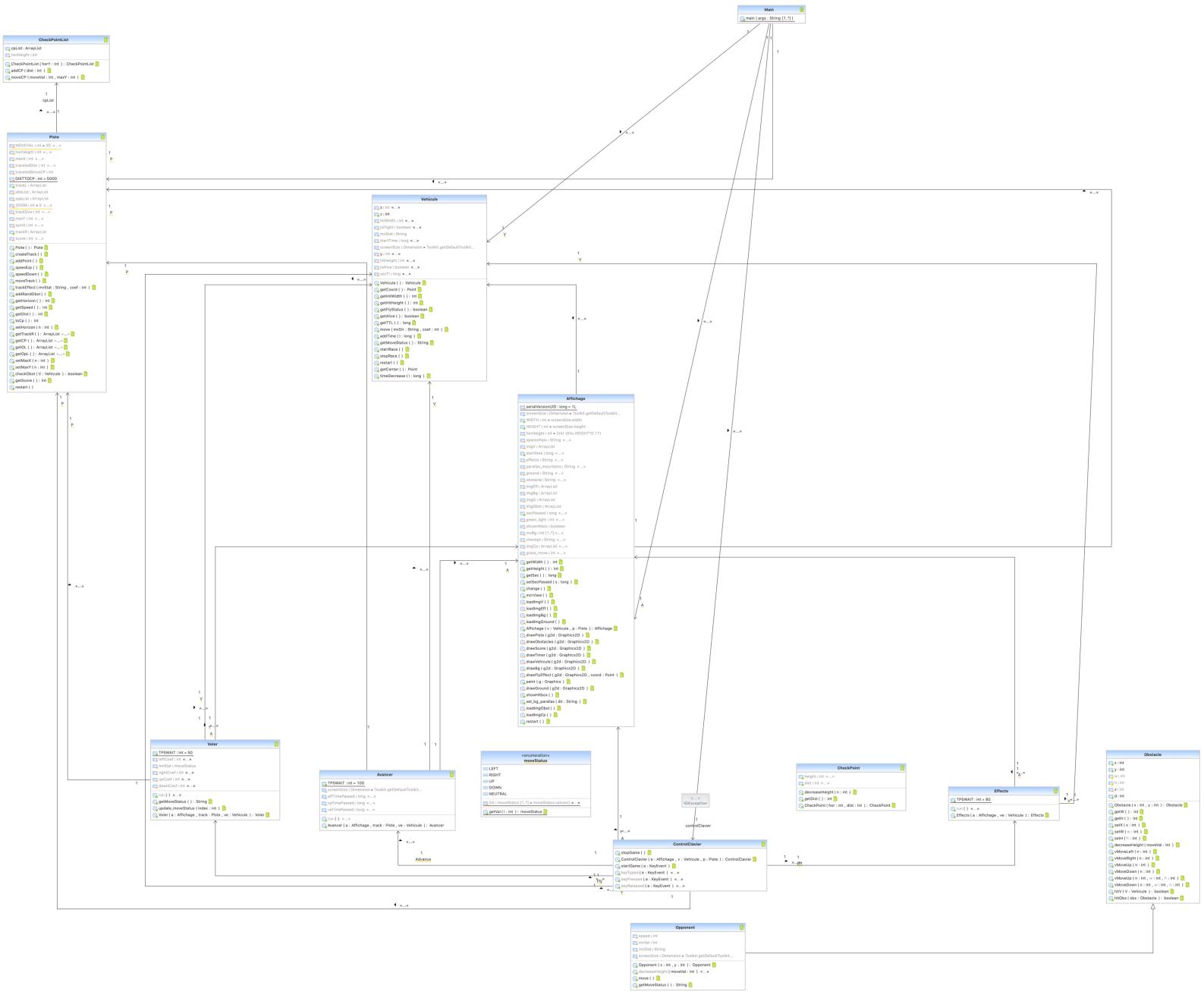
PROGRAMMATION CONCURRENTE ET INTERFACES INTERACTIVES

CAHIER DES CHARGES

Fonctionnalités REQUISSES		
OUI	R1	<p>Une vue avec :</p> <ul style="list-style-type: none"> - Un véhicule, représenté par un dessin ou par un ensemble d'images ; - Un horizon (représenté par un trait) surmonté d'un décors qui se déplace à droite et à gauche en fonction des mouvements du véhicule ; - Une piste infinie, calculée à partir d'une ligne brisée verticale limitée par l'horizon et générée aléatoirement ; - À intervalles réguliers apparaissent des points de contrôles matérialisés par une bande horizontale sur la piste.
OUI	R2	Un mécanisme de contrôle au clavier pour déplacer le véhicule horizontalement ou verticalement
OUI La vitesse du joueur ne peut jamais atteindre 0 par choix	R3	<p>Un modèle de jeu comprenant :</p> <ul style="list-style-type: none"> - L'état du véhicule : position et vitesse ; - Un mécanisme de calcul de l'accélération du véhicule en fonction de la position par rapport à la piste ; - Un mécanisme de calcul de la vitesse en fonction de l'accélération ; - Des données de jeu : temps restant et kilométrage ; - À chaque point de contrôle, un temps supplémentaire est alloué. - Lorsque la vitesse du joueur atteint 0 ou lorsque le temps alloué atteint 0, la partie se termine et le score est affiché.
Fonctionnalités COMPLÉMENTAIRES		
OUI	C1	<p>Apparition aléatoire d'obstacles « immobiles » sur la piste (générés hors du champ de vision et apparaissant au fur et à mesure).</p> <ul style="list-style-type: none"> - Lorsque le véhicule percute un obstacle, sa vitesse diminue d'une valeur fixe.
OUI	C2	<p>Obstacles « mobiles » sur la piste : il s'agit de concurrents qu'il faut dépasser.</p> <p>On attribue un bonus de 10 points au score pour chaque concurrent dépassé. Ces concurrents disparaissent lorsqu'ils sortent du champ de vision.</p>
OUI	C3	<p>Le dessin du véhicule change en fonction des actions du joueur pour suggérer les mouvements à droite et à gauche (par exemple, vous pouvez incliner la moto lorsqu'elle tourne).</p>
OUI	C4	Le véhicule redescend tout seul vers le sol lorsqu'il perd de la vitesse.
OUI	C5	Le dessin de la piste se rétrécit au bout pour créer une sensation de profondeur.
OUI	C6	Ajouts d' objets de décors autour de la piste .
NON Effet parallax mis en place	C7	Défilement d'objets sur le fond (nuages)
Fonctionnalités OPTIONNELLES		
NON	O1	Mémorisation des meilleurs scores.
NON Jugée inutile	O2	Possibilité de contrôler la vitesse de la moto (accélérer/ralentir).
Fonctionnalités ADDITIONNELLES		
OUI	A1	Montrer / Cacher les hitbox de la piste, des véhicules, des obstacles et des adversaires
OUI	A2	Possibilité de relancer la partie une fois cette dernière finie.

PROGRAMMATION CONCURRENTE ET INTERFACES INTERACTIVES

DIAGRAMME DE CLASSE DE L'APPLICATION



PROGRAMMATION CONCURRENTE ET INTERFACES INTERACTIVES

DEVELOPPEMENT DE L'APPLICATION

Choix du support

Le projet est réalisé en Java, en utilisant l'API Swing afin de créer l'interface interactive.

Stratégie

Nous avons structuré ce projet en quatre packages différents : le premier contenant une classe Main qui crée les objets nécessaires et qui lance le jeu, et les trois autres correspondant aux trois parties du modèles MVC : le modèle *model*, la vue *view* et le contrôleur *control*.

Fonctionnement

Au lancement du programme, celui-ci crée une piste et un véhicule affiché sur l'interface, avec un décor au dessus de l'horizon. Une fenêtre de dialogue s'affiche pour informer l'utilisateur des différentes commandes.

Le véhicule est accompagné de petites lumières vertes qui tournent en dessous de lui.

On détecte si l'utilisateur appuie sur les flèches du clavier avec une classe keyListener, ce qui permet de lancer le jeu, de déterminer si le véhicule bouge ainsi que le sens du déplacement. Un autre thread exploite ces valeurs pour modifier le véhicule est la piste : le véhicule est déplacé dans la direction indiquée. Les points de la piste se déplacent vers la gauche, si le véhicule va à droite et inversement si le véhicule va à gauche. Si le véhicule monte, les points de la piste s'éloignent pour donner une impression de profondeur, et si le véhicule descend, on les rapproche.

Un thread différent est utilisé pour donner l'impression que le vaisseau avance. Il diminue la vitesse si le vaisseau s'éloigne de la piste et inversement. Ce thread appelle des fonctions permettant d'augmenter la hauteur des points de la piste (en visuel donne l'impression qu'ils descendent de l'horizon et partent vers l'arrière du vaisseau). De nouveaux points sont créés à l'horizon si besoin, afin que la piste soit générée à l'infini.

Le décor affiché au dessus de l'horizon est également modifié : les différentes couches le composant se décalent dans la direction opposée à celle du vaisseau. Afin de donner une impression de perspective, les éléments se trouvant à l'arrière sont moins déplacés que ceux devant.

La valeur de la variable représentant le score est augmentée en parallèle de la distance parcourue. La vitesse indique à quel point la piste et le score sont modifiés.

Des points de contrôles sont générés tous les 5000m lorsque le vaisseau avance d'une certaine distance. Ils sont représentés par des damiers formant une ligne horizontale sur l'affichage et avancent avec piste.

Des obstacles classiques représentés par des arbres et des adversaires mobiles constituent les obstacles qui apparaissent au niveau de l'horizon. Les arbres se déplacent avec la piste. Les adversaires mobiles incorporent une notion de vitesse. Les obstacles se déplacent et s'agrandissent selon les mouvements du vaisseau ils changent aussi de taille selon leur distance. Plus ils sont proche de l'horizon, plus ils sont petits.

L'apparition proche de la piste et l'abscisse des obstacles ainsi que les déplacements des adversaires sont aléatoires. Enfin, si le vaisseau touche un obstacle, sa vitesse diminue à la valeur la plus basse.

Une gestion du temps est implémentée en calculant la différence du temps courant par rapport au temps du début de programme. Le temps est affiché en haut à droite à l'écran. Si il s'écoule, le joueur perd la partie et le jeu se termine. Ce temps est augmenté lorsque le vaisseau dépasse un point de contrôle.

Tous les threads modifiant les modèles doivent également notifier la vue, afin qu'elle puisse prendre en compte les changements.

PROGRAMMATION CONCURRENTE ET INTERFACES

ORGANISATION DU TRAVAIL

Les rôles

Nous nous sommes mis en binôme quelque peu après le début de la séance et Patricia avait déjà implémenté un simple squelette du code basé sur le modèle fait lors du tutoriel Flappy Bird.

Notre premier réflexe fut de créer une repo sur GitHub, cela facilite grandement le travail et fluidifie l'avancée du projet : <https://github.com/upsudghosts/MVNI.git>.

Dans un second temps nous avons mis sur papier, après avoir relu le sujet, les éventuelles classes dont nous aurions besoin. Plus généralement nous nous sommes mis d'accord sur la direction globale qu'allait prendre notre code afin d'éviter tout soucis d'uniformisation pendant l'avancement du programme.

Enfin la répartition des tâches s'est faite assez naturellement. Nous suivions l'ordre des choses comme indiqué dans le sujet. Chacun choisit rapidement ce qu'il souhaitait développer et sans plus attendre commença à travailler.

Toutefois, nous gardions à l'esprit que le travail est commun.

Ainsi, chaque semaine eut lieu un débriefing de la part de chacun des membres afin d'expliquer sa partie à l'autre. (si il n'y avait aucune corrélation entre les tâches. Ex: mettre des images pour le background vs faire le timer) Aussi nous n'hésitions pas à nous entraider lors d'éventuels blocages.

Le Projet évolue ainsi sans encombre et les résultats sont très satisfaisants.

Le diagramme de Gantt

Afin de modéliser le diagramme de Gantt, nous avons choisi d'utiliser un outil en ligne permettant une manipulation rapide, claire et efficace pour notre travail.

Nous allons donc fournir de simples explications en annexe pour faciliter la navigation dans le diagramme ainsi que le lien d'accès.

Voici le lien pour le visualiser :

<https://app.agantty.com/#/sharing/35d9e4264b64620b27492a37fb14ab26>

PROGRAMMATION CONCURRENTE ET INTERFACES INTERACTIVES

CONCLUSION

Travail

Le travail fournit pour réaliser le développement était conséquent. Bien que répartie sur plusieurs séances. Certaines difficultés ont ralenti le projet.

Les plus importantes étant :

- la gestion du timer peu évidente avec les différences de temps, la conversion en secondes etc.
- la gestion de la perspective à incorporer en incluant le fait que les véhicules ennemis se déplacent aussi et que leur aspect doit changer en accordance.

Résultat

Sur un travail à deux personnes avec partage d'information et répartition des tâches, le développement a progressé de manière fluide. De solides bases en Java, algorithmique, et géométrique ont permis de mener le projet à bien sans encombre particuliers.

PROGRAMMATION CONCURRENTE ET INTERFACES INTERACTIVES

DOCUMENTATION

Utilisateur

L'utilisateur dispose des touches directionnelles pour déplacer son véhicule.

Le véhicule est reconnaissable car bien plus clair que les autres pouvant apparaître à l'écran.

Le but est de survivre le plus longtemps, éviter tout contact avec les arbres ou autres soucoupes volantes apparaissant à l'écran.

Attention toutefois à la vitesse qui est optimale au restant proche du centre de la piste noire affichée à l'écran. Toute sortie de piste, ou prise de hauteur entraîne une chute de vitesse.

Si il est compliqué de déterminer la zone de collision du véhicule ou encore la zone d'accélération de la piste : appuyer sur H pour afficher toutes les Hitbox.

Développeur

Il est possible de modifier la taille de la fenêtre en modifiant les valeurs de WIDTH, la largeur et HEIGHT, la hauteur, qui valent initialement les dimensions de l'écran et qui se trouvent dans la classe *Affichage*.

Dans la même classe, il est également possible de modifier la hauteur de l'horizon, donné dans la variable horHeight. On peut aussi mettre la valeur du booléen showHitBox à True, si l'on veut que les hitbox des différents éléments soient affichées par défaut.

Si l'on veut changer les images des différents éléments, on peut modifier les images ou en rajouter de nouvelles dans le dossier assets du projet. On peut ensuite charger ces nouvelles images en modifiant les images chargées dans les fonctions dont le nom commence par loadImg.

On peut modifier la largeur et la hauteur du véhicule avec les variables hitWidth et hitHeight de la classe Véhicule. Dans ce cas, il faudra également modifier les coordonnées x et y afin que le véhicule reste centré. Il faut aussi modifier les limites d'écran et les zones qui délimitent l'accélération du véhicule dans Avancer.

Il est possible de modifier le temps limite de départ avec la variable startTime de la classe Véhicule et on peut changer l'augmentation en modifiant la valeur donnée en paramètre à V.addTime(), à la ligne 120 de la classe Avancer.

Enfin, on peut modifier les dimensions des obstacles avec les variables w et h de la classe Obstacle et celles des adversaires avec les variables w et h de la classe Opponent.

PROGRAMMATION CONCURRENTE ET INTERFACES INTERACTIVES

ANNEXES : AGANTTY

The screenshot shows the Agantty Gantt Chart interface. At the top, there's a header with 'GANTT CHART' and 'DASHBOARD'. A date range 'FEBRUARY 2020' is selected. The main area displays a Gantt chart for a project named 'Moto Volante PCII' with tasks like 'Découverte du sujet', 'Rédaction du plan de développement', etc. A right sidebar lists 'Tasks/Milestones' with their details. Red numbers are overlaid on the interface:

- 1**: A red circle around the 'Add new project' button in the top-left.
- 2**: A red circle around the 'SEARCH' field in the top-left.
- 3**: A red circle around the date range selector at the top.
- 4**: A red circle around a task in the Gantt chart.
- 5**: A red circle around the zoom slider at the bottom of the Gantt chart.
- 6**: A red circle around the scroll bar on the right side of the interface.

MENU LATÉRAL GAUCHE

- 1 - Montrer/Masquer le point de vue général des tâches. (Celui dans lequel se trouve la zone **5**)
- 2 - Montrer/Masquer les tâches pour chaque collaborateur.

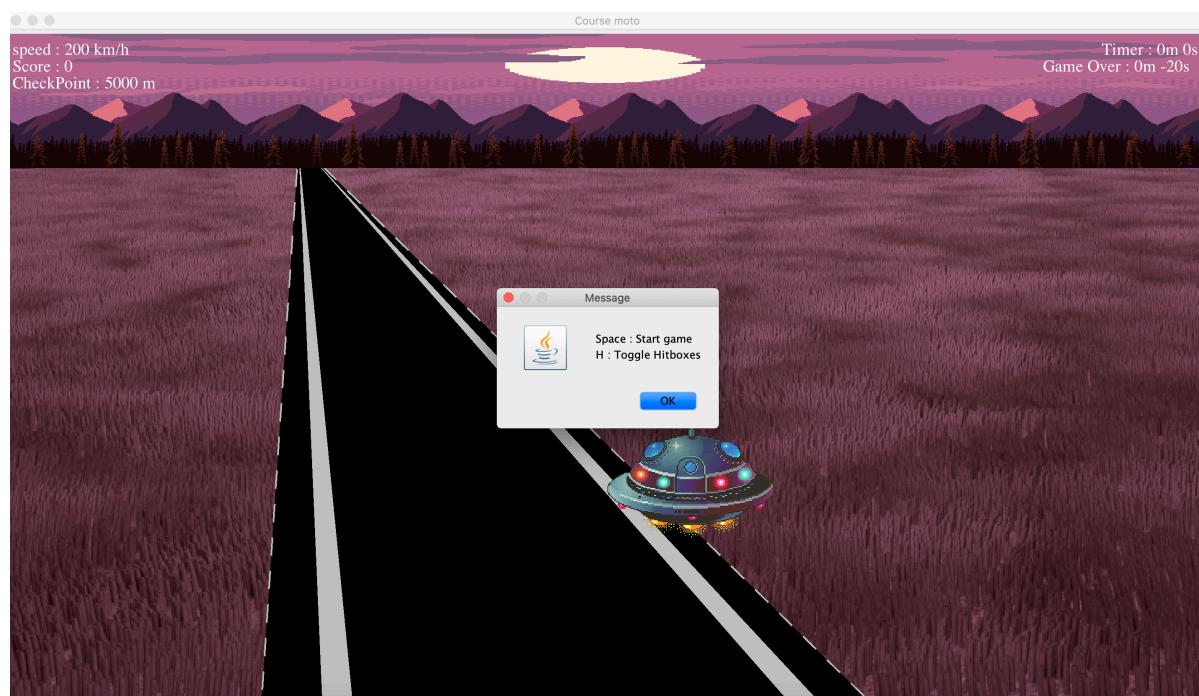
GÉNÉRAL

- 3 - Masquer les menus latéraux
- 4 - Appuyer sur une tâche montre ses informations dans le menu latéral droit. Survoler au moins 2 secondes avec la souris fera sortir une petite fenêtre pop Up avec les infos de la tâche.
- 5 - Permet d'ajuster le Zoom sur les jours / Les semaines.
- 6 - Étends la vue dans le calendrier. On utilise ensuite les barres coulissantes pour faire défiler la timeline.

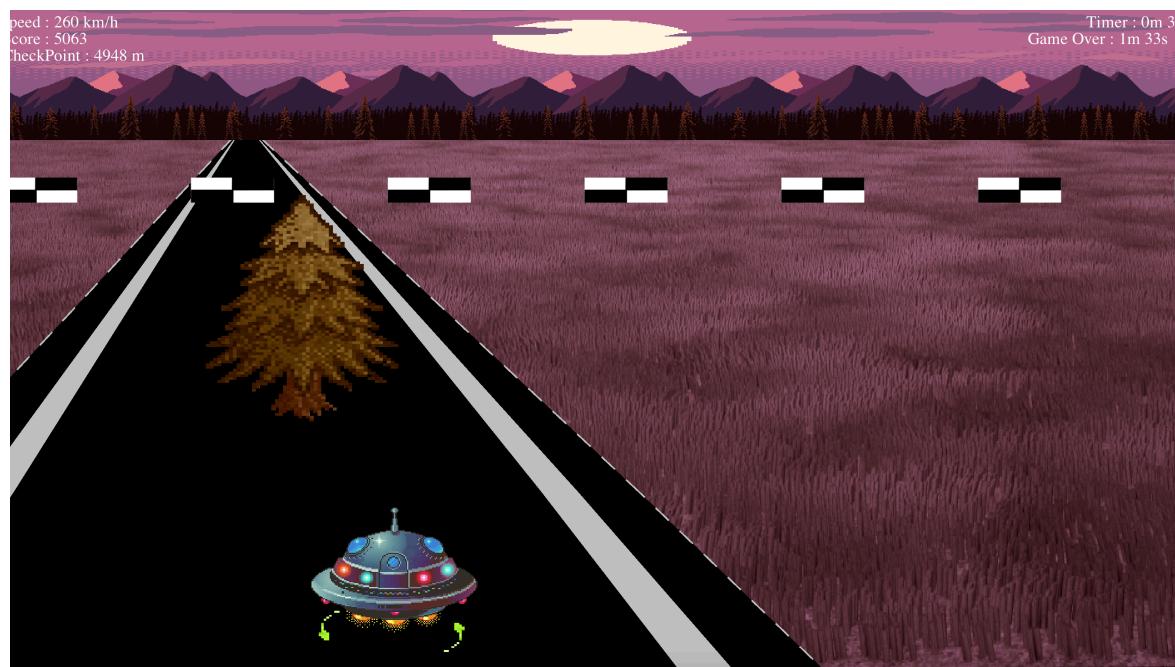
PROGRAMMATION CONCURRENTE ET INTERFACES INTERACTIVES

ANNEXES : LE JEU

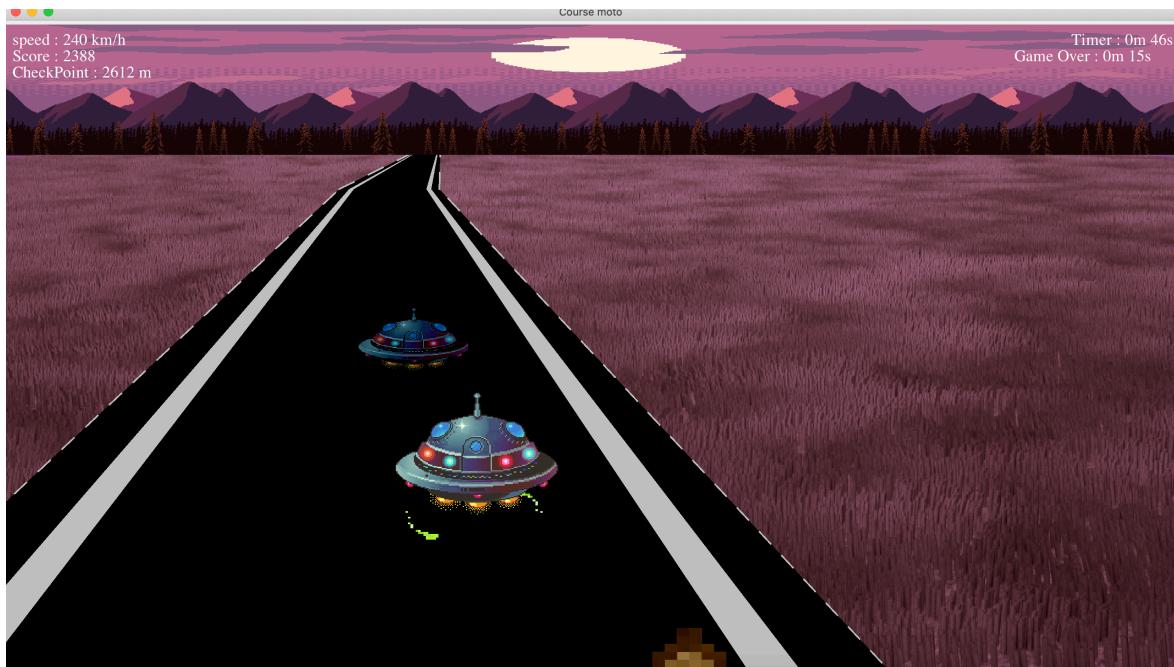
DEBUT DE PARTIE



OBSTACLE IMMOBILE : ARBRE & CHECKPOINT



OBSTACLE MOBILE : VAISSEAU ENNEMI



FIN DE PARTIE

