Rapport : Projet Usine du futur

Bongiovanni Arthur, Joseph Wilkens, Vanicotte–Hochman Alexandre, Withanage Perera Sakun

1. Description de la fonctionnalité

Objectif global

L'objectif est d'ajouter une dimension stratégique au jeu **Super Jumping Sumo Kart** en introduisant un système de gestion de carburant. Ce système contraint les joueurs à économiser leur énergie lors du déplacement du drone, avec la possibilité de se ravitailler aux pit stops en récompense de l'exploration du circuit. Lorsque le carburant devient critique, le drone ralentit ; lorsqu'il s'épuise complètement, le drone perd sa capacité à avancer normalement et ne peut se déplacer qu'en zigzag.

Composants implémentés

1.1 Système de carburant

- **Jauge de carburant** : Capacité maximale de 100 unités
- **Consommation** : Le carburant se consomme uniquement lors de l'appui sur le bouton "marche avant" (non continu)
  - Vitesse normale (20) : consommation de 5 unités par appui
  - Vitesse réduite (10) : consommation de 2 unités par appui
- **Seuil critique** : À 10 unités, le drone passe automatiquement en vitesse réduite
  - Indication visuelle : la jauge passe au rouge
- **État sans carburant** : À 0 unités, le drone ne peut plus avancer normalement et ne peut se déplacer qu'en zigzag
- **Feedback visuel** :
  - Vert (100 à 10 unités)
  - Jaune au démarrage (initialisation)
  - Rouge (10 à 0 unités)

1.2 Système de Pit Stop

- **Détection** : Le pit stop est identifié via un **ARTag "B"** (objet banane)
- **Ravitaillement** : Le drone remplit sa jauge de carburant (+25 unités) sous trois conditions :
  1. Le drone détecte le tag ARTag "B" à une distance suffisante
  2. Le drone est immobile au moment de la détection
  3. Un délai minimal `timeSinceLastPit` s'est écoulé (évite les rechargements trop rapides)
- **Entrée en pit stop** : Le drone entre dans le pit stop lorsqu'il détecte le tag ARTag "B" à proximité (distance suffisante)
- **Mode pit stop** : Une fois en pit stop, le drone cesse la consommation de carburant
- **Sortie du pit stop** : Le drone DOIT détecter le tag ARTag "B" à une **distance suffisante** pour quitter le pit stop
  - **Cas critique** : Si le drone effectue un virage sec en quittant le pit stop et perd le tag de vue assez longtemps, il peut rester bloqué en mode pit stop jusqu'à détecter à nouveau

le tag à une distance valide

2. État d'avancement et limitations connues

2.1 Fonctionnalités implémentées

- Système de jauge de carburant avec 3 états visuels (vert/jaune/rouge)
- Consommation de carburant à l'appui du bouton "marche avant"
- Ralentissement automatique au-dessous du seuil critique (10 unités)
- Mode zigzag lorsque le carburant est à zéro
- Détection du pit stop via ARTag "B"
- Ravitaillement en carburant dans le pit stop (si drone immobile)
- Suspension de la consommation en mode pit stop
- Délai `timeSinceLastPit` pour éviter les rechargements trop rapides

2.2 Limitations et comportements spécifiques

- **Consommation non-continue** : Le carburant ne diminue QUE lors d'un appui sur "marche avant", pas continuellement à chaque frame. Cela signifie qu'un drone qui ne se déplace pas ne consomme pas, même s'il est en mouvement passif.
- **Problème de détection à la sortie du pit stop** : Si le drone effectue un virage serré en quittant le pit stop et que le tag "B" n'est plus visible à distance valide, le drone reste bloqué en mode pit stop. Il ne pourra en sortir que lorsqu'il détectera à nouveau le tag à une distance suffisante.
- **Dépendance à la détection ARTag** : Tout le système de pit stop repose sur la détection fiable du tag "B". Les angles de vue, l'éclairage et la distance affectent directement le comportement.

2.3 Recommandations pour les évolutions futures

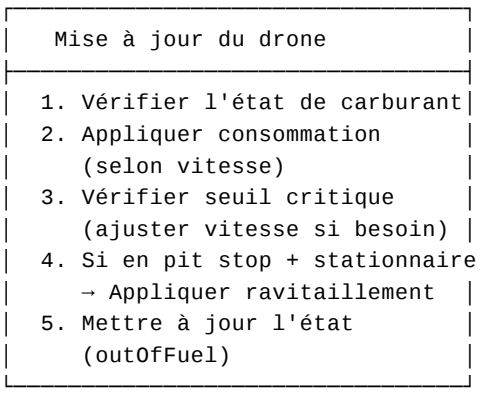
- Ajouter un mécanisme de détection de sortie alternatif (ex: basé sur la distance parcourue ou le temps écoulé)
- Implémenter une zone de "sortie sûre" du pit stop pour éviter le blocage
- Ajouter des logs visuels (HUD) pour afficher clairement l'état du pit stop
- Considérer une consommation continue plutôt qu'à l'appui pour plus de réalisme

3. Logique générale d'implémentation

Architecture

Le système est implémenté dans la classe `Drone` qui représente un drone en tant que joueur dans l'application (et non comme un périphérique).

Flux d'exécution



Méthodes principales

`consumeFuel(byte speed)`

- Consomme du carburant basé sur la vitesse actuelle
- N'agit que si le drone n'est pas en pit stop et possède du carburant
- Sauvegarde les détails dans les logs pour le débogage

`refillFuel()`

- Augmente le carburant de 25 unités par appel
- Garanti par trois conditions : pit stop actif, carburant < max, drone stationnaire
- Évite les incohérences logiques

`setCurrentFuel(int fuel)`

- Définit le carburant avec deux protections :
  - Plafonnement à MAX_FUEL (100)
  - Plancher à 0
- Met à jour automatiquement le flag `outOfFuel`

`canMoveAtFullSpeed()`

- Retourne vrai si carburant > 10
- Utilisée par le système de physique/déplacement pour réduire la vitesse

`isOutOfFuel() et isInPitStop()`

- Fonctions de requête pour l'état actuel
- Utilisées par le système de jeu pour ajuster le comportement du drone

4. Choix techniques et justifications

3.1 Constantes statiques

Choix : Toutes les valeurs numériques sont des constantes statiques (`MAX_FUEL` , `FUEL_CONSUMPTION_NORMAL` , etc.)

Justification :

- **Maintenabilité** : Facilite l'ajustement de la difficulté du jeu
- **Clarté** : Les noms explicites rendent le code auto-documenté
- **Évite la "magie des nombres"** : Facilite la compréhension pour les futurs développeurs

3.2 Seuil critique (10 unités)

Choix : Un seuil spécifique déclenche une réduction de vitesse avant l'épuisement complet

Justification :

- **Gameplay stratégique** : Força les joueurs à anticiper leur ravitaillement
- **Prévention de frustration** : Évite que le drone s'arrête brutalement en milieu de course
- **Feedback progressif** : Le joueur reçoit un avertissement (ralentissement) avant l'arrêt complet

3.3 Ravitaillement (25 unités)

Choix : Taux de 25 unités (1/4 du réservoir) par mise à jour

Justification :

- **Équilibre de temps** : Permet une recharge rapide mais pas instantanée
- **Coût stratégique** : Arrêter pour se ravitailler a un vrai coût temporel
- **Récupération progressive** : Rend les pit stops pertinents sans les rendre obligatoires

3.4 Conditions de ravitaillement

Choix : Trois conditions doivent être satisfaites simultanément :

1. `inPitStop == true`
2. `currentFuel < MAX_FUEL`
3. `!this.isMoving`

Justification :

- **Condition 1** : Sécurité (revérifier que le drone est bien en pit stop)
- **Condition 2** : Logique (pas de gaspillage/débordement de carburant)
- **Condition 3** : Gameplay (impose un coût d'arrêt, empêche le "ravitaillement au ralenti")

3.5 Booleans d'état

Choix : Deux booleans : `inPitStop` (modifiable) et `outOfFuel` (dérivé)

Justification :

- **`inPitStop`** : Contrôlé de l'extérieur par le système de détection de marqueurs
- **`outOfFuel`** : Mis à jour automatiquement via `setCurrentFuel()` pour garantir la cohérence

3.6 Logging

Choix : Logs détaillés dans les méthodes de consommation et ravitaillement

Justification :

- **Débogage facilité** : Permet de tracer les changements d'état du carburant
- **Monitoring** : Utile pour tester les équilibrages de gameplay

5. Intégration avec le système global

Points d'intégration

- **Système de déplacement** : Utilise `canMoveAtFullSpeed()` pour ajuster la vitesse
- **Détection de marqueurs** : Gère `setInPitStop()` selon la détection de pit stop
- **Boucle de mise à jour du jeu** : Appelle `consumeFuel()` et `refillFuel()` à chaque frame
- **Interface utilisateur** : Affiche la jauge via `getCurrentFuel()` ou `getFuelPercentage()`

Sensibilité au contexte

Le système est conscient de l'état du drone :

- Ralentit automatiquement en cas de carburant faible
- Refuse la consommation en pit stop
- Synchronise avec la mobilité du drone pour le ravitaillement

6. Évolutions futures possibles

- **Boost temporaire** : Consommation accrue mais vitesse augmentée
- **Dégradation progressive** : La vitesse diminue progressivement avec le carburant (vs. seuil brusque)
- **Types d'items** : Certains items pourraient affecter la consommation
- **Pit stops multiples** : Différents taux de ravitaillement selon le pit stop
- **Dommages** : Fuites de carburant en cas de collision

7. Conclusion

Le système de carburant et pit stop ajoute une couche stratégique au jeu sans compliquer excessivement l'architecture existante. Les choix techniques permettent une évolution future facile tout en restant simple à comprendre et maintenir.