Rapport - Itération 02	Équipe 04	14/03/2024
ST-HILAIRE, Jean-Félix	STHJ09099500	
BOMPARD Noah	BOMN73360101	
TREMBLAY, Anthony	TREA17099702	
DUCROS, Vincent	DUCV90370101	

Wacky Wheels Riot

Liste des fonctionnalités implantées

Fonctionnalité	Commit(s)	Responsable(s)	Fichier(s)
Intégration de la carte dans HUD	4885cbb 62ffe06	Jean-Félix	Scenes/MainScene.unity Material/HUD/MatMaps.mat Material/HUD/map-hud-fill.png Textures/UI/MapTexture.renderTexture Material/MapsTexture.mat ../cameraFollow.cs
IA voitures adverses avec apprentissage par renforcement et réseau neuronal	0765f8e	Vincent	Prefabs/KartClassic/Audi_Quattro_IA.prefab Scripts/AI/KartAgent.cs Prefabs/AI/kart-999961.onnx Prefabs/AI/kart-better.onnx
Sauvegarde des stats (temps circuit)	e0006df 1c9e136	Anthony	Scripts/GameModes/LapCheckpoint.cs Scripts/GameModes/LapObject
Amélioration mécanique de drift + des VFX liés	8d89611 et ddef35a	Noah	Prefabs/VFX/DirectionalLight.prefab Art/Materials/… et Art/PhysicsMaterials/ Scripts/KartSystems/ArcadeKart.cs
Amélioration orientation caméra quand on drift	ddef35a	Noah	Scenes/PhysicsPlayground.unity -> Paramètres de la CinemachineVirtualCam
Support manette	4885cbb	Jean-Félix	Scenes/IntroMenu.unity Scenes/LoseScene.unity Scenes/WinScene.unity Scenes/MainScene.unity
ajouter les fonctionnalités de modification du son et de la résolution dans les options	4885cbb 3f9c3f2 552039e	Jean-Félix	Scripts/UI/VolumeSliders.cs Scenes/MainScene.unity Scenes/IntroMenu.unity Audio/MainAudioMixer.mixer

			Scripts/UI/InGameMenuManager.cs Scripts/UI/TakeScreenshot.cs Scripts/UI/WindowDropdownScript.cs Prefabs/UI/InGameMenu.prefab
Effet de screenshake lors d'une collision avec une IA	03241aa	Vincent	Scripts/VFX/CameraShakeManager.cs Scripts/KartSystems/ArcadeKart.cs

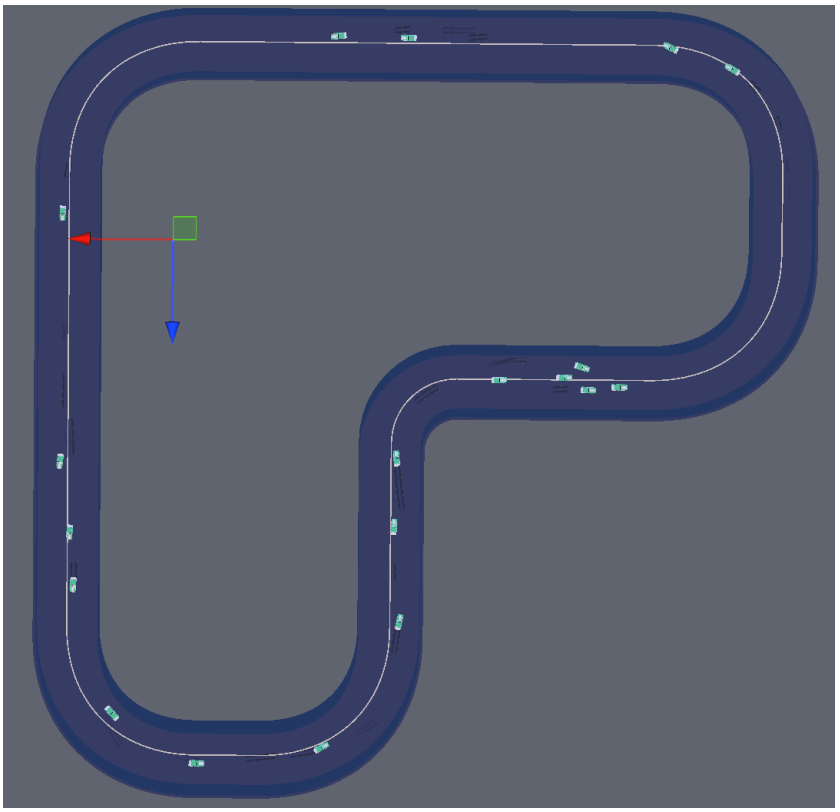
*** Toutes les classes sont situées au début de l'arborescence "Assets/Karting"*

Description détaillée des fonctionnalités

Intégration de la carte dans le HUD

Nous avons intégré une carte en haut à gauche de l'écran de jeu. Cette carte consiste en une caméra suivant le joueur, capturant une zone étroite et rapprochée de la piste depuis le haut. Pour afficher la carte avec des coins arrondis, nous avons utilisé un plugin (https://github.com/kirevdokimov/Unity-UI-Rounded-Corners) facile à utiliser. Nous avons tout de même conservé le cercle gris et noir qui servait de substitut en attendant l'implémentation de la caméra. Celui-ci est utilisé pour mieux différencier la carte de la vue du joueur, en ajoutant un contour subtil.

AI voitures adverse avec réseau neuronal



L'entraînement des IA a été réalisé sur une map de test (pour éviter un surentraînement spécifique à une map) sur 30 minutes (pas assez pour obtenir un comportement correct avec les paramètres de reward actuels, mais suffisant pour démontrer la fonctionnalité). On utilise l'API Python ML-Agents fourni par Unity pour faire l'entraînement.
Ce dernier consiste à lancer les IA avec le script KartAgent en mode **Training** pour les faire spawner à différents checkpoints répartis tout au long de la carte. Si elles touchent un mur ou sortent de la piste, elles sont détruites et recommencent. Grâce aux senseurs placés sur chaque agent (des raycasts), ils peuvent prendre des décisions éclairées sur les inputs à appliquer pour maximiser leur récompense.
Le système de récompense sont les suivants pour les 2 modèles .onnx générés :

Nom de la récompense	Valeur pour kart-999961.onnx	Valeur pour kart-better.onnx
Hit Penalty	-1	-1.5
Pass checkpoint reward	1	3
Towards checkpoint reward	0.03	0.04
Speed Reward	0.02	0.05

On lance l'entraînement via :

```
$ mlagents-learn Prefabs\AI\kart_mg_trainer_config.yaml --run-id=KartAI
```

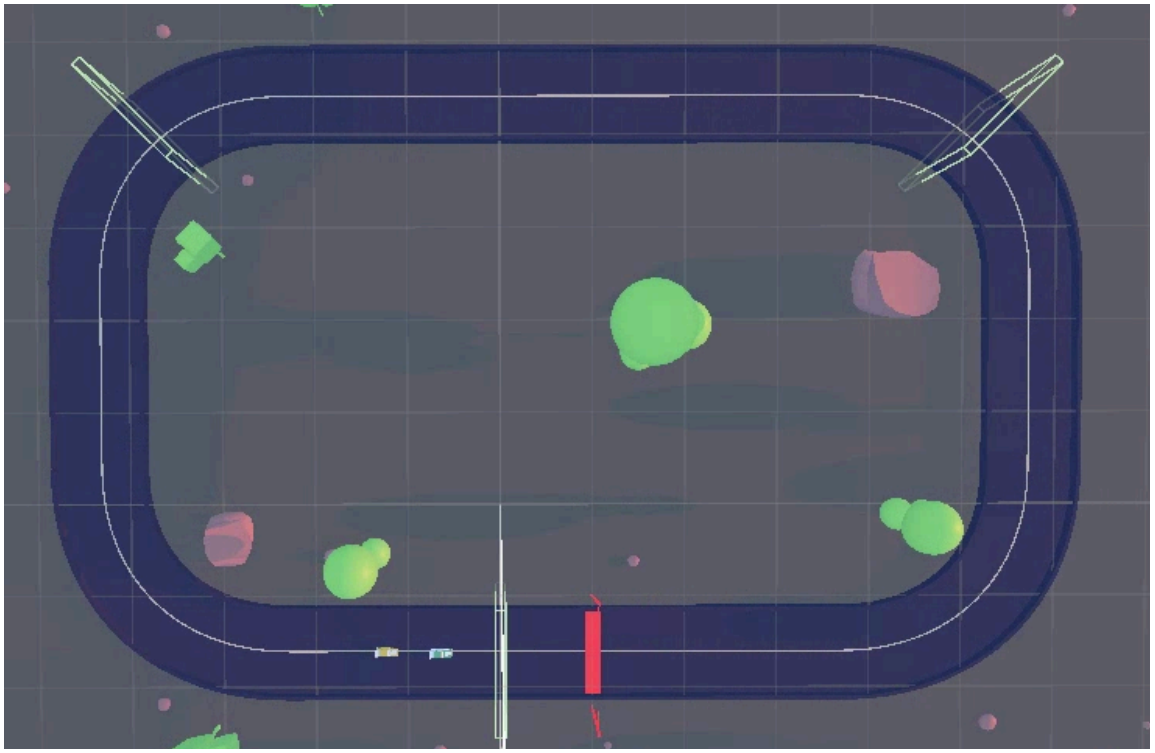
Ensuite, tous les resultats sont compilés dans un dossier **results** à la racine du projet. On peut avoir une représentation visuelle de l'entraînement (loss value, reward…) en lançant la commande suivante :

```
$ tensorboard --logdir results
```

Depuis ce dossier résultat, on peut récupérer le modèle .onnx généré et l'intégrer au projet, en le passant au composant Behaviour Parameters de l'agent et en remettant le script KartAgent en mode inférence.

Sauvegarde des *stats* (temps d'un tour)

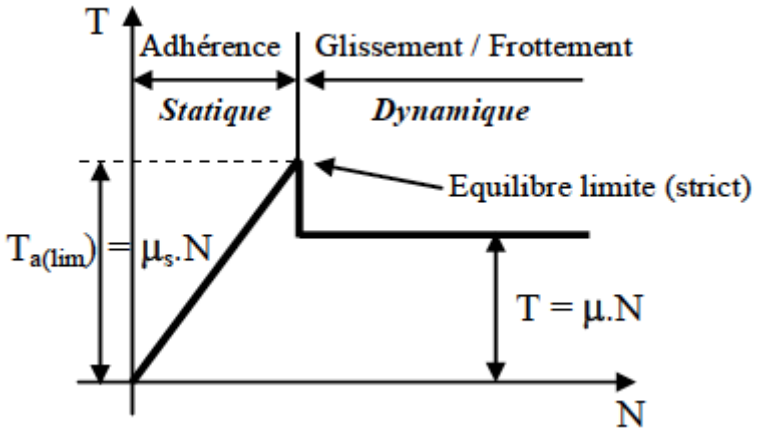
Le système de jeu basé sur des tours était presque entièrement mis en place. Nous avons jugé crucial d'intégrer un système de vérification afin d'empêcher les joueurs de prendre des raccourcis illégaux. Ainsi, nous avons ajouté des "checkpoints" que les joueurs doivent franchir pour que leur tour soit validé. Dans l'image ci-dessous, vous pouvez voir les trois "checkpoints" pour le circuit de départ. Le nombre de "checkpoints" peut varier selon les circuits.



Amélioration mécanique de drift + des VFX liés

Le drift

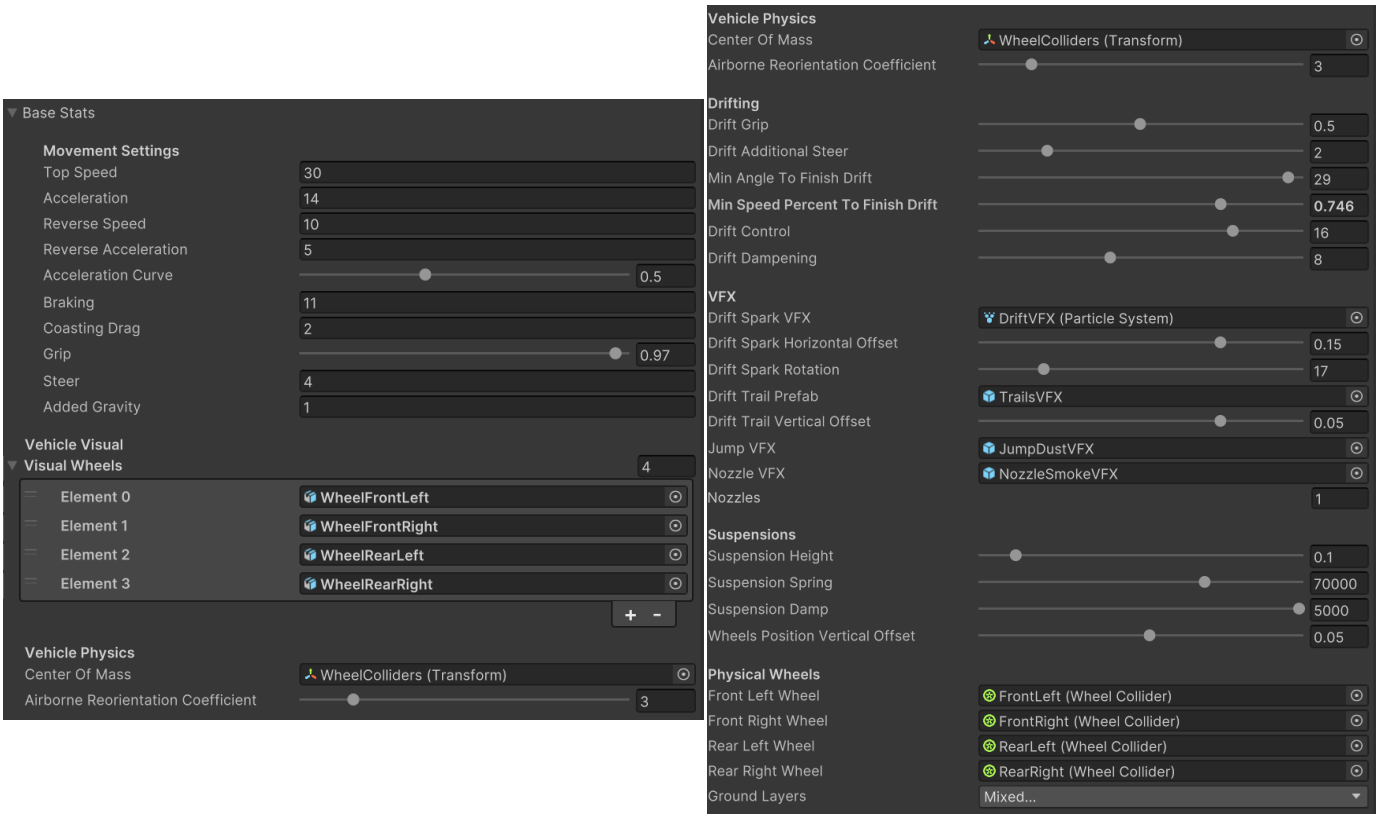
Pour améliorer la mécanique de drift, il faut la comprendre mathématiquement,



source

Globalement, ce qu'explique la loi de coulomb (la loi sur l'adhérence des matériaux secs) c'est que pour qu'une voiture drifte, il faut une vitesse minimale, qui permet de créer une friction suffisamment forte pour rentrer dans le "dynamique".
Une fois qu'on drifte, on peut aller moins vite, on continuera a drifter selon l'angle de la voiture par rapport au vecteur vitesse.

Et ça se calcule, de plus, selon chaque voiture, son poids, sa maniabilité (capacité à tourner), son accélération, on va avoir un 'point de drift' différent, d'où la nécessité d'introduire des paramètres modifiables pour personnaliser les voitures que va pouvoir conduire le joueur.



Exemple de réglages effectués pour l'Audi Quattro.

Exemples de drifts, ici on a un angle largement inférieur à celui "limite" qui termine le drift et pourtant on drifte, c'est le résultat de l'inertie sur la voiture (et la source de l'aqua planning, des accidents, etc…).

Les VFX liés

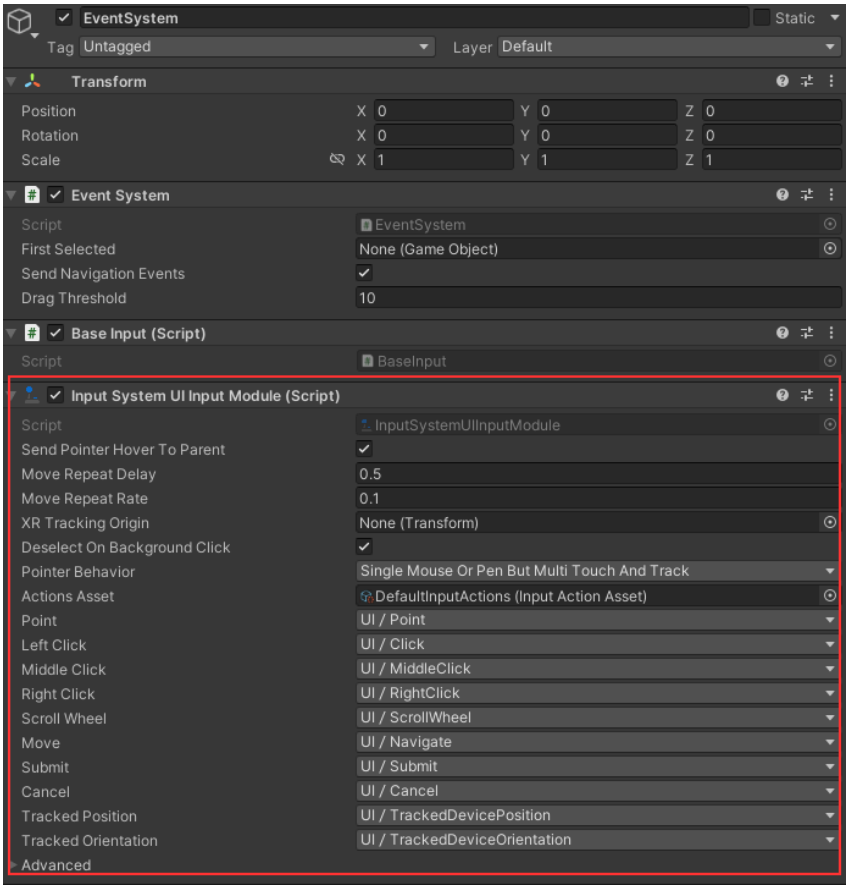
En plus pour donner un effet de drift, plus arcade et notifier au joueur qu'il drifte, on a amélioré le système de particules pour en faire un plus visible et plus ‘Mario Kart’ like (j’ai reproduit avec unity les émetteurs de particules de Mario kart)

particules quand on drifte / bruits du moteur

En plus de ça, on ajoute un bruit de drift, le bruit du moteur et un bruit différent quand il recule.

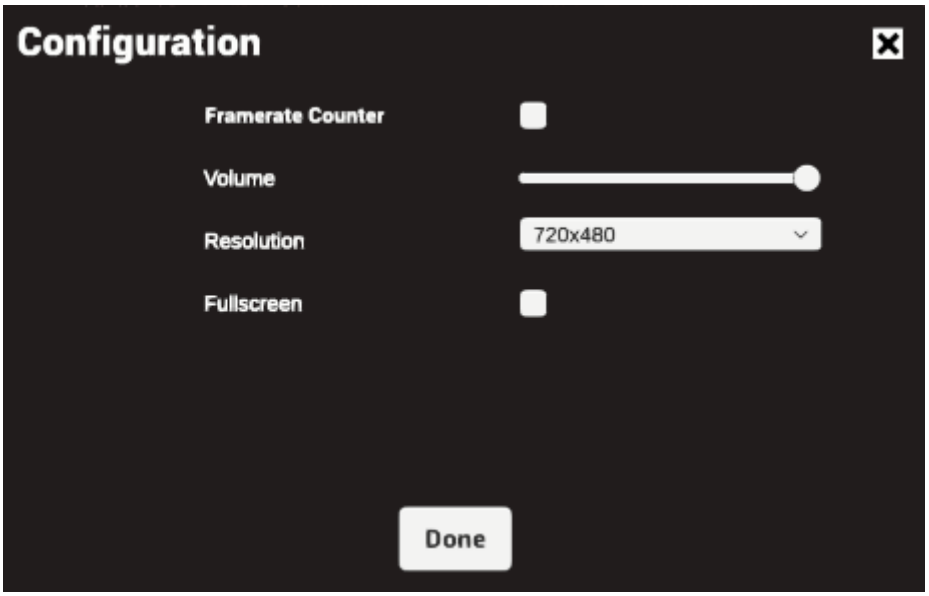
Support manette

Pour le support des manettes, nous avons simplement utilisé un *plugin* plus performant que celui de base d'Unity, nommé simplement "*Input System*". Ce plugin nous permet d'assigner facilement des actions aux touches du clavier, de la souris, des manettes ou de tout autre dispositif permettant une entrée (par exemple, il existe également des actions pour les casques de réalité virtuelle/augmentée). Étant donné que nous utilisons des fenêtres superposées pour les menus, une modification de l'index du curseur était nécessaire pour le faire fonctionner correctement.



Ajout de fonctionnalités dans le menu (volume et résolution)

Dans les menus (tant dans la scène principale que dans le menu contextuel pendant une partie), nous avons effectué des ajustements pour faciliter la navigation. Dans le menu principal, nous avons séparé le bouton "Configuration" du bouton "Contrôles", dans le but de simplifier l'expérience utilisateur. Nous avons supprimé l'option de capture d'écran, car elle n'était pas requise et prenait de l'espace inutilement. De plus, nous avons retiré le basculement de l'ombrage, car nous avons jugé qu'il s'agissait d'une fonctionnalité peu pertinente à implémenter. La seule fonctionnalité restante du menu de base est le compteur de framerate. En outre, nous avons ajouté une barre de défilement (slider) pour le son, un menu déroulant (dropdown) pour le choix de la résolution et un autre basculement pour l'option du mode plein écran. Le menu contextuel d'une partie est identique à celui du menu principal (il utilise le même code pour les modifications), à l'exception d'un bouton supplémentaire permettant d'afficher les contrôles du jeu (ce que nous n'avons pas besoin de faire dans le menu principal car le bouton correspondant y est déjà présent).

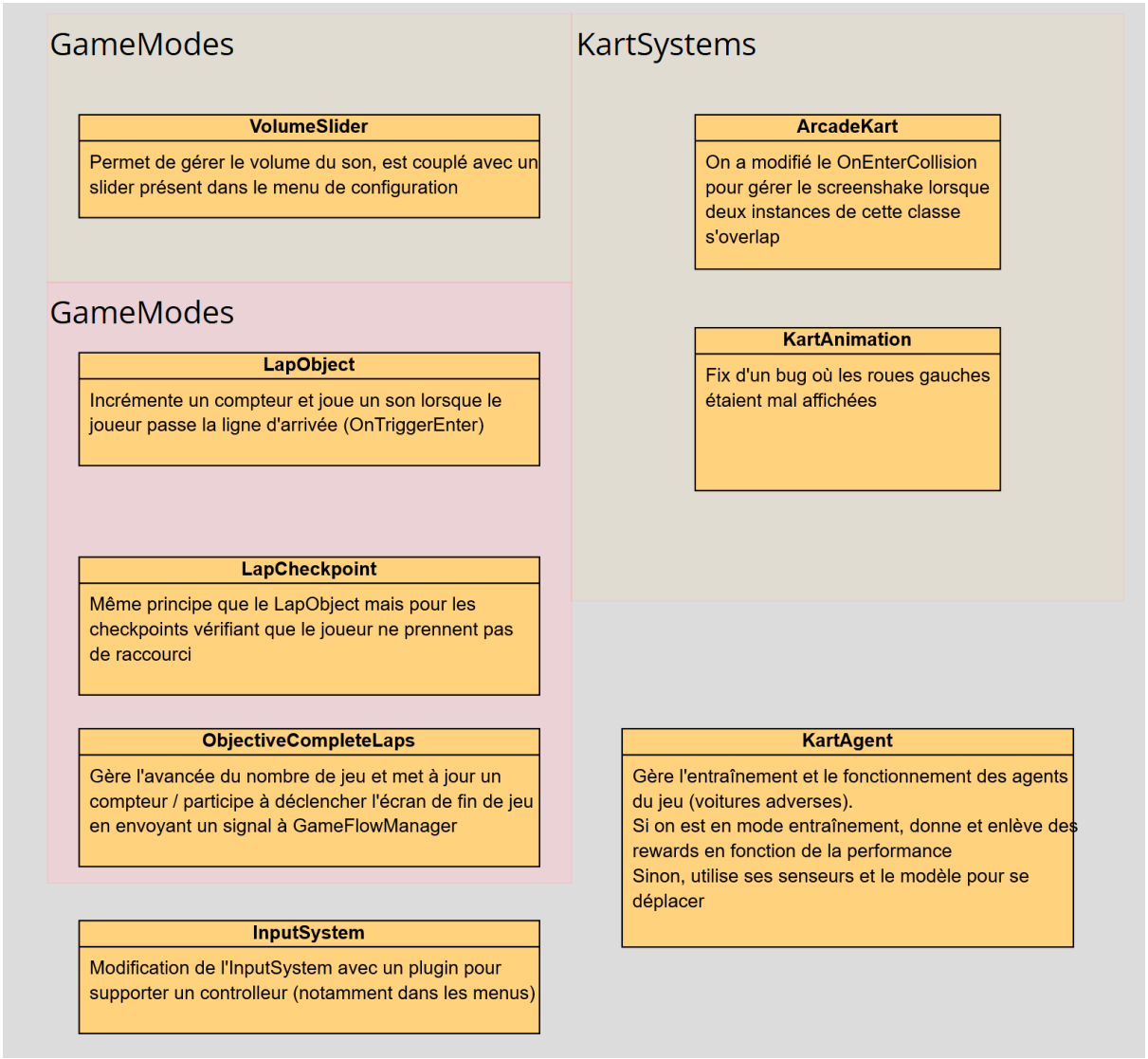


Effet de screenshake lors d'une collision avec une autre voiture

On ajoute à la CinemachineVirtualCamera un listener pour être mis au courant de CinemachineImpulsion et faire bouger la caméra. Dans les agents, on ajoute un composant permettant d'invoquer une impulsion cinémachine où l'on paramètre l'effet obtenu. Enfin, on place dans la scène un CameraShakeManager qui va générer une nouvelle instance de la vibration lorsque l'événement OnCollisionEnter est déclenché dans le code de ArcadeKart et qu'on est bien en collision avec le joueur.

La force de la vibration va être déterminée par la vitesse du joueur, ainsi plus on va vite, plus l'écran va trembler.

Diagramme de classes



Description des défis techniques

L'apprentissage par renforcement a apporté son lot de défi, notamment sur le temps nécessaire pour entraîner un modèle, et le choix des paramètres de reward et des hyperparamètres pour créer un comportement réaliste et fun à affronter. Pour l'instant, nous sommes assez loin d'avoir un résultat convainquant, donc nous allons poursuivre nos recherches en s'inspirant de modèles utilisées avec succès dans d'autres jeux de course. Une solution de secours serait d'implémenter un système de waypoints simple et d'avoir des agents qui essayent de se diriger en suivant ces derniers.

L'intégration du code a aussi été un peu plus compliqué qu'avant, heureusement, Unity propose un outil de merge intelligent pour résoudre les conflits de scène. Il suffit de paramétrer le fichier config dans le dossier .git puis d'appeler :
`$ git mergetool`
après avoir lancer un merge.

Fonctionnalités - itération suivante

Fonctionnalité	Responsable	Catégorie	Points
Améliorer le fonctionnement des agents intelligents, notamment la gestion des collisions	Jean-Félix & Vincent	IA	4
Implémenter le système qui track la position du joueur par rapport aux IA	Vincent	IA / UI	4
Créer la carte de course finale	Noah	Environnement	4
Harmonisé le design du jeu (comprenant, mais ne se limitant pas à : HUD, menus, texte).	Jean-Félix	UI	1
Ajout des objets de boost	Anthony	Gameplay	2
Ajoute de la personnalisation de la peinture du véhicule	Anthony	Gameplay	2
Système de sélection du véhicule et de la carte	Noah	Gameplay / UI	3
Système de sauvegarde (des paramètres, des véhicules et peintures débloqués)	Vincent	Système	3
Ajouter (avec le bouton "Score" dans le menu	Jean-Félix	UI	3

principal) un tableau de score			
Total			26

Bibliographie

https://assetstore.unity.com/ : Asset Store d'Unity d'où viennent nos préfabs

https://kenney.nl/tools/asset-forge : Asset Forge est un logiciel de modélisation 3D dont on s'est servi pour concevoir le modèle de la voiture.

https://www.youtube.com/watch?v=CgyLIWyDXqo
: Tutoriel pour l'ajout de l'effet de screenshake

https://github.com/Unity-Technologies/ml-agents
Page GitHub du projet de l'API Python ML Agents pour l'apprentissage par renforcement

https://docs.unity3d.com/Manual/SmartMerge.html
Page de la documentation de Unity sur l'outil de merge intelligent.

https://public.iutenligne.net/mecanique/mecanique-du-solide/charbonnieras/mecanique/132_lois_du_frottement_sec_lois_de_coulomb.html
Image expliquant la loi de Coulomb