Rapport OUV-INF : Jeu "Guns" en pygame

Pierre BERTHOLOM, Kinan Al MOHAMMAD, Gatien AUFFRET,
JOUAN François, COJEAN Baptiste

Cycle Préparatoire Année 2
Ouverture Info - S3

18 janvier 2024

Sommaire

1 Introduction

Le jeu s'appelle Guns et est un jeu de 1 contre 1 en réseau/online. Le but est de tirer et toucher le joueur adverse. Chaque joueur a 4 coeurs et faire perdre les 4 coeurs du joueurs adverse fait gagner un point/une manche. Le jeu se termine quand un des deux joueurs atteint 5 points.
Le jeu se joue au clavier pour se déplacer et on peut modifier les touches dans le menu settings. On peut se déplacer de droite à gauche et on peut sauter deux fois, un saut simple depuis le sol et un double jump dans les airs. On peut aussi tomber vers le bas plus rapidement en appuyant sur la touche de direction vers le bas. Pour viser et tirer, il faut utiliser la souris, on vise avec le curseur et on tire en faisant un clique gauche. Il n'y a pas de limite de balles mais une durée entre chaque tire.
Durant la partie, des powerups/bonus apparaissent sur la carte, il en existe 3 types :

- Un bonus de vitesse qui double la vitesse du joueur pendant 3 secondes
- Un bonus d'invulnérabilité qui rend invincible soit invulnérable aux tirs pendant 4 secondes
- Un bonus de santé qui restaure un point de vie (un cœur)

Les powerups ayant une durée peuvent s'accumuler, par exemple, prendre 2 bonus de vitesse à la suite ou en même temps, confère un bonus de vitesse pendant 6 secondes.

2 Organisation

Pour la réalisation de notre jeu Pygame, nous avons opté pour une gestion efficace de notre projet en utilisant Trello et GitLab. Trello a été notre outil principal pour la répartition des tâches au sein de l'équipe, tandis que le README.md de GitLab nous a permis de lister les différentes tâches. Grâce à l'interface intuitive de Trello, nous avons pu créer des tableaux collaboratifs, assigner des responsabilités à chaque membre, et suivre en temps réel l'avancement de chacune des étapes du développement. Cela a grandement facilité la coordination de nos efforts. De plus, nous avons utilisé GitLab pour le partage et la gestion des fichiers du projet. En utilisant des dépôts partagés, nous avons pu collaborer de manière transparente en téléchargeant, modifiant et fusionnant les différentes versions du code. Cette combinaison d'outils a grandement contribué à la réussite de notre collaboration et à la réalisation efficace de notre jeu.

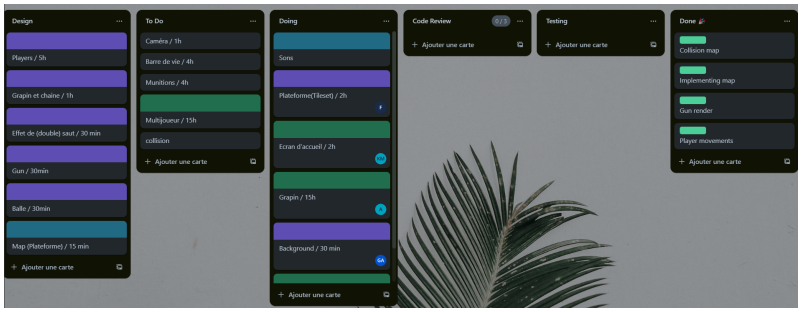

FIGURE 1 – Trello du projet

README.md

Game Project

- [ ] Design :
  - [ ] Players / 5h
  - [ ] Tileset / 2h
  - [ ] Grappling hook and chain / 1h
  - [ ] Arrow / 30min
  - [ ] Bow / 30 min
  - [ ] (Double) jump effect / 30 min
  - [ ] Background / 30 min
- [ ] Code :
  - [ ] Grappling hook / 15h
  - [ ] Home screen / 2h
  - [ ] Multiplayer / 15h
  - [ ] Shoot / 5h
  - [ ] Health bar / 4h
  - [ ] Ammunition / 4h
  - [ ] Camera / 1h
  - [ ] Map (platform) / 30 min
  - [x] Collision map
  - [x] Implementing map
  - [x] Gun render
  - [x] Player movements

FIGURE 2 – README de Gitlab

Dans l'élaboration de notre planning pour le développement, nous avons pris la décision stratégique de commencer par la création de l'environnement basique du jeu avant de nous plonger dans les détails spécifiques.

Cette approche nous a permis de poser des fondations solides de notre projet en concentrant nos efforts initiaux sur la mise en place de l'univers dans lequel notre jeu évoluera. En construisant d'abord les bases, tels que la map, les déplacement du joueur et d'autres éléments essentiels, nous avons pu établir une structure robuste qui nous a permis l'intégration progressive des éléments plus complexes par la suite.

Nous avons ainsi pu diviser notre groupe en deux parties distinctes, une partie chargée de majoritairement coder le jeu, pendant que l'autre partie travailleait sur les graphismes. Par exemple, Baptiste s'est chargé de coder la fonctionnalité de tir pendant que Pierre se chargeait de travailler sur la fonctionnalité de grappin (ultèrieurement abandonnée car la physique de celui-ci ne convenait pas). Kinan a pu se charger de mettre en place un menu, et une page de settings/paramètres. Gatien et François avaient eux, pour mission de faire les graphismes du jeu, que ce soit les menus ou les graphismes des personnages, map etc...

Cette approche séquentielle nous a également offert une vision claire de l'évolution du jeu, permettant une meilleure planification des étapes suivantes en fonction des besoins croissants. En fin de compte, cette stratégie nous a aidés à optimiser notre processus de développement en assurant une progression cohérente et ordonnée dans la création de notre jeu.

3 Présentation

3.0.1 Déroulement d'un partie

Tout d'abord, au lancement du jeu nous arrivons sur un menu dans lequel il nous est possible de changer nos touches et couper la musique en allant dans settings, quitter le jeu avec exit et bien sur jouer avec play. Voici ce menu :

FIGURE 3 – Menu du jeu

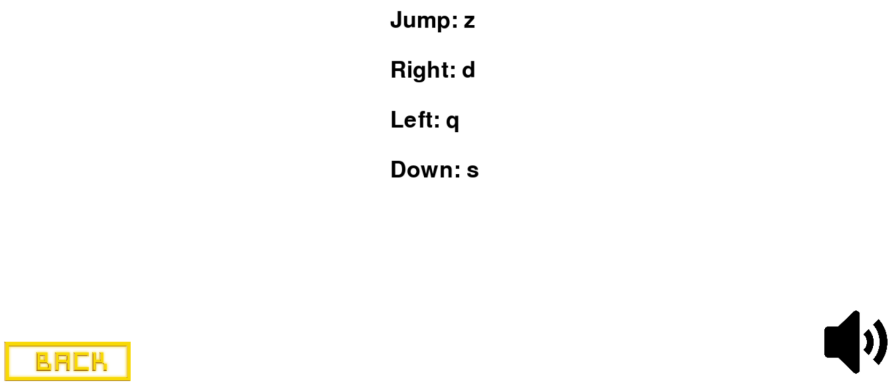

FIGURE 4 – Settings du jeu

Une fois le bouton play cliqué, une image d'attente du deuxième joueur apparaît, sauf si ce dernier est déjà en attente :

FIGURE 5 – Écran d'attente du deuxième joueur

Une fois le deuxième joueur dans la partie, le combat commence. Entre musiques épiques et vitesse de jeu élevée, pas le temps de se reposer, l'objectif : abattre l'ennemi. Pour cela nous devons lui tirer dessus à 4 reprises ou qu'il tombe dans le vide. Une fois l'ennemi abattu, nous gagnons un point. Le premier ayant 5 points remporte la partie. Voici des images du jeu en lui-même :

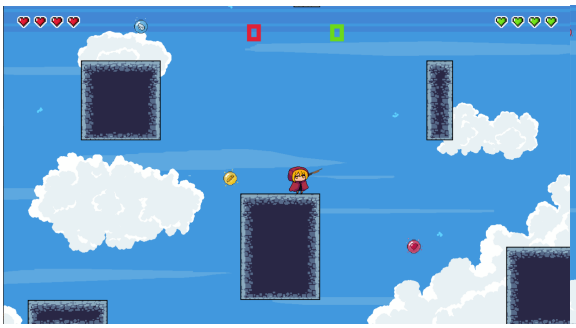

FIGURE 6 – Écran du Joueur 1

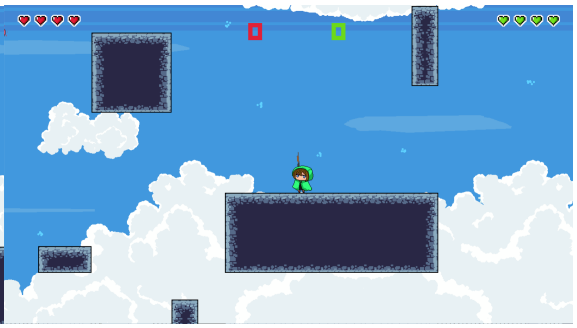

FIGURE 7 – Écran du Joueur 2

Quand un des deux joueurs atteint 4 points, c'est la manche finale, la musique ainsi que le fond du jeu change pour mettre en exergue la proximité de fin de partie. C'est le prélude à la victoire pour l'un et à la défaite pour l'autre. À ce stade de la partie, l'ombre de la défaite se dissipe pour laisser place à l'éclat d'une dernière chance, une ultime opportunité suspendue dans l'éther du jeu :

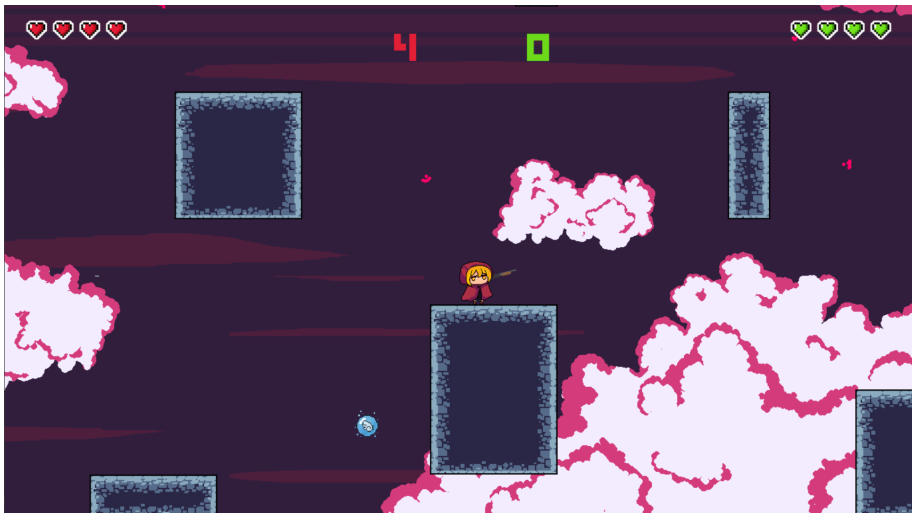

FIGURE 8 – Dernière chance

À la fin de la partie, un écran personnalisé s'affiche, un pour le vainqueur, un pour le vaincu :

FIGURE 9 – Écran du vainqueur

FIGURE 10 – Écran du vaincu

Après quelques secondes, chaque joueur est renvoyé sur le menu.

3.0.2 Explications techniques

Nous allons maintenant aborder plus en détails certaines caractèristiques de notre jeu. Attardons nous tout d'abord sur la création de la map.

La map n'est pas généré aléatoirement mais est défini par un fichier CSV et un fichier image contenant les tiles. Une tile est une case, un bloc de construction de la map, un bloc pour la partie haute d'une plateforme, un bloc pour la partie basse etc..

Chaque tile fait 64x64 pixels et est assigné une valeur, par exemple le deuxième bloc de la partie haute d'une plateforme a la valeur 1. Ainsi, le fichier CSV va déterminer chaque partie de la map comme un bloc ayant une valeur précise, s'il n'y a pas de bloc, la valeur est -1.

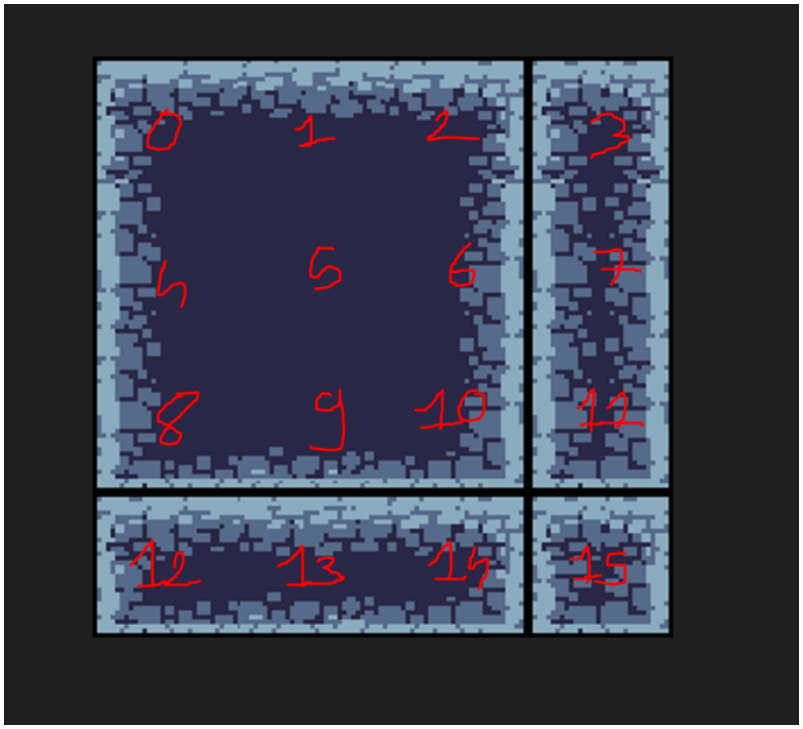

FIGURE 11 – Fichier contenant les tiles avec les IDs des tiles

Il serait fastidieux de faire le fichier CSV à la main, c'est pourquoi, on a utilisé le logiciel Tiled qui permet de graphiquement créer la map en posant chaque bloc à la main puis générer un CSV de notre création.

Dans la partie code désormais, nous avons besoin de découper l'image contenant les tiles en leur attribuant leur valeur utilisé dans le CSV. Puis, lire le CSV et créer un surface avec les tiles. La surface est créée au lancement et est sa position est actualisée en fonction de celle de joueur (de façon à ce que la caméra soit toujours centrée sur le joueur).

Notre personnage possède la capacité de faire un double saut. Il fonctionne de telle sorte que, grâce à un compteur de sauts, si le nombre de sauts est de 1, il peut sauter une deuxième fois et si il saute, on ajoute des particules de poussière. Si le compteur est à deux, le joueur ne peut plus sauter avant de toucher le sol, en touchant le sol, le compteur est réinitialisé.

Pour gérer les collisions avec l'environnement, on utilise les "rect" du joueur et de l'environnement (moins gourmant en performance que les masks et entraîne moins de bugs), si on détecte une collision le rectangle du joueur est modifié de façon à être collé au bloc touché (si le bas du rect du joueur touche le sol, le bas du rect devient le haut du rect du bloc de sol). Les collisions se font uniquement pour le joueur du client, on ne fait que récupérer la position de l'autre joueur et de l'afficher.
Pour les collisions avec les balles, tout se fait encore localement (non idéal mais plus simple à implémenter), le client gère les collisions avec une balle qui le touche et touche l'autre joueur. S'il détecte qu'une balle l'a touché, il perd un coeur. Cette implémentation amène un léger problème étant que si un joueur A détecte une collision de sa balle (A) avec le joueur B et que le joueur B ne détecte pas la collision de la balle (A) avec lui-même alors le joueur B ne perd pas de vie. Une solution serait de gèrer la décision de perte de vie par le serveur, si les deux joueurs détectent qu'une balle a bien touché un joueur, le joueur s'étant fait touché perd une vie.

Cette implémentation n'a pas été mise en place pour garder la partie réseau du jeu la plus simple possible. Le serveur ne fait office que de passerelle pour échanger les données entre les deux joueurs. L'intégralité de la boucle de jeu est gérée en local.

Notre jeu comporte également des powerups tels qu'un gain de vie supplémentaire, un sort d'invulnérbilité et un boost de vitesse. Le powerup de vitesse permet au joueur de se déplacer deux fois plus vite pendant 3 secondes et peut s'additionner permettant, si bonne gestion du personnage, de conserver cette vitesse très longtemps. Le powerup d'invincibilité permet de résister aux tirs adverses pendant 4 secondes. Les powerups sont générés en utilisant np.random.rand() qui génére un float entre 0 et 1, si le nombre généré est plus petit que 0.003, ont génère un powerup, soit 1 chance sur 333 environ de générer un powerup par appel de la fonction, en prenant en compte que le jeu tourne à 60fps, et que la fonction est appelée à chaque frame, il y a en moyenne 1 powerup toutes les 5 secondes environ. Les types de powerup sont déterminer aléatoirement selon les taux suivant :

FIGURE 12 – Différents powerups avec leur taux

4 Limites du jeu

Bien que notre jeu présente des caractéristiques prometteuses, il comporte certaines limites que nous avons identifiées tout au long du processus de développement. Premièrement, la fréquence d'images est limitée, même avec un ordinateur performant, ($\approx$ 54 fps), ce qui affecte l'expérience et la fluidité du jeu et contraint les joueurs ayant un ordinateur peu puissant. De plus, la vérification des collisions ne s'effectue pas côté serveur, ce qui peut entraîner des confusions ou des désynchronisations dans le déroulement du jeu. En outre, le jeu est actuellement conçu pour une seule carte, limitant la variété des environnements disponibles. L'absence d'une option pour jouer contre une intelligence artificielle constitue également une restriction, limitant l'expérience solo des joueurs. De plus, une fois la partie lancée, il n'est pas possible de revenir au menu, ce qui peut être perçu comme une contrainte d'ergonomie. Par ailleurs, la fenêtre de jeu n'ai pas redimensionnable et ne prend pas en charge le mode plein écran, car cela entraîne une perte de performance (dans le cas de l'augmentation de la taille de la fenêtre). Enfin, il est important de noter que les joueurs doivent être sur le même réseau pour participer, ce qui peut limiter l'accessibilité du jeu à des contextes spécifiques.

5 Conclusion

En conclusion, dans le cadre de la découverte du module informatique de l'ESIR, notre projet de développement d'un jeu vidéo en ligne type 1 contre 1 a été une expérience enrichissante, mais elle a également été ponctuée de défis significatifs.

Globalement, nous sommes satisfaits du résultat final, mais il est important de souligner les aspects positifs et les domaines où des améliorations pourraient être apportées.

Du côté des satisfactions, le fonctionnement correct du mode multijoueur constitue un point fort majeur. Cela témoigne de la réussite de notre organisation et de notre capacité à mettre en place les connaissances réseaux obtenus l'an dernier. Cependant, nous avons été confrontés à des difficultés, notamment lors de la mise en place du grappin, qui s'est avéré être une tâche plus complexe que prévue et nous avons finalement décidé d'abandonner cette fonctionnalité dont la physique était trop complexe pour obtenir un résultat satisfaisant qui ne cassait pas la dynamique du jeu. De plus, l'ambition élevée dans le développement de la map et du design a généré des challenges supplémentaires. Malgré cela, l'aspect principal du jeu a été bien respecté, en cohérence avec nos attentes, ce qui témoigne de notre engagement envers la vision initiale du projet. En ce qui concerne l'organisation, nous reconnaissons que celle-ci a été globalement bien effectuée. Cependant, des travaux parfois trop complexes ont entraîné des pertes de temps inutiles. De même, l'ordre de programmation, avec le codage du multijoueur à la fin, a conduit à des ajustements nécessaires, ce qui aurait pu être évité avec une approche plus stratégique et plus d'anticipation.

En rétrospective, nous aurions fait différemment en modifiant l'organisation pour la programmation en pensant le code pour l'implémentation du multijoueur. Cette modification aurait permis d'optimiser notre temps et d'éviter des ajustements ultérieurs. De plus, nous aurions adopté une approche moins ambitieuse dans le développement de la map et du design, afin de garantir la réalisation réussie de l'ensemble des fonctionnalités prévues.

Finalement,les enseignements tirés de cette expérience nous serviront de base solide pour des projets futurs, en nous incitant à prioriser la planification stratégique et à ajuster nos ambitions en fonction des contraintes de développement.