**Projet Java - Formula 0b1 - Livrable**

**Introduction**

Au cours de notre premier semestre de 1ère année en cycle ingénieur à l’ISEP, nous avons reçu des premiers cours de Java qui nous ont introduits à la méthode de programmation, au langage et à quelques algorithmes nous permettant de réaliser de petits programmes.

L’objectif final était de réaliser un jeu de course inspiré de l’application Android « Paper Racing ». Ce jeu de voiture a comme originalité de se jouer au tour par tour et de retranscrire une réelle prise en compte de l’inertie, le rendant plus difficile à jouer que ce que laissent penser les visuels.

Nous devions donc programmer un jeu similaire avec comme fonctionnalités importantes : les déplacements de la voiture (avec une bonne corrélation entre positon, vitesse et accélération) et la prise en compte des effets de l’herbe, de l’eau et des crashs.

Dans ce rapport, nous allons revenir sur notre projet et vous présenter le résultat final. Nous parlerons tout d’abord de son architecture puis nous reviendrons rapidement sur son design. Pour finir nous détaillerons ses fonctionnalités et les quelques améliorations que nous avons apportées par rapport au cahier des charges initial.

**Architecture**

Notre programme s’articule autour de 3 classes :

* ***Game*** qui contrôle la « façade » du jeu et son lancement, ainsi que les différents affichages au niveau de la console.
* ***Land*** qui détaille le terrain, son affichage et les différentes caractéristiques de celui-ci, que ce soit les effets de l’herbe, de l’eau, des crashs mais aussi notre fonction « téléporteur » que nous détaillerons plus tard.
* ***Car*** qui gère l’affichage de la voiture, son contrôle et ses mouvements

Nous avons essayé d’ajouter un mode 2ème joueur à l’aide d’une 4ème classe « Car 2 » mais comme le résultat n’a pas été probant, nous n’en parlerons pas dans ce rapport.

La classe Game comporte comme fonction principale « play() » qui va lancer le jeu et permettre de facilement rejouer une partie sans avoir à relancer tout le programme. Nous avons aussi une fonction console() qui va afficher tous les messages dans la console du logiciel.

Comme nous en parlerons dans la section « Design » nous avons fait comme choix d’utiliser la console du logiciel comme interface vis-à-vis du joueur. Cela était plus simple pour nous, mais permettait aussi de profiter pleinement de l’affichage pour notre carte.

La classe Land comporte de nombreuses fonctions similaires et nous pouvons les décomposer en trois catégories : la lecture du fichier .txt représentant la carte, la texture du terrain ainsi que ses effets sur la voiture du joueur.

Notre programme intégrant plusieurs cartes, nous utilisons une variable String « s » qui va permettre d’aller appeler différents fichiers (grid1.txt, grid2.txt etc.) facilement en fonction des choix du joueur. Puis, la fonction fieldChoice() reconstitue un tableau de caractères pour nous permettre de nous repérer facilement sur les différents caractères utilisés dans la carte (un espace, un point, et différentes lettres : o pour l’eau etc.).

Notre fonction texture() va parcourir le tableau de caractères et pour chaque signe, elle associe une image. Cela nous permet de pouvoir rapidement rajouter une fonction ou bien de change une image.

Enfin, nous avons ajouté tout un tas de fonctions permettant de controler les effets du terrains : grass(), water(), crash() mais aussi start() et end() qui vont permettre de placer la voiture sur la ligne de départ et de détecter la fin de la course (et de relancer le jeu en rappelant la fonction play() décrite plus haut).

A chaque fois, nous réutilisions la même méthode : le tableau de caractères « décompo » est parcouru et si le symbole est détecté et si la position en x et y de la voiture se trouve au niveau de ce symbole, différents effets ont lieu : la voiture est ralentie sur l’herbe, son accélération est nulle sur l’eau et un crash provoque sa remise sur le terrain avec une pénalité de temps (nous y reviendrons plus tard).

La classe Car permet de contrôler la position de la voiture, sa vitesse et son accélération, et à chaque fois selon les x ou les y. Nous y retrouvons aussi la fonction carDisplay() qui va afficher la voiture avec un angle différent en fonction de sa vitesse en x et en y. Là aussi, nous avons utilisé une string « c » pour changer facilement la couleur de la voiture (rouge, bleu ou vert).

Pour faciliter le code, c’est aussi dans cette classe que nous trouvons circle() qui affiche, comme dans l’application Android, les 9 futures positions possibles.

La fonction acceleration() relie le jeu au clavier grâce à la commande isKeyPressed. Pour chaque touche appuyée, l’accélération est modifiée pour l’augmenter, la diminuer, ou bien garder la même.

La fonction position() contient tout un ensemble d’appel aux fonctions de Land comme end(), crahs(), ou encore texture() pour permettre de contrôler tout cela à tout moment lors du déplacement de la voiture. Afin de percevoir correctement les déplacements du véhicule, nous réitérons ces appels 10 fois (en faisant augmenter la position en x ou y à chaque fois d’un dixième de la vitesse Vx ou Vy).

L’architecture de notre programme n’est donc pas particulièrement compliquée mais est remplie de connexions entre nos classes, un peu trop sans doute, c’est ce qui a compliqué la tache lorsque nous avons voulu ajouter un deuxième joueur.

**Design**

Deux choix s’offraient à nous pour l’interface graphique de notre projet : StdDraw, que nous utilisons depuis nos premières séances de Java ou bien Swing qui nous semblait bien plus riche en possibilités mais qui nous était totalement inconnu.

C’est pour cette raison que nous avons fait le choix de rester sur StdDraw, au risque d’être limité mais cela nous permettait de travailler dans un environnement que nous maitrisions déjà.

Cela a pour conséquence de rendre notre jeu un peu moins beau visuellement avec une interface très « carrée », surtout au niveau de la texture du terrain. Mais cela reste simple et efficace pour pouvoir modifier, améliorer, les éléments de notre jeu.

Nous avons fait le choix d’agrandir au maximum la fenêtre de StdDraw afin d’avoir une taille de voiture raisonnable par rapport au reste du terrain tout en pouvant définir des terrains assez grands. Nos terrains font aussi tous la même taille afin de simplifier la méthode pour changer de terrain.

**Fonctionnalités**

Au cours de ce projet nous avons réussi à implémenter de multiples fonctionnalités sur ce programme. De nombreuses sont des ajouts par rapport au cahier des charges défini par nos professeurs.

Comme prévu, le jeu prend bien en charge l’inertie de la voiture pour modifier ses positions. Le tour par tour fonctionne de manière quasiment transparente. La voiture se contrôle soit avec les touches a,z,e,q,s,d,w,x,c soit par l’intermédiaire du pavé numérique. Nous y avons ajouté cette deuxième commande pour être compatible avec tous les ordinateurs sur lesquels nous étions amenés à travailler mais aussi pour préparer l’arrivée d’un mode « Multi-joueurs ».

De plus, les 9 prochaines positions possibles s’affichent pour permettre au joueur de faire son choix de prochaine action en toute connaissance de cause concernant les risques de crash ou autre. Le jeu dispose aussi d’un compteur qui s’incrémente chaque tour d’une seconde.

L’herbe, après de nombreuses difficultés, réduit bien la vitesse en x et y par 1 / -1 si celle-ci est supérieure / inférieure à ces valeurs.

Sur l’eau, l’accélération est nulle. Et nous sommes parvenus à modifier l’affichage des 9 prochaines positions pour n’afficher que la position centrale correspondant à celle où la vitesse reste constante. Idem pour les commandes qui sont alors réellement limités à la touche centrale.

Les crashs ont été difficiles à réaliser mais nous sommes parvenus à un résultat proche de la perfection avec une pénalité immédiate sans prendre le risque de voir la voiture sortir du terrain en profitant d’un bug (nous avons notamment eu du mal avec les déplacements en diagonale). Pour cela, nous avons ajouté un cadre tout autour du terrain, et dédoublé nos obstacles en utilisant 4 caractères différents mais une seule image pour que le programme puisse détecter si la voiture se « crash » par le bas, le haut, la gauche ou la droite sans pour autant modifier l’affichage au niveau du joueur.

Lors d’un crash, le véhicule est replacé juste avant le mur, sa vitesse en x et en y est ajoutée sous formes de pénalités au temps du jeu avant d’être remises à zéro pour ne pas avantager le joueur « casse-cou » puisque l’objectif est de finir une carte le plus rapidement possible.

En parlant des cartes, 5 sont incluses dans notre version avec une première carte « de test » qui permet essentiellement de découvrir les différentes fonctions du jeu. D’autres sont décorées avec les éléments à notre disposition ou bien mettent en avant des bonus que nous avons incorporés.

Notre « touche personnelle » est dans l’incorporation de téléporteurs afin de diversifier l’expérience et l’appréhension des cartes qui ne se limitent plus forcément à un couloir jonché d’obstacles. Il existe deux types de téléporteurs : l’un est aléatoire et va vous replacer n’importe où, au hasard, sur la carte. A vous donc de prendre le risque de gagner un temps considérable ou bien de devoir tout recommencer. L’autre fonctionne par paire et associe une entrée à une sortie placée spécifiquement quelque part sur le terrain. Les deux types avant la même apparence, le risque est double !

En sortie de téléporteur, nous avons choisi de conserver l’inertie de notre véhicule, afin de rendre hommage au jeu Portal, mais aussi au cas où la voiture aurait le malheur de se retrouver sur l’eau, qu’elle puisse se déplacer même avec une accélération nulle.

Le « menu » de notre jeu se situe au niveau de la console. Au lancement du programme, après un message d’accueil, le jeu vous invite à choisir une carte ainsi que la couleur de votre véhicule. Trois couleurs sont disponibles : rouge, bleu et vert (et cela change également la couleur des cercles de positions). Puis, le jeu affiche à chaque fin de tour la position (x/y), la vitesse (Vx/Vy) et l’accélération (Ax, Ay) du véhicule ainsi que les différentes conséquences de votre position : un message d’avertissement pour votre présence sur l’herbe ou l’eau ou bien l’affichage de votre pénalité lors d’un crash qui s’ajoute au temps qui s’affiche à chaque fin de tour également.

Lorsque la voiture franchie (ou plutôt atteint) la ligne d’arrivée, un message de félicitations vous est adressé, accompagné de votre temps de parcours (qui correspond en quelque sorte au score) puis le programme vous demande si vous souhaitez rejouer ou non.

Nous avons donc ajouté en plus des fonctions de base, de nombreuses améliorations et bonus au niveau du déroulement du jeu pour rendre le tout plus agréable à jouer.

**Conclusion**

A la fin de ce projet Java, nous sommes parvenus à avoir un programme qui fonctionne, qui propose 5 cartes, plusieurs couleurs de véhicules, un menu complet au niveau de la console du logiciel ainsi que des petits bonus inédits dans notre groupe.

Cela nous a permis de prendre confiance en nous en réalisant un projet complet et réellement pratique avec un vrai jeu de course dans un affichage sympathique (même si perfectible).

Notre seul regret est de ne pas être parvenus à incorporer un mode multi-joueurs qui semblait à notre portée. Mais cela nous a permis d’ajouter d’autres fonctions à notre programme.