|  |  |
| --- | --- |
|  | **Une image contenant texte  Description générée automatiquementDépartement d’informatique**  **Programmation Réalité Virtuelle**  **420-425-RI Groupe 02** |

**Informatique 420.00**

**Laboratoire 2**

**Course à obstacle**

**Remis à**

**Dave Lacourse**

**Présenté par**

**Fannie Hamel Thibault**

**24/03/2023**

Les vrais aventures vraies de Simon-Olivier Soucis

# Synopsis

Le Capitaine Simon-Olivier Soucis cherche à rentrer au bercail après plusieurs semaines d’aventure ! Ramène le chez lui en déplaçant son avion jusqu’à l’objectif brillant des trois niveaux du jeu. Ne t’égare pas dans les nuages et fait attention aux obstacles !

# Présentation des niveaux

Tous les niveaux ont le même canevas de base pour le terrain de jeu, soit un plane 10x10x10 nommé ‘’Sol’’ transparent servant de terrain et un mur carré formé du prefab Cloud-B du package Free Aircraft Pack (FAP) d’Understone. Du même package, le jeu utilise le prefab Aircraft-A-A comme élément ‘’Player’’. Les skyboxes proviennent toutes du package Free Stylized Skybox (nommé Ciel dans la librairie des assets).

Une collision avec un mur extérieur du jeu entraîne la perte de 2 points et ramène le joueur à son point de départ pour le niveau en cours. Une collision avec n’importe quel obstacle intérieur du jeu n’impacte pas la position du joueur, mais lui fait perdre 1 point.

Pour gagner un niveau et passer au niveau suivant, il faut entrer dans la zone de l’objectif brillant, ce qui fait gagner au joueur 10 points. L’objectif est représenté visuellement par l’effet de particule CFX\_Magical\_Source du package JMO Assets (JMO). Un message console affiche alors le nombre de collisions et le pointage actuel ainsi que le temps qu’il a fallu pour passer le niveau.

### Pistes d’amélioration

Dans le futur, il serait intéressant d’afficher des données plus précises par rapport aux différents niveaux. Par exemple, il faudrait qu’en plus du temps qu’il a fallu au joueur pour passer un niveau, il faudrait aussi afficher le nombre de collisions et le pointage propre au niveau. Cela serait facilement réalisable par une méthode similaire que pour la sauvegarde du temps.

Il est tout à fait possible de soigner davantage l’apparence de mon jeu en passant plus de temps à travailler dessus. Graphiquement, il est loin d’être ‘’terminé’’.

## Niveau 1 : La zone militarisée

*« Alerte S.O.S. ! Vous avez pénétré une zone militarisée en plein conflit ! Évitez les projectiles pour atteindre l’objectif le plus rapidement possible. »*

Dans ce premier niveau, jusqu’utilise le prefab Cloud-A de FAP pour faire deux obstacles immobiles dans mon terrain de jeu.

Le même prefab sert aussi de base pour trois cannons qui sont des obstacles potentiels, mais qui servent surtout à tirer des boulets qui sont des obstacles en mouvement. Lorsqu’ils tirent un boulet, les cannons instancient également l’effet de particule CFX\_Explosion\_B\_Smoke+Text de JMO. L’un de ces cannons fait un mouvement de va et vient sur l’axe des z et ses projectiles sont tirés à partir de sa position actuelle.

### Pistes d’amélioration

Originalement, les trois cannons devaient bougés, mais la décision fut prise de n’en faire bouger qu’un seul pour une question de balance. Cependant, peut-être qu’en faire bouger un deuxième rendrait le niveau plus attirant visuellement. Il est très simple de faire bouger les nuages dans la direction voulu en reprenant le code du script Mouvement OE et en modifiant le vecteur v.

Au moment de rendre le travail, les cannons ne tirent pas dans le bon sens, ce qui rend leur présence caduque. Il m’a fallu beaucoup de temps pour comprendre comment réutiliser la matière de mon cours d’Algèbre linéaire pour faire bouger quelque chose dans le sens que je le souhaite. Après avoir fait le script pour mes nuages en mouvement à l’aide d’un tutoriel sur internet, j’ai compris un peu mieux l’utilisation des vecteurs. La solution serait probablement donc de donner des noms individuels à chacun de mes cannons, de les mettre dans une liste et de les faire tirer dans une direction propre à un vecteur que je leur aurai créé individuellement.

## Niveau 2 : La zone de turbulence

*« Catastrophe S.O.S. ! En vous échappant de la zone militarisée, vous avez pénétré une zone de turbulence ! Même si votre vision est moins bonne, vous devez trouver une façon d’éviter les éléments pour vous mettre à l’abris dans l’œil de la tempête. »*

Dans ce deuxième niveau, le joueur doit manœuvrer l’avion jusqu’à l’objectif dans un parcours très simple en forme de M tout en évitant d’entrer en collision avec les murs. Son chemin est complexifié par la présence de petites tornades qui apparaissent et disparaissent aux 2.3 secondes ainsi que par sa perspective qui lui permet difficilement de voir sa position par rapport aux obstacles. Les tornades sont fait du prefab Cloud-A de FAP ainsi que des effets de particule CFXR3 Shield Leaves, CFX\_Tornado et CFX\_ElectricityBall de JMO. Ils bougent très légèrement de bas en haut pour le plaisir des yeux.

### Pistes d’amélioration

Je suis honnêtement assez satisfaite de ce niveau ! Je crois que si j’avais à l’améliorer, je ferais en sorte que mes tornades apparaissent et disparaissent à des fréquences différentes pour rendre le visuel plus intéressant.

## Niveau 3 : L’œil de la tempête

*« O.K. S.O.S. ! Vous êtes à présent dans l’œil de la tempête. Nous vous avons envoyé des coordonnées très précises afin de vous sortir de là : touchez les quatre nuages d’or pour remporter la victoire, mais attention à ne pas vous faire prendre par la tempête ! »*

Dans ce troisième niveau, le joueur doit toucher les quatre nuages d’or pour terminer le jeu. Les nuages sont fait avec le prefab-A de FAP. Au centre, une roue composée d’un cylindre au centre et de deux cubes tourne et force le joueur à se déplacer à un certain rythme. S’il rate l’objectif pendant qu’il est dans la zone propice pour le toucher, il devra attendre que la roue tourne assez pour lui permettre d’atteindre sa cible à nouveau.

### Pistes d’amélioration

Je suis aussi assez heureuse de ce niveau dans l’ensemble. Je crois qu’une zone de jeu plus grande aurait permis de rajouter des obstacles supplémentaires au parcours et ainsi offrir un niveau final mixant les pièges des deux précédents niveaux.

## Lien GitHub