# Analyse post-mortem PP2I, semestre 6

## Contributions individuelles :

Tarek :

Mécaniques de jeu

- Mise en place de la règle "3 pas en arrière → Game Over"
- Génération procédurale de la carte avec :
    - Possibilité de faire jusqu'à 2 pas en arrière
    - Écart entre les safe-zones grandissant avec le score
    - Ajout d'arbres sur les safe-zones sans obstruer le point de spawn du joueur

Gestion des véhicules

- Ajout de véhicules se déplaçant indépendamment du joueur, chacun avec sa propre vitesse
- Utilisation d'une liste doublement chaînée pour permettre l'ajout dynamique par le haut et la suppression par le bas
- Mise en place de routes avec plusieurs véhicules selon la difficulté, sans superposition :
    - Création d'une structure "EffectQueue" gérant des événements différés
    - Gestion précise de l'intervalle d'apparition des véhicules, prenant en compte leur vitesse et leur taille
    - Prise en charge de la taille des véhicules (lorsqu'un véhicule de taille > 1 quitte l'écran, sa queue disparaît progressivement)
    - Détection complète des collisions, y compris avec l'ensemble des segments du véhicule (tête, corps, queue)

Rivières et rondins

- Ajout de rivières et de rondins
- Le joueur peut monter sur un rondin, qui le transporte automatiquement
- Si le rondin sort de l'écran, le joueur perd
- Rondins avec logique opposée aux voitures : vitesse plus lente, taille qui diminue avec le temps

Logiques internes et robustesse

- Refonte de la logique de mise à jour des effets (correction d'un bug de freeze)
- Séparation claire des logiques fonctionnelles
- Remplacement des deque par des queues simples
- Verrouillage des bords de la carte : impossibilité de sortir de la grille (correction des indices de départ/fin)
- Nettoyage mémoire effectué en fin de jeu

Interface terminale et qualité générale

- Correction du scintillement de l'écran
- Affichage directionnel des véhicules : distinction entre ceux allant vers la droite (">") et vers la gauche ("<")
- Gestion précise du score, avec prise en compte des cas spéciaux (ex. : décès en mouvement)
- Gestion de l'abandon : détection de l'inactivité prolongée du joueur

Rédaction d'un fichier README clair

Nils:

Logique du jeu:

- Remplacement de l'effect queue
- Création des rowManagers
- Amélioration de l'algorithme de génération des lignes (placement des arbres, des véhicules, gestion de la difficulté)
- Création des lignes de train
- Création des lignes de glace

Version CLI :

- Structure de base (game state, boucle de gameplay)

Version Graphique :

- Implémentation de SDL, création de l'interface graphique (affichage des sprites, du score, des textes)
- Ecran titre
- Ecran game over

Théo:

State of the Art:

- Recherche et documentation sur les différentes méthodes et algorithmes d'IA dans les jeux tels que Crossy Road :
    - Recherches sur internet
    - Recherches sur github
    - Lecture de papiers scientifiques
- Rédaction du State of the Art

Algorithme d'IA:

- Comparaison des différents algorithmes et choix de l'algorithme à implémenter
- Implémentation de l'algorithme A* :
    - Ajout de structure de données Nœud, représentant les cellules et leurs caractéristiques pour l'algorithme

    - Ajout de structure PriorityQueue, permettant de retirer simplement le meilleur candidat parmi plusieurs nœuds
        - Fonctions de création, de recherche par index, de recherche par coordonnées, d'actualisation par le haut, d'actualisation par le bas, d'affichage, de tirage du meilleur nœud
    - Ajout de l'algorithme et des fonctions utilitaires simplifiant le code
    - Adaptation de l'algorithme aux mouvements de voitures en prédisant leur position
- Création de l'agent d'IA :
    - Calcul du point d'arrivée
    - Appelle l'algorithme A* pour déterminer le chemin entre sa position et l'arrivée
    - Lien entre le chemin récupéré et l'appel des fonctions de déplacements du joueur

Gestion de projet :

- Rédaction des comptes-rendus de réunion
- Création du ClickUp et initialisation des tâches à effectuer
- Rédaction de la fiche projet et du compte-rendu

## Évaluation des objectifs initiaux

| Objectif | Atteint | Commentaire |
|---|---|---|
| Version terminale fonctionnelle | Oui | Finalisé assez rapidement |
| Interface graphique fluide avec SDL | Oui | Interface propre, fluide et esthétique + sons en bonus |
| IA capable de jouer automatiquement | En grande partie | Intégration dans le jeu fluide et scores assez élevés mais parfois quelques problèmes de prédiction des voitures |
| Gestion de projet | Oui | Bonne coordination et organisation, notamment grâce aux outils de gestion de tâches utilisés |

## Conclusion

Points forts du projet :

Nous sommes satisfaits de la qualité du jeu livré, qui donne vraiment envie de jouer. De plus il y a eu une très bonne cohésion et organisation entre les membres de l'équipe, ce qui a facilité les

intégrations et les ajouts de fonctionnalités. Nous avons beaucoup appris de ce projet, notamment en termes de structures de code, d'anticipation des fonctionnalités à ajouter et en adaptation à de nouvelles technologies (SDL / IA). Cela nous a aussi appris à rendre du code robuste dans un temps défini.

Points faibles du projet :

Quelques difficultés d'intégration sur les dernières fonctionnalités et manque d'anticipation du temps nécessaire à la fusion des différentes branches et à la gestion des conflits. Nous aurions aussi pu prévoir un peu plus de temps pour les tests unitaires et l'optimisation du code.

Leçons retenues pour un futur projet :

Une communication claire est la clé de la réussite d'un projet, surtout lorsqu'on travaille sur des tâches différentes. Prévoir plus de temps pour tester et optimiser le code. Anticiper les éventuels conflits et bugs liés aux différentes versions du projet.