Rot it or Lose it

Rapport Final de Projet

SAE J3D

Membres du groupe :

Oscar Pinto Fernandes
Lucas Koch
Flavien Teboul
Amadou Diaw

EPITA
2025

Table des matières

1 Introduction

1.1 Présentation du projet

Dans le cadre de notre projet de développement de jeu vidéo, nous avons réalisé un jeu en 2D nommé *Rot it or Lose it*. Le projet repose sur l'univers des casinos et des jeux d'argent, avec une ambiance rétro inspirée des jeux arcade et pixel art.

Le joueur peut se déplacer librement dans un casino et accéder à plusieurs mini jeux différents comme la roulette, le blackjack, le Buckshot Roulette ou encore des jeux d'arcade. Le projet mélange donc exploration, gestion de l'argent virtuel et gameplay basé sur le hasard et la prise de décision.

L'univers du jeu est volontairement exagéré et caricatural. Nous avons choisi de reprendre les codes classiques des casinos et des jeux d'argent en les poussant à l'extrême afin de créer une ambiance décalée et facilement reconnaissable. Le projet ne cherche pas à représenter un casino réaliste, mais plutôt une vision stylisée et volontairement excessive de cet univers.

L'objectif principal du projet était de créer un jeu cohérent, jouable et agréable à parcourir, tout en développant une architecture capable de gérer plusieurs systèmes différents dans un même environnement.

Ce projet nous a également permis de travailler sur de nombreux aspects du développement, comme la gestion des interfaces, les collisions, les sauvegardes, les maps, le réseau local, mais aussi l'organisation du travail en équipe.

1.2 Concept du jeu

Rot it or Lose it se déroule dans une ville fictive appelée *Rotted City*. Dans cet univers, le casino représente le principal lieu de vie et d'activité. Le joueur peut s'y déplacer librement et participer à plusieurs activités autour des jeux d'argent.

Le fonctionnement du jeu est basé sur une monnaie virtuelle appelée *Rot Coins*. Le joueur utilise cet argent pour accéder aux différents mini jeux du casino. Selon les résultats obtenus, il peut gagner ou perdre de l'argent afin de continuer sa progression.

Chaque mini jeu possède son propre fonctionnement et son propre gameplay. La roulette repose principalement sur le hasard, tandis que le blackjack demande davantage de réflexion et de prise de décision. Le Buckshot Roulette, quant à lui, met davantage l'accent sur la tension et le risque.

Nous avons voulu créer un projet qui ne soit pas uniquement une succession de menus ou de mini jeux séparés. Le joueur évolue dans un véritable environnement avec des déplacements, des transitions entre zones, des interactions et une identité visuelle cohérente.

1.3 Objectifs du projet

Le premier objectif du projet était de réaliser un jeu fonctionnel capable de regrouper plusieurs systèmes dans une même architecture. Nous voulions créer un projet stable, structuré et suffisamment flexible pour intégrer plusieurs mini jeux différents.

Nous avions également pour objectif de proposer une ambiance reconnaissable, avec une direction artistique rétro et chaleureuse inspirée des anciens jeux d'arcade et des casinos.

Sur le plan technique, le projet devait intégrer plusieurs éléments importants : la gestion des maps, les collisions, les interfaces utilisateur, les sauvegardes, la gestion des données joueur ainsi qu'un mode multijoueur en réseau local.

Ce projet avait aussi pour but de nous faire progresser sur le travail en groupe. Nous avons dû apprendre à organiser un projet assez important, répartir les tâches, corriger les conflits de code et travailler à plusieurs sur une même base de projet.

Enfin, nous voulions obtenir un résultat suffisamment complet pour proposer une vraie démonstration jouable lors de la soutenance finale.

1.4 Contexte de développement

Le projet a été réalisé dans le cadre de notre année, au sein d'un groupe de plusieurs étudiants. Le développement s'est étalé sur plusieurs mois, avec différentes phases de conception, de développement, de tests et d'intégration.

Nous avons principalement utilisé Python avec la bibliothèque Pygame pour le développement du jeu. Les maps ont été réalisées avec le logiciel Tiled, ce qui nous a permis de créer plus facilement les différentes zones du casino.

Le projet a évolué progressivement entre les différentes soutenances. Au départ, seules les bases du moteur et certaines fonctionnalités principales étaient présentes. Au fil du temps, plusieurs mini jeux ont été ajoutés, les maps ont été finalisées, les interfaces ont été améliorées et un système réseau a commencé à être intégré.

Le développement n'a pas toujours été simple. Nous avons rencontré plusieurs problèmes techniques liés à l'intégration des systèmes, à l'organisation du code et à la gestion du temps. Certaines fonctionnalités prévues au départ ont dû être simplifiées ou abandonnées afin de pouvoir concentrer le travail sur les éléments les plus importants du projet.

Malgré ces difficultés, le projet a beaucoup évolué entre le début et la soutenance finale, aussi bien sur le plan technique que visuel.

2 Cahier des charges et objectifs

2.1 Objectifs de départ

Au début du projet, notre objectif principal était de créer un jeu vidéo original mélangeant exploration, mini jeux et ambiance rétro autour de l'univers des casinos.

Nous voulions proposer un projet qui ne se limite pas à une simple suite de mini jeux indépendants. L'idée était de créer un véritable environnement dans lequel le joueur pourrait se déplacer librement et accéder à différentes activités dans un même univers.

Le projet devait également proposer une identité visuelle forte, inspirée des jeux rétro en pixel art. Nous avons rapidement choisi une direction artistique volontairement caricaturale et exagérée afin de renforcer l'ambiance du casino et du jeu d'argent.

Nous voulions aussi créer un projet techniquement ambitieux pour notre niveau, avec plusieurs systèmes reliés entre eux comme les maps, les collisions, les sauvegardes, les interfaces et le multijoueur en réseau local.

2.2 Fonctionnalités prévues

Lors de la rédaction du cahier des charges, plusieurs fonctionnalités avaient été prévues pour le projet.

Le joueur devait pouvoir se déplacer librement dans le casino et interagir avec les différents éléments présents dans les maps. Plusieurs mini jeux étaient également prévus, notamment la roulette, le blackjack, le Buckshot Roulette ainsi qu'un mode arcade.

Un système de sauvegarde devait permettre de conserver la progression du joueur ainsi que son argent virtuel. Nous avions également prévu d'ajouter un mode multijoueur en réseau local afin de permettre à deux joueurs de jouer ensemble.

Certaines fonctionnalités secondaires étaient aussi envisagées, comme des personnages non joueurs, des dialogues ou encore un jeu de cartes personnalisé. Certaines de ces idées ont ensuite été simplifiées ou abandonnées au cours du développement afin de concentrer le travail sur les systèmes principaux du projet.

2.3 Contraintes techniques

Le projet devait respecter plusieurs contraintes imposées dans le cadre de la SAE.

Le jeu devait être développé en Python avec la bibliothèque Pygame et fonctionner sur Windows. Le multijoueur réseau était également une obligation importante du sujet.

Nous avons choisi d'utiliser Tiled pour la création des maps ainsi que plusieurs bibliothèques complémentaires comme PyTMX et PyScroll pour faciliter la gestion des décors et des collisions.

Le projet devait également rester jouable et fluide malgré la présence de plusieurs systèmes différents exécutés en même temps.

Une autre difficulté importante concernait l'organisation du projet. Le développement s'est déroulé sur plusieurs mois avec plusieurs personnes travaillant simultanément sur les mêmes fichiers et les mêmes systèmes.

2.4 Évolution des objectifs pendant le projet

Au fil du développement, plusieurs objectifs ont évolué par rapport au cahier des charges initial.

Certaines fonctionnalités prévues au départ se sont révélées plus complexes que prévu, notamment le réseau local et certains systèmes liés aux mini jeux. Nous avons donc dû adapter plusieurs parties du projet afin de garder un ensemble cohérent et présentable pour la soutenance finale.

Le jeu de cartes personnalisé a finalement été abandonné afin de concentrer davantage le travail sur les éléments déjà avancés comme le Buckshot Roulette, les maps et les interfaces.

À l'inverse, certaines parties du projet ont été beaucoup plus développées que prévu au départ. C'est notamment le cas de l'ambiance visuelle, des maps, de l'intégration des mini jeux dans le casino ou encore du travail réalisé sur les interfaces utilisateur.

Même si tous les objectifs initiaux n'ont pas été atteints entièrement, le projet final reste beaucoup plus complet et abouti que les premières versions réalisées au début de l'année.

3 Présentation générale du projet

3.1 Univers et direction artistique

L'univers de *Rot it or Lose it* se déroule dans une ville fictive appelée *Rotted City*. Cette ville représente une version volontairement exagérée et caricaturale d'un environnement urbain centré autour du jeu, de l'argent et des casinos. L'objectif n'était pas de créer une représentation réaliste, mais plutôt une ambiance reconnaissable et marquante.

Le casino constitue le lieu du jeu. C'est dans cet environnement que le joueur évolue, découvre les différents mini jeux et interagit avec les éléments du décor. Nous avons choisi de donner au casino une ambiance à la fois rétro, chaleureuse et légèrement sombre afin de renforcer l'identité du projet.

Dès le début du développement, nous avons voulu nous orienter vers un style graphique en pixel art. Ce choix avait plusieurs avantages. D'abord, ce style correspondait bien à l'ambiance arcade et rétro que nous voulions créer. Ensuite, il permettait de produire plus facilement des éléments graphiques compatibles avec notre niveau et notre temps de développement.

La majorité des éléments graphiques du projet ont été réalisés ou modifiés par le groupe afin de garder une cohérence visuelle entre les différents mini jeux, les maps et les interfaces. Toutefois, certains éléments graphiques ont été générés par intelligence artificielle. Nous avons notamment travaillé sur les tilesets, certaines interfaces, les tables de jeu, les écrans de menu et plusieurs éléments décoratifs présents dans le casino.

Le style visuel du projet repose principalement sur des couleurs chaudes, des lumières tamisées et des éléments rappelant les anciens jeux d'arcade ou les casinos classiques. Les différentes salles possèdent chacune une ambiance légèrement différente afin d'éviter que le joueur ait l'impression d'être toujours dans le même environnement.

Le Buckshot Roulette possède par exemple une atmosphère beaucoup plus sombre et tendue que la salle principale du casino. À l'inverse, les zones d'arcade utilisent des couleurs plus vives et une ambiance plus dynamique.

Nous avons également essayé de rendre les interfaces cohérentes avec le reste du projet. Les boutons, les menus et les écrans de jeu reprennent le style pixel art général afin de garder une identité graphique homogène.

3.2 Gameplay global

Le gameplay principal de *Rot it or Lose it* repose sur l'exploration du casino ainsi que l'accès aux différents mini jeux disponibles.

Le joueur contrôle un personnage qui peut se déplacer librement dans les différentes zones du casino. Il peut explorer les salles, interagir avec les tables de jeu, accéder aux bornes d'arcade ou encore changer de zone grâce aux transitions présentes dans les maps.

Le fonctionnement général du jeu repose sur une monnaie virtuelle appelée les *Rot Coins*. Cette monnaie permet au joueur de participer aux différents mini jeux du casino. Selon les résultats obtenus pendant les parties, le joueur peut gagner ou perdre de l'argent.

Le gameplay alterne donc entre plusieurs phases :

— exploration du casino

— interaction avec les éléments du décor

— participation aux mini jeux

— gestion de l'argent virtuel

Le système de déplacement a été conçu de manière simple afin de permettre une prise en main rapide du jeu. Le joueur peut se déplacer dans toutes les zones accessibles du casino tout en évitant les collisions avec les éléments du décor.

Plusieurs systèmes viennent compléter le gameplay principal :

— transitions auditives entre les maps

— menus interactifs

— sauvegarde des données joueur

— interfaces de mise

— interactions avec certains personnages

Le projet mélange volontairement plusieurs styles de gameplay différents. La roulette repose principalement sur le hasard, le blackjack demande davantage de réflexion, tandis que le Buckshot Roulette met davantage l'accent sur la tension et la prise de risque.

Cette variété permet de rendre l'expérience moins répétitive et donne au joueur plusieurs façons de jouer dans le même environnement.

Nous avons également essayé de garder un gameplay accessible. Même si certaines mécaniques sont inspirées de vrais jeux de casino, le but principal restait de proposer un jeu amusant et facilement compréhensible.

3.3 Fonctionnement du casino

Le casino représente le centre principal du projet. Toutes les activités du jeu sont organisées autour de cet environnement.

Le joueur apparaît dans une salle principale depuis laquelle il peut accéder aux différents mini jeux disponibles. Chaque salle possède sa propre fonction et sa propre ambiance.

La salle principale permet principalement de circuler entre les différentes zones du casino. C'est également dans cette zone que plusieurs éléments d'interface et de navigation sont accessibles.

Certaines salles sont dédiées à des jeux précis. Le Buckshot Roulette, par exemple, possède sa propre salle afin de renforcer l'ambiance particulière de ce mode de jeu.

Les déplacements dans le casino se font grâce à un système de maps et de transitions. Lorsqu'un joueur atteint certaines zones, une transition vers une autre salle peut être déclenchée.

Le fonctionnement du casino repose également sur un système d'interaction. Lorsqu'un joueur s'approche d'un élément interactif, comme une table de jeu ou une borne d'arcade, une fenêtre peut apparaître afin de proposer différentes actions.

Les mini jeux utilisent tous le même principe général :

— le joueur interagit avec une table

— il choisit éventuellement une mise

— la partie démarre

— le résultat modifie l'argent du joueur

Cette organisation permet de relier tous les mini jeux dans un même univers plutôt que de proposer uniquement une succession de menus séparés.

Le casino possède également plusieurs éléments visuels destinés à renforcer l'immersion :

— décorations
— lumières
— tables de jeux
— bornes d'arcade
— comptoirs
— éléments animés

L'objectif était de donner l'impression d'un véritable lieu vivant malgré les limites techniques du projet.

3.4 Présentation des mini-jeux

Le projet contient plusieurs mini jeux accessibles directement depuis le casino.

Le premier mini jeu développé est la roulette. Ce mode permet au joueur de miser une certaine somme avant de lancer une roue générant un résultat aléatoire. Le système reste volontairement simple mais permet de recréer le fonctionnement principal d'un jeu de roulette classique.

Le blackjack constitue un autre mini jeu important du projet. Le joueur affronte un croupier contrôlé automatiquement par le jeu. Le but est d'obtenir une valeur de cartes supérieure à celle du croupier sans dépasser 21 points.

Le Buckshot Roulette représente probablement le mini jeu le plus important du projet. Inspiré du jeu du même nom, il repose sur un système de prise de risque où le joueur et son adversaire utilisent un fusil contenant des balles réelles et des balles à blanc.

Ce mini jeu possède plusieurs systèmes supplémentaires :

— gestion des vies
— animations
— états de jeu
— système de mise
— logique de décision de l'adversaire

Le projet contient également un mode arcade permettant de lancer des mini jeux supplémentaires depuis des bornes présentes dans le casino.

Tous les mini jeux possèdent un gameplay fonctionnel et intégré dans l'environnement principal du jeu.

L'objectif était surtout de proposer plusieurs expériences différentes dans le même projet afin de rendre le casino plus vivant et varié.

3.5 Ambiance sonore et immersion

L'ambiance sonore joue un rôle important dans l'immersion du joueur dans le casino.

Nous avons intégré plusieurs musiques et effets sonores afin d'éviter que le jeu paraisse vide ou silencieux pendant les déplacements et les parties.

La musique principale du casino cherche à créer une ambiance rétro et légèrement détendue rappelant certains anciens jeux d'arcade.

Le Buckshot Roulette possède une ambiance sonore plus lourde et plus tendue afin de renforcer la pression pendant les parties.

Même si la partie sonore du projet n'est pas totalement finalisée, elle contribue fortement à renforcer l'identité générale du jeu et à rendre l'expérience plus immersive.

4 Architecture générale du jeu

4.1 Organisation du projet

Le projet a été organisé de manière à séparer les différentes parties du jeu afin de rendre le développement plus clair et plus simple à maintenir.

Nous avons choisi de structurer le projet autour de plusieurs grands blocs :

— le moteur principal du jeu

— les mini jeux

— les assets graphiques et sonores

— les sauvegardes

— les systèmes réseau

Cette séparation permettait à plusieurs membres du groupe de travailler en parallèle sans modifier constamment les mêmes fichiers.

Le noyau principal du projet contient les éléments communs à tout le jeu :

— la boucle principale

— la gestion de la fenêtre

— les déplacements

— les collisions

— les transitions de maps

— les interactions

Les mini jeux possèdent ensuite leurs propres fichiers afin de garder une architecture plus lisible. Chaque mini jeu possède son fonctionnement, ses états et ses interfaces tout en restant compatible avec le moteur principal.

Les assets sont également séparés par catégories :

— sprites

— musiques

— interfaces

— maps

Cette organisation nous a permis de garder un projet relativement propre malgré la taille grandissante du jeu au fil des soutenances.

Le projet a également été hébergé sur Git afin de permettre un travail en groupe plus efficace et de conserver différentes versions du projet pendant le développement.

4.2 Technologies utilisées

Le projet a principalement été développé en Python avec la bibliothèque Pygame.

Nous avons choisi cette technologie car elle permettait de développer rapidement un jeu en 2D tout en restant adaptée à notre niveau.

Pygame a été utilisé pour :

— l'affichage graphique

— la gestion des entrées clavier

— les collisions

— les animations

— les sons

— les événements

Pour les maps, nous avons utilisé le logiciel Tiled. Cet outil permet de créer des environnements en 2D à partir de tilesets.

Les fichiers TMX générés par Tiled sont ensuite chargés directement dans le jeu grâce à la bibliothèque PyTMX.

Nous avons également utilisé PyScroll afin de faciliter la gestion du rendu des maps et du déplacement de la caméra.

Le site web a été réalisé avec :

— HTML

— CSS

— JavaScript

L'hébergement du site a été effectué avec Hostinger.

Pour la création de l'exécutable Windows, nous avons utilisé PyInstaller, tandis que l'installation et la désinstallation du jeu ont été gérées grâce à Inno Setup afin de permettre une installation simple du projet sans devoir installer Python.

Plusieurs outils annexes ont également été utilisés pendant le développement :

— Git

— GitHub

— Trello

— Discord

— Google Sheets

Ces outils ont principalement servi à l'organisation du groupe et au suivi de l'avancement du projet.

4.3 Architecture du moteur

Le moteur du jeu repose sur une architecture relativement simple mais suffisamment flexible pour permettre l'intégration de plusieurs mini jeux différents dans le même environnement.

La classe principale du projet est la classe `Game`. Elle agit comme le centre du jeu et contrôle :

— la boucle principale

— les mises à jour

— le rendu

— les événements

— le lancement des mini jeux

Le fonctionnement général du moteur suit un cycle classique :

1. récupération des événements clavier et souris
2. mise à jour des objets du jeu
3. affichage des éléments graphiques

4. rafraîchissement de l'écran

La classe `Player` gère le joueur et ses déplacements dans les maps. Elle s'occupe notamment :

— des déplacements

— des collisions

— des interactions

— des animations

Les maps sont gérées séparément grâce à une classe dédiée. Cette architecture permet de changer facilement de salle sans relancer complètement le jeu.

Chaque mini jeu possède ensuite son propre système interne, mais tous utilisent le même principe général :

— lancement depuis le casino

— exécution du mini jeu

— retour du résultat au jeu principal

Cette architecture permet de garder une cohérence globale entre les différentes parties du projet.

Même si le moteur reste relativement simple comparé à des moteurs professionnels, il est suffisamment modulaire pour supporter les différentes fonctionnalités du jeu.

4.4 Gestion des états du jeu

Le projet repose sur plusieurs états de jeu différents permettant de gérer correctement les transitions entre les menus, les maps et les mini jeux.

Le système fonctionne principalement grâce à plusieurs variables et conditions permettant de savoir dans quel état se trouve le joueur à un instant précis.

Le jeu alterne notamment entre :

— le menu principal

— l'exploration du casino

— les mini jeux

— les menus pause

— les écrans de fin

Cette organisation permet de séparer plus facilement les différentes phases du gameplay et d'éviter que plusieurs systèmes soient actifs en même temps.

Lorsqu'un mini jeu est lancé, le moteur principal suspend temporairement certaines parties du casino afin de laisser uniquement le mini jeu actif.

Une fois la partie terminée, le joueur revient automatiquement dans la map principale avec ses données mises à jour.

Cette logique nous a permis de garder un fonctionnement relativement simple malgré la présence de plusieurs systèmes différents dans le projet.

4.5 Gestion des ressources

Le projet contient une grande quantité de ressources graphiques et sonores utilisées dans les différentes parties du jeu.

Afin de garder une organisation claire, les assets ont été séparés dans plusieurs dossiers spécifiques :

— sprites

— maps

— interfaces

— sons

— musiques

— animations

Les images sont principalement chargées grâce aux fonctions de Pygame comme `pygame.image.load(`

Certaines ressources utilisent également `convert_alpha()` afin de gérer correctement la transparence des sprites.

Nous avons aussi dû redimensionner plusieurs images afin de conserver une cohérence visuelle entre les différents éléments du projet.

Le style pixel art du jeu demandait également de faire attention aux proportions et au rendu final des sprites afin d'éviter des images floues ou déformées.

Les sons et les musiques sont chargés séparément grâce au module audio de Pygame.

Nous avons essayé de limiter les chargements inutiles pendant les parties afin d'éviter certaines pertes de performance.

Même si le projet reste relativement léger, cette organisation nous a permis de garder un chargement plus stable et plus simple à maintenir pendant le développement.

4.6 Organisation des classes

Le projet repose sur plusieurs classes permettant de séparer les différentes responsabilités du jeu.

La classe principale `Game` agit comme le centre du projet. Elle gère notamment :

— la boucle principale

— les événements

— le rendu

— les changements de map

— le lancement des mini jeux

La classe `Player` gère principalement :

— les déplacements

— les animations

— les collisions

— les interactions

Les mini jeux possèdent également leurs propres classes afin de garder un code plus lisible et plus indépendant du moteur principal.

Cette séparation permettait de modifier plus facilement certaines parties du projet sans casser l'ensemble du jeu.

Le système de sauvegarde possède également sa propre classe afin de centraliser les opérations liées aux fichiers JSON.

Même si l'architecture du projet reste relativement simple, cette organisation nous a permis de mieux répartir le travail entre les membres du groupe pendant le développement.

4.7 Gestion des maps

Les maps du projet ont été réalisées avec le logiciel Tiled.

Chaque zone importante du jeu possède sa propre map :

— salle principale

— salle de Buckshot

— bar

— arcade

Les maps sont enregistrées au format TMX puis chargées directement dans le jeu grâce à PyTMX.

Nous avons organisé les maps avec plusieurs couches :

— sol

— décorations

— collisions

— interactions

Cette séparation permet de modifier certaines parties de la map sans devoir reconstruire complètement l'environnement.

Les collisions sont définies directement dans Tiled grâce à des objets placés manuellement.

Les zones interactives utilisent également des objets spécifiques. Lorsqu'un joueur entre dans ces zones, certaines actions peuvent être déclenchées :

— ouverture d'un mini jeu

— changement de salle

— affichage d'une interface

Le système de transitions entre maps permet au joueur de circuler librement dans le casino sans écran de chargement visible.

Le rendu des maps est géré avec PyScroll afin de permettre un déplacement fluide de la caméra autour du joueur.

Nous avons également ajouté un mode debug permettant d'afficher les collisions et les zones interactives afin de faciliter les tests.

4.8 Construction des environnements

La création des différentes maps du projet a demandé un travail important de réflexion sur l'organisation des espaces.

Le casino devait rester suffisamment grand pour donner une impression d'exploration, tout en restant lisible et simple à parcourir.

Chaque salle a donc été pensée avec plusieurs objectifs :

— guider naturellement le joueur

— éviter les zones trop vides

— mettre en valeur les mini jeux

— conserver une cohérence visuelle

Les différentes zones du casino ont été construites progressivement à partir de plusieurs couches dans Tiled.

Nous avons séparé :

— le sol

— les décorations

— les collisions

— les interactions

— certains effets visuels

Cette organisation permettait de modifier facilement certaines parties de la map sans devoir reconstruire entièrement l'environnement.

Les zones de passage ont donc été intégrées directement dans les maps afin de rendre les déplacements plus naturels.

Le travail sur les maps a également demandé plusieurs phases de tests, notamment pour :

— vérifier les collisions

— corriger les blocages

— ajuster les tailles des salles

— améliorer la circulation du joueur

Certaines zones ont été entièrement reconstruites plusieurs fois au cours du développement afin d'obtenir un résultat plus cohérent.

Même si le casino reste relativement compact, l'objectif principal était surtout de créer un environnement vivant et crédible pour accueillir les différents mini jeux.

4.9 Gestion des collisions

Le système de collisions est essentiel dans le projet car il empêche le joueur de traverser les murs ou certains objets du décor.

Les collisions sont principalement basées sur des rectangles de type `pygame.Rect`.

Chaque obstacle important possède une hitbox définie directement dans Tiled.

Lorsqu'un déplacement est effectué, le jeu vérifie si la nouvelle position du joueur entre en collision avec un obstacle.

Si une collision est détectée, le déplacement est annulé.

Cette logique relativement simple permet d'obtenir un système stable et suffisamment précis pour les besoins du projet.

Certaines difficultés sont apparues pendant le développement, notamment avec les formes complexes dans Tiled.

Les collisions triangulaires ou inclinées étaient parfois mal interprétées. Nous avons donc dû découper certaines zones en plusieurs rectangles afin d'obtenir un résultat plus fiable.

Un système de debug a également été ajouté afin d'afficher visuellement les hitboxes pendant les phases de test.

Cela nous a permis de corriger plus facilement les problèmes de placement et d'ajuster les collisions de certaines maps.

4.10 Gestion des sauvegardes

Le projet possède un système de sauvegarde permettant de conserver certaines données du joueur entre les sessions.

Les sauvegardes sont stockées dans des fichiers JSON.

Ce choix permet d'obtenir :

— des fichiers lisibles

— une structure simple

— une sauvegarde rapide

Le système de sauvegarde conserve principalement :

— l'argent du joueur

— sa position

Une classe `SaveManager` a été créée afin de centraliser toutes les opérations liées aux sauvegardes.

Cette classe permet notamment :

— de créer une sauvegarde

— de charger une sauvegarde

— de supprimer un slot

Le jeu effectue également des sauvegardes avant certains mini jeux afin de limiter les pertes de progression en cas de problème.

Même si ce système reste relativement simple, il fonctionne correctement et répond aux besoins du projet.

4.11 Interface utilisateur

L'interface utilisateur joue un rôle important dans le projet car elle permet au joueur de comprendre rapidement le fonctionnement du jeu et d'interagir facilement avec les différents mini jeux du casino.

Nous avons essayé de garder une interface simple et lisible afin de ne pas surcharger l'écran avec trop d'informations.

Le HUD principal affiche principalement :

— l'argent du joueur,

Le menu principal permet :

— de lancer une partie,

— de désactiver la musique,

— de quitter le jeu.

Nous avons également ajouté un menu pause accessible avec la touche Échap afin de permettre au joueur de quitter rapidement le jeu ou revenir au menu principal.

Chaque mini jeu possède ensuite sa propre interface adaptée à son gameplay.

La roulette utilise par exemple un tapis interactif sur lequel le joueur peut placer ses mises directement avec la souris. Plusieurs jetons sont disponibles avec différentes valeurs, et l'interface affiche également l'argent actuel du joueur ainsi que certaines instructions liées au fonctionnement du mini jeu.

Le blackjack possède lui aussi une interface spécifique inspirée des tables de casino classiques. Les différentes zones permettant de poser les cartes des joueurs sont directement

affichées sur la table, avec une organisation pensée pour rester lisible même à plusieurs joueurs.

Le croupier est affiché au centre de l'écran afin de renforcer l'ambiance du casino et rendre le mini jeu plus vivant visuellement.

Le Buckshot Roulette possède également plusieurs écrans spécifiques :

— phase de mise,

— tour du joueur,

— tour de l'ennemi,

— écran de fin.

Ce mini jeu affiche différentes informations importantes :

— les vies restantes,

— la mise actuelle,

— les messages d'action,

— certains sprites et animations.

Nous avons essayé de conserver une cohérence graphique globale entre toutes les interfaces du projet grâce à l'utilisation :

— des mêmes couleurs,

— des mêmes polices,

— du même style pixel art,

— d'une ambiance casino rétro commune.

L'objectif principal était de garder une identité visuelle homogène dans l'ensemble du projet malgré les différences de gameplay entre les mini jeux.

5 Développement des mini-jeux

5.1 Implémentation de la roulette

5.1.1 Concept

La roulette est l'un des premiers mini-jeux intégrés dans le projet. Elle repose sur un système de hasard inspiré des roulettes de casino classiques.

Le joueur peut placer différents types de mises sur un tapis interactif, puis lancer un tirage aléatoire afin de connaître le résultat.

L'objectif principal de ce mini-jeu était de proposer une expérience simple à comprendre mais suffisamment dynamique pour s'intégrer naturellement dans l'univers du casino.

La roulette représente également un bon exemple de gameplay centré sur le risque et la récompense, deux notions importantes dans l'identité du projet.

5.1.2 Gestion des mises

Le système de mise permet au joueur de sélectionner plusieurs types de paris directement sur le tapis de roulette.

Chaque case du tapis correspond à une zone interactive détectée avec la souris.

Le joueur peut notamment miser sur :

— un numéro précis
— une couleur
— pair ou impair
— des groupes de nombres

Chaque mise possède un coefficient différent afin d'équilibrer les probabilités de gain.

Les mises sont automatiquement déduites des Rot Coins du joueur avant le lancement du tirage.

L'objectif était de reproduire le fonctionnement général d'une roulette tout en gardant un système relativement simple à implémenter et à comprendre.

5.1.3 Calcul des gains

Une fois le tirage effectué, le résultat est comparé aux paris placés par le joueur.

Le calcul des gains dépend du type de mise choisi :

— les mises précises offrent un gain élevé
— les mises plus générales offrent des gains plus faibles

Le système applique automatiquement les gains ou les pertes sur l'argent du joueur.

Cette logique permet d'obtenir des parties rapides tout en conservant une forte part d'aléatoire.

Le fonctionnement général suit les étapes suivantes :

1. placement des mises
2. génération aléatoire du résultat
3. comparaison avec les paris
4. calcul des gains ou des pertes

5.1.4 Interface utilisateur

L'interface de la roulette repose principalement sur un tapis interactif.

Le joueur peut cliquer directement sur les différentes zones afin de placer ses mises.

Nous avons essayé de reproduire l'apparence générale des tapis de roulette classiques tout en conservant une cohérence avec le style pixel art du projet.

Plusieurs éléments sont affichés à l'écran :

— montant de la mise
— argent du joueur
— résultat du tirage
— gains obtenus

L'objectif principal était de proposer une interface claire et rapide à utiliser.

5.1.5 Difficultés rencontrées

Le principal problème rencontré concernait la gestion des zones cliquables du tapis.

Chaque case devait être détectée individuellement afin de permettre des interactions précises avec la souris.

Nous avons également rencontré plusieurs difficultés liées à l'alignement visuel des éléments de l'interface et à la gestion des différents types de paris.

L'équilibrage des gains a également demandé plusieurs tests afin d'éviter des parties trop déséquilibrées.

5.2 Implémentation du blackjack

5.2.1 Prototype console

Le blackjack a d'abord été développé sous forme d'un prototype console.

Cette première version permettait principalement de tester :

— la logique des cartes

— le calcul des scores

— les règles du jeu

Le prototype fonctionnait uniquement avec du texte affiché dans le terminal. Même si cette version restait très simple, elle nous a permis de vérifier rapidement les principales mécaniques du blackjack sans devoir développer directement une interface graphique complète.

Cette étape était importante car elle permettait de corriger plus facilement les erreurs de logique avant l'intégration dans le moteur principal du projet.

Nous avons notamment pu tester :

— le tirage des cartes

— le calcul automatique des scores

— les conditions de victoire

— les dépassements de 21

— la gestion des égalités

Ce prototype nous a permis de valider rapidement le fonctionnement du mini-jeu avant son intégration graphique dans le projet principal.

Une fois les mécaniques principales stabilisées, le blackjack a été progressivement intégré au moteur principal du jeu.

5.2.2 Gestion des cartes

Le blackjack repose sur un système classique de gestion des cartes.

Chaque carte possède :

— une valeur

— une couleur

— un symbole

Le calcul du score suit les règles traditionnelles du blackjack. Les figures valent 10 points tandis que l'as peut prendre la valeur 1 ou 11 selon la situation.

Le système gère automatiquement :

— la distribution des cartes

— le calcul des scores

— la détection des dépassements

Le joueur peut ensuite choisir entre :

— tirer une carte

— doubler (tirer une carte et doubler sa mise si il n'a pas encore tiré)

— rester

Les cartes sont stockées dans un deck mélangé aléatoirement au début de chaque partie.

Lorsqu'une carte est distribuée à un joueur ou au croupier, elle est retirée du deck actif afin d'éviter qu'elle puisse être tirée plusieurs fois pendant une même manche.

Cette logique permet de reproduire un fonctionnement proche d'un véritable blackjack.

Le système devait également gérer certains cas particuliers, notamment la gestion des as qui peuvent modifier leur valeur en fonction du score total du joueur.

Le calcul des scores est automatiquement mis à jour après chaque action afin de garder un affichage cohérent pendant la partie.

Une partie importante du travail concernait également l'affichage des cartes à l'écran.

Chaque carte possède son propre sprite et doit être correctement positionnée dans la zone du joueur ou du croupier.

Nous avons essayé de garder une disposition claire afin que les informations restent facilement lisibles même avec plusieurs joueurs.

5.2.3 Gestion des tours

Le blackjack repose sur un système de tours permettant de gérer correctement les actions des différents joueurs.

Chaque joueur joue à son tour avant de laisser la main au joueur suivant puis au croupier.

Pendant son tour, un joueur peut :

— tirer une carte

— rester

— doubler sa mise dans certaines situations

Le système vérifie automatiquement les conditions de fin de tour.

Si un joueur dépasse 21 points, son tour se termine immédiatement.

Une fois tous les joueurs passés, le croupier joue automatiquement selon les règles définies dans le mini jeu.

Cette organisation permet de garder un déroulement clair et compréhensible pendant les parties.

Le système devait également rester suffisamment flexible pour fonctionner en solo comme en multijoueur LAN.

Certaines actions sont donc automatiquement synchronisées entre les joueurs afin de garder le même état de partie sur toutes les machines connectées.

5.2.4 Logique du croupier

Le croupier utilise une logique automatique relativement simple.

Son comportement suit les règles classiques du blackjack :

— tirer tant que le score reste faible

— s'arrête dès qu'il a atteint au moins 17

Cette logique permet de simuler un véritable adversaire sans nécessiter une intelligence artificielle complexe.

Le résultat final dépend ensuite de la comparaison entre le score du joueur et celui du croupier.

Même si le comportement du croupier reste relativement simple, il permet de reproduire correctement le fonctionnement classique du blackjack.

Le système doit également gérer plusieurs situations particulières :

— égalité

— blackjack naturel

— dépassement du joueur

— dépassement du croupier

Les résultats sont ensuite automatiquement affichés à l'écran et les gains du joueur sont calculés en fonction de la mise réalisée au début de la partie.

5.2.5 Mode multijoueur

Un mode multijoueur local a également été envisagé pour le blackjack.

L'objectif était de permettre à plusieurs joueurs de participer à une même partie.

Les joueurs se connectent grâce à une connexion LAN locale.

Le système réseau utilise principalement des échanges simples de données afin de synchroniser :

— les cartes

— les scores

— les actions des joueurs

— l'état de la partie

Cette fonctionnalité représentait également une bonne manière de tester les premiers systèmes réseau du jeu.

Le mode LAN a demandé plusieurs tests afin d'éviter des problèmes de désynchronisation entre les différents joueurs.

Nous avons également dû gérer certaines situations particulières comme les déconnexions ou les joueurs qui terminent leur tour plus rapidement que les autres.

5.2.6 Interface utilisateur

L'interface du blackjack a été pensée afin de rester lisible et cohérente avec le reste du projet.

Le mini jeu utilise une table de blackjack inspirée des casinos traditionnels avec :

— une zone pour le croupier

— plusieurs zones pour les joueurs

— les cartes affichées directement à l'écran

— les informations de score

— les montants des mises

Nous avons essayé de conserver une ambiance chaleureuse et rétro grâce à l'utilisation :

— de couleurs sombres

— de tons rouges et verts

— d'éléments pixel art

Le placement des cartes et des textes a demandé plusieurs ajustements afin de garder une interface claire même avec plusieurs joueurs affichés.

Les différentes actions possibles sont également affichées directement à l'écran afin de rendre le mini jeu plus simple à comprendre.

5.2.7 État final du mini-jeu

Certaines fonctionnalités prévues au départ n'ont pas pu être terminées par manque de temps.

Cependant, les mécaniques principales restent fonctionnelles :

— gestion des cartes

— calcul des scores

— logique du croupier

— système de tours

Le mini-jeu reste donc jouable même si plusieurs améliorations étaient encore prévues après la soutenance.

Il est également possible d'y jouer en multijoueur LAN jusqu'à quatre joueurs simultanément.

Même si certains détails restent perfectibles, le blackjack possède aujourd'hui une structure complète et correctement intégrée dans le reste du casino.

Le mini jeu représente également une bonne démonstration des différents systèmes développés pendant le projet, notamment :

— les interfaces

— la gestion des états

— les systèmes réseau

— la synchronisation des données

5.3 Implémentation du Buckshot Roulette

5.3.1 Concept du mini-jeu

Le Buckshot Roulette est le mini-jeu principal du projet.

Inspiré du jeu du même nom, il repose sur une mécanique de prise de risque permanente.

Le joueur affronte un adversaire dans un duel au tour par tour où chacun peut choisir :

— de tirer sur l'adversaire

— de se tirer dessus

Le gameplay repose sur un mélange de hasard, de stratégie et de tension psychologique.

Ce mini-jeu représente la partie la plus ambitieuse du projet et celle ayant connu la plus forte évolution pendant le développement.

Le Buckshot Roulette est rapidement devenu l'une des fonctionnalités centrales du projet car il possède une identité beaucoup plus forte et plus marquante que les autres mini jeux du casino.

L'objectif était de créer une expérience plus stressante et plus imprévisible afin de casser le rythme plus classique de la roulette ou du blackjack.

Le mini jeu repose volontairement sur une ambiance plus sombre et plus agressive afin de créer un contraste avec le reste du casino.

5.3.2 Règles et fonctionnement

Chaque partie commence par une phase de mise.

Le joueur choisit une somme qui sera immédiatement retirée de son argent.

Le joueur et l'ennemi possèdent chacun plusieurs points de vie.

Le fusil contient un barillet composé :

— d'une balle réelle

— de plusieurs balles à blanc

À chaque tir :

— une balle est consommée

— les dégâts sont éventuellement appliqués

— le barillet peut être rechargé

Lorsqu'une vraie balle est tirée :

— la cible perd un point de vie

— le barillet est rechargé

Si le joueur gagne :

— il récupère le double de sa mise

Sinon :

— la mise est perdue

Le fonctionnement du mini jeu repose volontairement sur des règles simples afin de rendre les parties faciles à comprendre dès les premières secondes.

Même si les mécaniques restent relativement limitées, le système fonctionne principalement grâce à la tension créée par les décisions du joueur.

Le joueur doit constamment choisir entre :

— jouer de manière prudente

— prendre un risque important

Cette logique permet de créer des situations différentes à chaque partie malgré un gameplay relativement simple.

5.3.3 Gestion du barillet

Le système du barillet constitue le cœur du gameplay.

Le barillet est représenté sous forme d'une structure contenant les différentes balles disponibles.

Lors du chargement :

— les balles sont mélangées aléatoirement

— l'ordre reste inconnu du joueur

À chaque tir :

— une balle est retirée du barillet

— le système vérifie si elle est réelle ou non

Cette mécanique permet de maintenir une tension constante pendant toute la partie.

Le joueur doit constamment estimer les probabilités restantes sans jamais disposer d'une certitude totale.

Le système devait également gérer plusieurs situations particulières :

— chargeur vide

— rechargement

— victoire immédiate

— enchaînement des tours

Le contenu du barillet évolue donc constamment pendant la partie, ce qui oblige le joueur à adapter sa stratégie après chaque tir.

Même si le système repose principalement sur le hasard, le joueur conserve toujours une certaine part de réflexion dans ses décisions.

5.3.4 Intelligence artificielle de l'adversaire

Le Buckshot Roulette possède un adversaire contrôlé automatiquement par le jeu.

Cet adversaire utilise une logique de décision relativement simple afin de choisir ses actions pendant la partie.

Selon l'état du jeu, le nombre de vies restantes ou encore les risques liés au tir, l'adversaire peut décider de tirer sur lui même ou sur le joueur.

Le comportement reste volontairement simple mais permet de rendre les parties plus dynamiques et moins prévisibles.

Le but n'était pas de créer une intelligence artificielle complexe, mais plutôt un système suffisamment crédible pour simuler un adversaire et apporter davantage de tension pendant les parties.

L'adversaire peut parfois prendre des décisions risquées, ce qui rend certaines manches plus imprévisibles.

Cette logique permet également d'éviter des comportements trop répétitifs pendant les parties.

Même si le système reste relativement simple, il permet de garder un bon rythme de jeu sans nécessiter une intelligence artificielle avancée.

5.3.5 Gestion des états

Le mini-jeu repose sur plusieurs états permettant de contrôler le déroulement de la partie :

— phase de mise

— tour du joueur

— tour de l'ennemi

— fin de partie

Chaque état possède :

— ses propres interfaces

— ses propres actions

— ses propres animations

Cette organisation permet de simplifier la logique générale du mini-jeu et de garder un fonctionnement plus lisible.

Le système d'états permet également de mieux séparer les différentes phases du gameplay.

Certaines actions ne peuvent être réalisées que pendant des états précis.

Par exemple :

— les mises ne sont possibles que pendant la phase de préparation

— les tirs ne sont possibles que pendant le tour actif

— les résultats ne sont affichés qu'à la fin de la manche

Cette séparation permet d'éviter plusieurs bugs liés aux changements de tours ou aux actions simultanées.

Le mini jeu utilise également plusieurs messages affichés à l'écran afin d'indiquer clairement l'état actuel de la partie.

5.3.6 Gestion des animations

Même si le projet reste relativement simple visuellement, nous avons essayé d'ajouter plusieurs animations afin de rendre le mini jeu plus vivant.

Certaines animations sont utilisées pendant :

— les tirs

— les changements de tour

— les écrans de victoire

— les écrans de défaite

— les transitions

Le système repose principalement sur des changements de sprites et des temporisations simples.

L'objectif principal était surtout de donner davantage de feedback visuel au joueur pendant les parties.

Les animations permettent également de renforcer la tension et de rendre certaines actions plus marquantes.

Même si les animations restent limitées, elles participent fortement à l'identité du mini jeu.

5.3.7 Gameplay et tension

Le Buckshot Roulette repose principalement sur la tension créée par l'incertitude.

Chaque décision peut avoir des conséquences importantes, ce qui pousse le joueur à réfléchir avant chaque action.

Contrairement à la roulette classique, le joueur possède ici un véritable impact sur le déroulement de la partie.

L'alternance entre :

— hasard

— stratégie

— pression psychologique

permet de créer des parties beaucoup plus intenses.

Le mini jeu cherche volontairement à créer une sensation plus stressante et plus agressive que les autres activités présentes dans le casino.

Les sons, les animations et certains éléments visuels participent également à renforcer cette ambiance.

Le joueur ne connaît jamais précisément le contenu restant du barillet, ce qui crée une pression constante à chaque décision.

Même après plusieurs parties, le mini jeu conserve une part importante d'imprévisibilité.

Cette différence d'ambiance permet de rendre le Buckshot Roulette plus identifiable et plus marquant dans le projet final.

5.3.8 Mode multijoueur LAN

Le Buckshot Roulette possède également un mode multijoueur LAN.

Ce mode permet à deux joueurs présents sur le même réseau local de participer à une partie ensemble.

Le système repose sur une architecture relativement simple :

— un joueur crée une partie en tant qu'hôte

— un second joueur rejoint la partie grâce à une adresse IP

Le joueur hôte reste la source principale des informations afin d'éviter les désynchronisations.

Les principales données envoyées sur le réseau sont :

— les actions des joueurs

— l'état du barillet

— les points de vie

— les mises

— les changements de tours

Le système réseau utilise principalement des échanges simples de données JSON.

Même si le mode LAN reste relativement basique, il permet de jouer correctement à plusieurs et représente une fonctionnalité importante du projet final.

Le réseau a demandé plusieurs phases de tests et de corrections, notamment concernant :

— les connexions

— la synchronisation

— les déconnexions

— les mises à jour d'état

5.3.9 Difficultés techniques

Le Buckshot Roulette est probablement le système ayant demandé le plus de travail pendant le projet.

Nous avons rencontré plusieurs difficultés :

— gestion des états

— logique des tours

— affichage dynamique

— synchronisation des animations

— équilibrage du gameplay

Plusieurs bugs d'affichage et de logique sont apparus pendant le développement.

Nous avons dû restructurer une partie importante du code afin d'obtenir un fonctionnement stable et plus lisible.

La modularisation des fonctions a notamment permis de simplifier le débogage et de rendre le système plus maintenable.

Le mode multijoueur LAN a également demandé plusieurs corrections, notamment concernant la synchronisation des informations entre les deux joueurs.

Certains problèmes apparaissaient uniquement pendant les parties réseau, ce qui rendait le débogage plus compliqué.

Nous avons donc dû simplifier certaines parties du système afin de garder un fonctionnement plus stable pour la soutenance finale.

Même si certains détails restent perfectibles, le Buckshot Roulette représente aujourd'hui l'un des systèmes les plus complets du projet.

5.4 Implémentation de l'arcade

5.4.1 Objectif du mode arcade

Le mode arcade avait pour objectif d'ajouter davantage de variété dans le casino.

L'idée était de permettre au joueur d'accéder à plusieurs mini-jeux secondaires directement depuis des bornes d'arcade.

Ce système devait renforcer l'ambiance rétro du projet et rendre le casino plus vivant.

Contrairement aux autres jeux du casino qui reposent principalement sur les mises et le hasard, les bornes d'arcade proposaient des expériences davantage basées sur les réflexes et le score.

L'objectif était également de rappeler les anciennes salles d'arcade présentes dans certains casinos ou bars rétro.

Même si cette partie du projet reste plus simple que les autres mini jeux principaux, elle participe fortement à l'ambiance générale du casino.

5.4.2 Organisation des mini-jeux arcade

Le système arcade repose sur plusieurs mini jeux secondaires accessibles depuis différentes bornes présentes dans le casino.

Chaque borne lance un mini jeu indépendant avec :

— ses propres règles

— ses propres contrôles

— son propre système de score

Cette organisation permettait de proposer plusieurs expériences différentes tout en gardant une structure relativement simple.

Les mini jeux arcade utilisent le même moteur principal que le reste du projet mais possèdent leurs propres logiques internes.

Cela permettait d'ajouter plus facilement de nouveaux contenus sans modifier entièrement le fonctionnement du casino principal.

Le système d'arcade représente également une bonne démonstration de la modularité du projet.

5.4.3 Gameplay du Dodgem

Le principal mini jeu arcade développé repose sur un gameplay de type bullet hell et esquive.

Le joueur contrôle un personnage placé dans une arène et doit survivre le plus longtemps possible en évitant différents projectiles.

Le mini jeu utilise principalement :

— déplacements rapides

— génération aléatoire de projectiles

— gestion des collisions

— système de score basé sur la survie

Des canons placés autour de l'arène tirent régulièrement des projectiles en direction du centre de la map.

Le joueur doit constamment se déplacer afin d'éviter les tirs tout en restant dans les limites de l'arène.

Plus le joueur survit longtemps, plus le multiplicateur de gain augmente.

Le système repose volontairement sur une montée progressive de la difficulté afin d'augmenter la pression pendant la partie.

Ce mini jeu permettait surtout de proposer une expérience plus dynamique et plus rapide que les autres activités du casino.

5.4.4 Gameplay du Pile ou Face

Un autre mini jeu arcade développé repose sur un système de pile ou face.

Le joueur choisit :

— pile

— face

Le résultat est ensuite généré aléatoirement.

Si le choix du joueur correspond au résultat obtenu, le joueur gagne le double de sa mise.

Sinon, la mise est perdue.

Même si ce mini jeu reste très simple, il permet d'ajouter davantage de variété dans le casino et de proposer une expérience très rapide.

Le système utilise principalement :

— gestion des mises

— génération aléatoire

— affichage dynamique des résultats

L'interface du mini jeu reste volontairement minimaliste afin de rappeler certains jeux d'arcade rétro très simples.

5.4.5 Gameplay du Turret

Le mini jeu Turret repose sur un système de survie dans une zone fermée.

Le joueur doit éviter un ennemi contrôlé automatiquement par le jeu pendant une durée limitée.

Le gameplay repose principalement sur :

— les déplacements

— l'esquive

— les collisions

— la survie

Le mini jeu utilise également plusieurs obstacles placés dans la map afin de créer un petit labyrinthe.

L'ennemi poursuit constamment le joueur, ce qui oblige à se déplacer en permanence.

Le joueur gagne la partie s'il survit suffisamment longtemps.

Ce mini jeu demandait une gestion plus importante des collisions et des déplacements que les autres bornes d'arcade.

Le système utilise également plusieurs sprites animés pour le joueur et l'ennemi.

5.4.6 Gestion des collisions

Les mini jeux arcade reposent fortement sur la gestion des collisions.

Dans le Dodgem, les collisions servent principalement à détecter :

— les impacts avec les projectiles

— les limites de l'arène

Dans le Turret, les collisions sont également utilisées pour gérer :

— les murs

— les obstacles

— les déplacements du gardien

— les contacts avec le joueur

Ces systèmes étaient importants afin de garder des déplacements fluides et cohérents pendant les parties.

Certaines collisions ont demandé plusieurs ajustements afin d'éviter des comportements incorrects ou des blocages dans les maps.

5.4.7 Interface utilisateur

Les mini jeux arcade possèdent chacun leurs propres interfaces adaptées à leur gameplay.

Le Dodgem affiche notamment :

— le temps de survie

— le multiplicateur

— les informations de mise

Le Pile ou Face utilise une interface beaucoup plus simple avec :

— le montant de la mise

— les boutons pile et face

— l'affichage du résultat

Le Turret affiche principalement :

— le temps restant

— les informations de mise

— les messages de victoire ou défaite

Nous avons essayé de garder une cohérence visuelle avec le reste du projet grâce :

— au pixel art

— aux mêmes polices

— aux mêmes couleurs générales

5.4.8 Difficultés rencontrées

Le développement du mode arcade s'est révélé plus complexe que prévu.

Chaque mini jeu possédait pratiquement sa propre logique, ce qui demandait de développer plusieurs systèmes différents dans un temps relativement limité.

Plusieurs problèmes sont apparus :

— gestion des performances

— génération des projectiles

— équilibrage de la difficulté

— collisions

— synchronisation des déplacements

— manque de temps pour le contenu graphique

Le système Dodgem demandait par exemple de gérer plusieurs projectiles simultanément tout en gardant un framerate stable.

Le Turret demandait également plusieurs ajustements concernant les déplacements du gardien et les collisions avec les obstacles.

Même si les mini jeux arcade restent moins avancés que les autres systèmes principaux du projet, ils apportent davantage de variété et renforcent l'ambiance rétro du casino.

Ils permettent également de montrer la modularité du moteur principal et la capacité du projet à intégrer plusieurs styles de gameplay différents.

6 Mode multijoueur et réseau

6.1 Objectifs du mode LAN

L'un des objectifs du projet était d'ajouter un mode multijoueur local afin de permettre à plusieurs joueurs de jouer ensemble sur certains mini jeux.

Nous avons choisi de réaliser un système LAN simple, fonctionnant sur un même réseau local. L'objectif n'était pas de créer une infrastructure réseau complexe, mais plutôt de proposer une fonctionnalité jouable et cohérente avec le reste du projet.

Le mode réseau devait permettre :

— de créer une partie en tant qu'hôte,

— de rejoindre une partie via une adresse IP,

— de synchroniser les actions des joueurs,

— de partager l'état d'une partie entre plusieurs machines.

Cette fonctionnalité est principalement être utilisée sur les mini jeux compatibles comme le Buckshot Roulette et le Blackjack.

Le choix du réseau local permet également de garder une architecture relativement simple et adaptée à notre niveau, tout en ajoutant une dimension multijoueur intéressante au projet.

6.2 Architecture client / serveur

Le système réseau repose sur une architecture classique client / serveur.

Un joueur héberge la partie et devient le serveur, tandis que l'autre joueur rejoint la partie en tant que client.

Le serveur est responsable de :

— accepter les connexions,

— maintenir l'état principal de la partie,

— envoyer les informations importantes au client,

— gérer certaines actions du gameplay.

Le client, lui, se connecte au serveur et reçoit régulièrement les informations nécessaires à la synchronisation de la partie.

Le système a été développé avec le module `socket` de Python. Deux classes principales ont été créées :

— `BuckshotServer`

— `BuckshotClient`

Le serveur ouvre un socket TCP sur un port défini et attend la connexion d'un joueur.

Une fois connecté, le client peut envoyer des messages simples au serveur afin de transmettre ses actions en jeu.

Nous avons volontairement gardé une architecture simple afin de conserver un code lisible et facilement débogable.

Le système utilise également un fichier commun permettant de centraliser certaines informations réseau, comme le port utilisé ou la récupération de l'adresse IP locale.

6.3 Synchronisation des données

La synchronisation entre les joueurs repose principalement sur l'échange de messages texte.

Chaque message est envoyé sous forme de chaîne de caractères, suivie d'un retour à la ligne afin de pouvoir séparer facilement les différents paquets réseau.

Les messages sont stockés temporairement dans un buffer, puis découpés ligne par ligne afin de récupérer les informations correctement même lorsque plusieurs messages arrivent en même temps.

Cette méthode permettait de garder une logique simple sans devoir mettre en place un protocole réseau complexe.

Les informations synchronisées concernent principalement :

— les actions des joueurs,

— certains états du mini jeu,

— les débuts et fins de partie,

— les déconnexions.

Dans le Buckshot Roulette, cela permet par exemple de transmettre les choix des joueurs, les tirs effectués ou encore certains changements d'état.

Le système a été pensé pour être facilement réutilisable dans d'autres mini jeux comme le Blackjack.

6.4 Gestion des connexions

Le serveur démarre en ouvrant un socket TCP sur le port défini dans le projet. Il attend ensuite qu'un client tente de rejoindre la partie.

Le client doit entrer l'adresse IP de l'hôte afin d'établir la connexion.

Une fois la connexion établie :

— les deux joueurs peuvent échanger des messages,

— les sockets passent,

— les informations sont traitées en continu pendant la partie.

Nous avons également ajouté plusieurs sécurités simples afin d'éviter certains crashs lors des déconnexions ou des erreurs réseau.

Lorsque la connexion est interrompue, le système détecte automatiquement la fermeture du socket et ferme proprement la session réseau.

Même si le système reste relativement basique, cela permet déjà d'obtenir un mode multijoueur fonctionnel et stable dans la majorité des cas.

6.5 Tests réseau réalisés

Plusieurs tests ont été réalisés durant le développement afin de vérifier la stabilité du système réseau.

Nous avons testé :

— la connexion entre deux machines différentes,

— les échanges de messages,

— les déconnexions,

— la synchronisation des actions,

— les transitions entre différents états de jeu.

Les tests ont été réalisés principalement sur réseau local, avec plusieurs ordinateurs connectés au même WiFi.

Nous avons également testé plusieurs cas d'erreurs, comme :

— une mauvaise adresse IP,

— la fermeture brutale d'un joueur,

— l'arrêt du serveur pendant une partie.

Ces tests nous ont permis de corriger plusieurs problèmes liés aux buffers, aux sockets bloquants ou aux déconnexions inattendues.

6.6 Difficultés rencontrées

Le réseau a été l'une des parties les plus compliquées du projet et probablement celle qui nous a demandé le plus de temps pendant le développement.

Contrairement aux autres systèmes du jeu, les problèmes réseau étaient souvent difficiles à reproduire et pouvaient apparaître de manière aléatoire selon :

— les machines utilisées,

— la qualité de la connexion locale,

— l'ordre des actions des joueurs,

— les timings des échanges réseau.

Au départ, plusieurs problèmes apparaissaient régulièrement :

— pertes de synchronisation,

— erreurs de connexion,

— messages non reçus,

— blocages du programme,

— déconnexions inattendues.

Certaines erreurs étaient particulièrement compliquées à corriger car elles ne se produisaient pas systématiquement. Il arrivait par exemple qu'une partie fonctionne correctement plusieurs fois avant qu'un problème apparaisse soudainement sans raison visible.

Nous avons également rencontré plusieurs difficultés avec la gestion des sockets non bloquants, car certaines fonctions pouvaient provoquer des erreurs lorsqu'aucune donnée n'était reçue.

Le traitement des messages réseau a aussi demandé plusieurs essais avant d'obtenir quelque chose de stable.

Nous avons dû tester plusieurs méthodes différentes pour gérer correctement :

— l'envoi des données,

— la réception des messages,

— le découpage des paquets,

— les fermetures de connexion.

Une autre difficulté importante concernait la synchronisation des états du mini jeu.

Il fallait s'assurer que les deux joueurs voyaient exactement la même situation au même moment, ce qui demandait une gestion précise des échanges réseau.

Par exemple, lorsqu'un joueur tirait dans le Buckshot Roulette, les deux machines devaient :

— afficher la même action,

— utiliser le même état du barillet,

— appliquer les mêmes dégâts,

— changer de tour au bon moment.

Le moindre décalage pouvait provoquer des désynchronisations et casser complètement la partie.

Le réseau nous a également obligés à restructurer plusieurs parties du code afin de mieux séparer :

— la logique locale,

— les données envoyées,

— les mises à jour réseau.

Cette partie du projet nous a demandé beaucoup plus de temps que prévu initialement.

Nous avons passé une grande partie du développement à tester, corriger puis retester le système réseau afin d'obtenir quelque chose de suffisamment stable pour la soutenance finale.

Même si le système reste relativement simple, le fait d'avoir réussi à intégrer un mode LAN fonctionnel reste l'une des plus grosses réussites techniques du projet.

6.7 Résultat final

Le résultat final permet aujourd'hui de jouer en réseau local sur les mini jeux compatibles du projet.

Le système reste volontairement simple, mais il est fonctionnel et suffisamment stable pour être utilisé durant les démonstrations.

Un joueur peut héberger une partie, transmettre son adresse IP et permettre à un autre joueur de rejoindre la session.

Les échanges de données fonctionnent correctement dans la majorité des situations testées, et le système peut être réutilisé pour d'autres fonctionnalités multijoueur du projet.

Même si certaines améliorations restent possibles, ce mode LAN représente une étape importante du projet, car il ajoute une véritable dimension multijoueur à l'expérience proposée par le jeu.

7 Création du site web

7.1 Objectifs du site

Dans le cadre du projet, nous devions également réaliser un site web afin de présenter le jeu, le groupe ainsi que les différents livrables demandés pour la soutenance finale.

Le site avait plusieurs objectifs principaux. Le premier était de permettre une présentation claire et accessible du projet. Nous voulions que les visiteurs puissent rapidement comprendre l'univers du jeu, son fonctionnement et les différentes fonctionnalités présentes dans le casino.

Le second objectif concernait le téléchargement des différents fichiers. Le site devait permettre d'accéder facilement au jeu, au rapport technique ainsi qu'aux autres documents liés à la soutenance.

Nous voulions également utiliser le site comme support de communication autour du projet. Il permet de présenter les membres du groupe, leurs rôles respectifs et leur participation au développement du jeu.

Enfin, le site devait rester cohérent avec l'identité visuelle du projet. Nous avons donc essayé de reprendre les couleurs, les ambiances et le style rétro déjà présents dans le jeu.

Le développement du site a également permis de découvrir plusieurs aspects du développement web, notamment l'organisation des pages HTML, la mise en page CSS et l'adaptation du site sur différentes tailles d'écran.

7.2 Présentation des pages

Le site est organisé autour de plusieurs sections principales accessibles depuis une barre de navigation située en haut de la page.

La première section correspond à la page d'accueil. Elle présente rapidement l'univers du projet ainsi qu'une courte introduction au jeu. Cette partie sert principalement à donner une première impression visuelle du projet grâce à une grande bannière et à plusieurs éléments graphiques reprenant l'ambiance du casino rétro.

Une section de téléchargement permet ensuite d'accéder directement aux fichiers du projet. Depuis cette partie, les utilisateurs peuvent récupérer l'installateur du jeu ainsi que le rapport de soutenance.

Le site contient également une section dédiée à la présentation du projet. Cette partie résume les principales fonctionnalités du jeu :

— exploration du casino

— mini jeux

— ambiance rétro

— univers pixel art

Une autre section est consacrée à la présentation des membres du groupe. Chaque membre possède une carte contenant son nom, son rôle principal ainsi qu'un résumé de sa contribution au projet.

Enfin, le site contient une page présentant les différentes ressources et bibliothèques utilisées pendant le développement :

— Pygame

— PyTMX

— PyScroll

— Tiled

L'objectif général du site était surtout de proposer une présentation simple, claire et cohérente avec le reste du projet.

7.3 Téléchargement du jeu

Le site permet de télécharger le projet directement depuis une page dédiée. Nous avons choisi de séparer les différentes versions afin de rendre le téléchargement plus clair pour les utilisateurs.

Plusieurs plateformes étaient prévues :

— Windows

— Linux

— macOS

Même si la version principale du projet reste celle destinée à Windows, nous voulions préparer une structure capable d'accueillir plusieurs versions du jeu. Finalement nous n'avons fait que la version Windows.

La version Windows repose sur un exécutable généré avec PyInstaller ainsi qu'un installateur créé avec Inno Setup afin de permettre une installation plus simple du projet sans avoir besoin d'installer Python manuellement.

Le site devait également permettre de récupérer les sources du projet, les assets et les différents fichiers utiles à la soutenance.

L'organisation des téléchargements a été pensée de manière simple afin de permettre un accès rapide aux différents documents sans navigation complexe.

7.4 Téléchargement du rapport

Le rapport technique final est également accessible directement depuis le site web. Cette partie permet de centraliser tous les documents importants du projet sur une même plateforme.

Le téléchargement du rapport permet notamment d'accéder :

— aux explications techniques

— à l'organisation du projet

— aux choix de développement

— aux difficultés rencontrées

— aux annexes

Nous avons choisi de rendre cette partie facilement accessible depuis la page principale afin de simplifier la récupération des documents pendant la soutenance.

Le rapport est proposé au format PDF afin de garantir une compatibilité simple sur la majorité des plateformes.

7.5 Hébergement

Le site web est hébergé à l'aide de la plateforme Hostinger. Cette solution nous a permis de mettre le site en ligne rapidement et de disposer d'un hébergement stable pour présenter le projet.

L'utilisation de Hostinger permet également de rendre le site accessible depuis n'importe quel navigateur sans installation particulière.

Le site reste volontairement léger afin de limiter les temps de chargement et de garder une navigation fluide même sur des connexions plus faibles.

L'hébergement du site nous a aussi permis de mieux comprendre le fonctionnement général d'une mise en ligne de projet web, notamment la gestion des fichiers HTML, CSS et des assets.

7.6 Ergonomie et identité visuelle

L'identité visuelle du site reprend directement les choix graphiques présents dans le jeu. Nous avons essayé de garder une cohérence entre le site et l'univers général de *Rot it or Lose it*.

Le site utilise principalement :

— des couleurs sombres

— des tons orangés

— des éléments pixel art

— une police rétro inspirée des jeux d'arcade

Plusieurs éléments visuels du jeu ont été réutilisés dans le site, comme les logos, les icônes ou certains assets graphiques.

Nous avons également essayé de rendre le site agréable à parcourir grâce à une navigation simple et des sections bien séparées.

Une attention particulière a été portée à l'adaptation du site sur différentes tailles d'écran. Plusieurs media queries CSS ont été utilisées afin d'améliorer l'affichage sur tablette et téléphone.

Même si le site reste relativement simple techniquement, il permet de présenter efficacement le projet et de regrouper tous les éléments importants demandés pour la soutenance finale.

8 Création de l'exécutable et installation

8.1 Création du build

Afin de permettre un lancement du jeu sans installation manuelle de Python, nous avons généré un exécutable Windows grâce à PyInstaller.

Cet outil permet de regrouper :

— le code Python,

— les bibliothèques utilisées,

— les assets,

— les dépendances nécessaires au fonctionnement du jeu.

Le build a été réalisé principalement pour Windows, qui représentait la plateforme principale utilisée pendant le développement.

La génération de l'exécutable avec PyInstaller et la création de l'installateur avec Inno Setup ont demandé plusieurs ajustements, notamment concernant :

— les chemins des assets,

— les fichiers audio,

— les maps TMX,

— certaines dépendances Pygame,

— l'intégration correcte des fichiers dans le build final.

Nous avons également dû adapter plusieurs chemins du projet afin de garantir un fonctionnement correct après compilation et après installation via l'installateur Windows.

Le build final permet de lancer directement le jeu grâce à un fichier exécutable installé automatiquement, sans installation supplémentaire de Python sur la machine utilisateur.

L'installateur permet également :

— d'ajouter un raccourci sur le bureau,

— d'installer automatiquement les fichiers du projet,

— de désinstaller correctement le jeu depuis Windows.

8.2 Organisation des fichiers

Le projet final est organisé en plusieurs dossiers afin de séparer clairement les différents éléments du jeu.

Le dossier principal contient :

— l'exécutable du jeu,

— les assets graphiques,

— les fichiers audio,

— les maps,

— les sauvegardes,

— les bibliothèques nécessaires.

Les assets sont eux-mêmes organisés par catégories :

— sprites,

— interfaces,

— musiques,

— effets sonores,

— tilesets.

Cette organisation permet de simplifier :

— le développement,

— la maintenance,

— le débogage,

— l'ajout de nouveaux contenus.

Les sauvegardes sont stockées séparément afin de permettre une gestion plus simple des données joueur.

8.3 Procédure d'installation

L'installation du projet reste volontairement simple afin de permettre un accès rapide au jeu.

L'utilisateur doit :

1. télécharger l'installateur du jeu,
2. lancer le fichier d'installation,
3. suivre les différentes étapes proposées par Inno Setup.

L'installateur créé avec Inno Setup permet automatiquement :

— de copier les fichiers du jeu,

— de créer les raccourcis nécessaires,

— d'installer l'exécutable du projet,

— de préparer la désinstallation Windows.

Toutes les dépendances nécessaires sont directement intégrées dans le build généré avec PyInstaller.

Aucune installation manuelle de Python ou de bibliothèques supplémentaires n'est nécessaire.

Le projet a principalement été testé sur Windows 10 et Windows 11.

8.4 Procédure de désinstallation

Le projet peut être désinstallé directement depuis Windows grâce au système de désinstallation généré par Inno Setup.

L'utilisateur peut :

— utiliser le désinstallateur présent dans le dossier du jeu,

— ou passer par les paramètres Windows et la liste des applications installées.

La désinstallation supprime automatiquement les principaux fichiers du projet installés sur la machine.

Les sauvegardes locales peuvent également être supprimées manuellement si l'utilisateur souhaite effacer totalement sa progression.

Le projet ne modifie pas les fichiers système importants de Windows et ne nécessite donc aucune procédure complexe de nettoyage.

8.5 Compatibilité et tests

Plusieurs tests ont été réalisés pendant le développement afin de vérifier la stabilité et les performances du projet.

Le jeu a principalement été testé sur :

— différents ordinateurs Windows,

— plusieurs résolutions d'écran,

— plusieurs configurations matérielles.

Les tests ont permis de corriger :

— certains bugs d'affichage,

— des problèmes de collisions,

— des erreurs liées au chargement des assets,

— plusieurs problèmes réseau.

Des tests spécifiques ont également été réalisés sur le mode multijoueur LAN afin de vérifier :

— la stabilité des connexions,

— la synchronisation des actions,

— la gestion des déconnexions.

Même si certains bugs mineurs restent présents, le projet final reste globalement stable et jouable.

9 Organisation du projet

9.1 Répartition des tâches

Le projet a été réalisé en groupe avec une répartition des tâches définie dès le début du développement. Chaque membre possédait un rôle principal ainsi qu'un rôle secondaire afin de pouvoir aider sur plusieurs parties du projet lorsque cela était nécessaire.

Flavien s'est principalement occupé de la structure technique du jeu, notamment des bases du moteur, des collisions, des déplacements, du système de sauvegarde ainsi qu'une partie importante du Buckshot Roulette et du réseau local.

Oscar a travaillé principalement sur les interfaces utilisateur, les menus, le style visuel du projet ainsi qu'une partie du blackjack, de la roulette et du site web.

Lucas s'est occupé principalement des maps, des transitions, des collisions liées aux environnements ainsi que de plusieurs éléments visuels présents dans le casino.

Amadou a travaillé sur la partie sonore du projet avec la création des musiques et de l'ambiance audio du casino.

Même si chaque membre possédait des responsabilités principales, le projet a demandé beaucoup de travail collectif. Plusieurs systèmes ont été développés à plusieurs afin de résoudre certains problèmes techniques ou accélérer certaines phases du développement.

Au cours des différentes soutenances, les objectifs ont évolué en fonction de l'avancement réel du projet. Certaines fonctionnalités ont été davantage développées que prévu, tandis que d'autres ont été simplifiées ou abandonnées.

Le jeu de cartes personnalisé, par exemple, a finalement été abandonné suite au départ d'un membre du groupe ainsi qu'au manque de temps rencontré pendant la fin du développement. Nous avons préféré concentrer le travail sur les éléments les plus importants du projet comme les maps, les mini jeux et le réseau local.

Globalement, nous avons essayé de respecter au maximum les objectifs fixés lors des différentes soutenances même si certaines fonctionnalités se sont révélées plus complexes que prévu.

9.2 Organisation des réunions

L'organisation du groupe reposait principalement sur des réunions régulières afin de suivre l'avancement du projet et répartir les tâches entre les membres.

La majorité de ces réunions étaient organisées et préparées par Oscar. Ces moments permettaient de faire le point sur les fonctionnalités terminées, les problèmes rencontrés ainsi que les objectifs à atteindre avant les différentes soutenances.

Les réunions servaient également à coordonner le travail lorsque plusieurs personnes devaient intervenir sur les mêmes fichiers ou les mêmes systèmes.

Pendant les périodes importantes du projet, notamment avant les soutenances, la fréquence des réunions augmentait afin de pouvoir corriger rapidement les bugs et finaliser les fonctionnalités principales.

9.3 Utilisation de Trello

Afin de mieux organiser le projet, nous avons utilisé Trello pendant une grande partie du développement.

Cet outil nous permettait principalement de répartir les tâches, de suivre leur avancement et de visualiser les priorités du moment.

Les différentes fonctionnalités du projet étaient séparées sous forme de cartes contenant :

— la tâche à réaliser

— le responsable principal

— les objectifs

— l'état d'avancement

L'utilisation de Trello nous a aidés à mieux répartir le travail entre les membres du groupe et à garder une trace des fonctionnalités déjà réalisées ou encore en cours de développement.

Nous avons également utilisé cet outil pour préparer les différentes soutenances et définir les objectifs à atteindre avant chaque rendu.

Même si toutes les tâches prévues au départ n'ont pas été totalement finalisées, Trello a permis de mieux structurer le développement du projet sur plusieurs mois.

9.4 Communication du groupe

La communication occupait une place importante pendant le développement du projet.

Le groupe utilisait principalement Instagram ou Discord afin d'échanger rapidement sur les problèmes techniques, partager des captures d'écran, envoyer des fichiers ou encore organiser les réunions.

Une grande partie du travail se faisait en parallèle. Il était donc important de communiquer régulièrement afin d'éviter les conflits de code ou les problèmes liés aux différentes versions du projet.

La communication était particulièrement importante pendant les phases de débogage. Lorsqu'un problème apparaissait sur une fonctionnalité, plusieurs membres pouvaient intervenir ensemble afin de trouver une solution plus rapidement.

Nous avons également dû communiquer régulièrement sur les priorités du projet, notamment lorsque certaines fonctionnalités prenaient plus de temps que prévu.

Même si certains moments ont été plus compliqués, notamment pendant les périodes de forte charge de travail, la communication a permis de maintenir une progression globale du projet jusqu'à la soutenance finale.

9.5 Gestion des imprévus

Comme dans la majorité des projets de groupe, nous avons rencontré plusieurs imprévus pendant le développement.

Certaines fonctionnalités prévues au départ se sont révélées plus complexes que prévu, notamment le réseau local et plusieurs systèmes liés aux mini jeux.

Le développement du Buckshot Roulette a par exemple demandé beaucoup plus de temps que prévu à cause de la gestion des états, des animations et des interactions entre les différentes mécaniques du jeu.

Le réseau local a également posé plusieurs problèmes techniques et a demandé beaucoup de temps avant d'obtenir un système suffisamment stable et fonctionnel.

Nous avons aussi rencontré des difficultés liées à l'organisation du code. Plusieurs personnes travaillaient parfois sur les mêmes fichiers, ce qui entraînait certains conflits lors des fusions Git.

Le départ d'un membre du groupe a également obligé l'équipe à réorganiser plusieurs tâches prévues initialement.

Face à ces difficultés, nous avons dû adapter plusieurs objectifs afin de concentrer le développement sur les parties les plus importantes du projet.

Certaines fonctionnalités secondaires ont donc été simplifiées, repoussées ou abandonnées afin de garantir un résultat final stable et présentable pour la soutenance.

Même si tous les objectifs initiaux n'ont pas été atteints entièrement, le projet final reste beaucoup plus complet et abouti que les premières versions réalisées au début de l'année.

10 Bilan individuel des contributions

10.1 Contribution de Flavien

Depuis le début du projet, Flavien s'est principalement occupé de toute la partie technique centrale du jeu.

Il a développé les bases du moteur principal, notamment la boucle de jeu, la gestion des déplacements, les collisions, les animations du joueur, les transitions entre les maps ainsi que l'intégration globale des différents systèmes du projet.

Il a également travaillé sur la structure générale du code afin de garder une architecture claire et modulaire. Cela a permis de faciliter l'intégration des mini jeux et le travail des autres membres du groupe.

Flavien s'est aussi occupé du système de sauvegarde du jeu. Un système de sauvegarde en JSON a été mis en place afin de conserver les informations importantes comme l'argent du joueur ou sa position dans les maps.

Il a également travaillé sur le mini jeu d'arcade, avec notamment la logique de gameplay du mode bullet hell et la gestion des collisions et projectiles.

Une grande partie du développement du Buckshot Roulette a également été réalisée par lui. Il a participé à la gestion des états de jeu, du système de mise, des points de vie, des animations et du déroulement global des parties.

Enfin, il a participé au développement du réseau local LAN. Cette partie a demandé plusieurs essais avant d'obtenir quelque chose de fonctionnel et stable dans le projet.

Il s'est également occupé de la génération de l'exécutable final du jeu et de plusieurs phases de debug importantes durant le projet.

10.2 Contribution d'Oscar

Oscar a principalement travaillé sur les interfaces du jeu, les éléments visuels ainsi que l'organisation générale du projet.

Il s'est occupé du menu principal, des différents menus secondaires, des boutons, des interfaces de mini jeux ainsi que du HUD affichant les informations importantes comme l'argent du joueur.

Il a également créé presque tous les éléments du tileset du jeu afin de garder une cohérence graphique entre les maps et les mini jeux. Ce travail a demandé beaucoup de temps afin de conserver un style pixel art homogène sur l'ensemble du projet.

Oscar a aussi participé à l'intégration du Blackjack dans le projet. Il a travaillé sur la logique du jeu, et une partie de l'intégation visuelle.

Concernant la roulette, il a réalisé une première base technique du tapis de jeu avec les zones cliquables et la structure générale des mises. Lucas a ensuite poursuivi le développement de la logique complète du mini jeu.

Il s'est également occupé de l'organisation du groupe. Il gérait régulièrement les réunions, la répartition des tâches, le suivi de l'avancement ainsi que la coordination entre les membres.

Enfin, Oscar a créé et hébergé le site web du projet. Le site permet de présenter le jeu, les membres du groupe, les téléchargements et les différentes ressources utilisées.

10.3 Contribution de Lucas

Lucas s'est principalement occupé de la création des maps et d'une partie de l'ambiance visuelle du projet.

Il a conçu les différentes salles du casino avec le logiciel Tiled, notamment la salle principale, le bar et la salle du Buckshot Roulette.

Il a également configuré les collisions, les zones interactives, les transitions entre les maps et les différents points d'entrée. Ce travail était important afin de rendre les environnements jouables et cohérents avec le reste du projet.

Lucas a aussi créé plusieurs éléments graphiques présents dans les maps, comme certaines tables, décorations ou éléments du bar, afin de renforcer l'ambiance générale du casino.

Concernant les mini jeux, il s'est principalement occupé du développement de la roulette. Il a développé le système de mises, le tirage aléatoire, la gestion des gains et l'inté-

gration du mini jeu dans le casino.

Il a également participé à certaines phases de tests et d'intégration, notamment sur les maps et les collisions.

Le projet de jeu de cartes personnalisé devait aussi être développé en partie par Lucas, mais cette fonctionnalité a finalement été abandonnée afin de se concentrer sur les éléments principaux du projet.

10.4 Contribution d'Amadou

Amadou a principalement travaillé sur la partie sonore du projet.

Il s'est occupé de la recherche et de l'intégration des musiques et effets sonores afin de créer une ambiance cohérente avec l'univers du casino et des mini jeux.

Plusieurs sons ont été ajoutés afin de renforcer l'immersion, notamment dans les menus et certains mini jeux.

Il a également participé à plusieurs phases de tests du projet, ainsi qu'à certaines tâches secondaires liées aux interfaces et à l'intégration générale.

Amadou a aussi participé aux discussions et réunions du groupe afin de suivre l'avancement du projet et aider à la prise de certaines décisions concernant l'ambiance du jeu.

11 Difficultés rencontrées

11.1 Difficultés techniques

Le projet a présenté plusieurs difficultés techniques importantes, notamment à cause du nombre de systèmes différents présents dans le jeu.

La gestion des collisions dans les maps a demandé beaucoup de temps, car certaines hitbox complexes étaient mal interprétées par le moteur. Nous avons donc dû simplifier plusieurs collisions afin de garantir un comportement stable.

L'intégration des mini jeux dans une même architecture a également été compliquée. Chaque mini jeu possède sa propre logique, ses propres interfaces et ses propres systèmes d'interaction. Il a donc fallu adapter l'architecture afin de garder un fonctionnement cohérent dans l'ensemble du projet.

Nous avons également rencontré plusieurs bugs liés aux interfaces, au rendu graphique ou encore aux transitions entre les maps, ce qui a nécessité de nombreuses phases de correction et de tests.

11.2 Problèmes d'intégration

L'intégration des différentes parties du projet a parfois été compliquée, car plusieurs membres travaillaient en parallèle sur des systèmes différents.

Certaines fonctionnalités pouvaient fonctionner séparément mais créer des problèmes une fois intégrées dans le projet principal. Cela a notamment été le cas avec certaines interfaces ou certains mini jeux.

Nous avons donc dû régulièrement restructurer certaines parties du code afin de garder un projet stable et cohérent.

Le travail sur Git et les merges ont aussi parfois provoqué quelques conflits ou erreurs nécessitant des corrections manuelles.

11.3 Gestion du réseau

Le réseau local LAN a été l'une des parties les plus compliquées du projet.

Au départ, plusieurs tentatives ne fonctionnaient pas correctement ou provoquaient des problèmes de synchronisation entre les joueurs.

La gestion des échanges de données, des états de jeu et des interactions entre les deux clients a demandé beaucoup de tests et plusieurs réécritures de certaines parties du système.

Le réseau local reste aujourd'hui fonctionnel, mais cette partie a demandé bien plus de temps que prévu au départ.

11.4 Gestion du temps

L'organisation du temps a été un défi important durant le projet.

Certaines fonctionnalités ont demandé beaucoup plus de travail que ce que nous avions prévu au début du développement, notamment le réseau, les mini jeux et certaines interfaces.

Nous avons donc dû revoir certaines priorités au fil des soutenances afin de nous concentrer sur les éléments les plus importants du projet.

Certaines idées ou fonctionnalités secondaires ont été repoussées ou abandonnées afin de garantir un jeu stable et jouable.

11.5 Organisation du groupe

L'organisation du groupe s'est progressivement améliorée durant le projet.

Des réunions régulières ont été mises en place afin de suivre l'avancement des tâches et répartir le travail entre les membres.

Nous avons utilisé plusieurs outils comme Discord, Instagram, Google Sheets et Trello afin de communiquer et suivre l'évolution du projet.

Oscar s'est principalement occupé de la gestion des réunions, de l'organisation générale et du suivi des tâches.

Même si certains moments ont été plus compliqués, la communication globale du groupe est restée correcte pendant la majorité du développement.

11.6 Adaptation suite au départ d'un membre

Le départ d'un membre du groupe a eu un impact important sur le projet.

Certaines tâches prévues initialement ont dû être redistribuées entre les autres membres, ce qui a augmenté la charge de travail globale.

Le projet de jeu de cartes personnalisé a notamment été abandonné, car il représentait une fonctionnalité secondaire qui demandait encore beaucoup de développement.

Nous avons donc décidé de recentrer nos efforts sur les parties les plus importantes du projet, comme les mini jeux principaux, les maps, le réseau local et la stabilité générale du jeu.

12 Tests et validation

12.1 Tests gameplay

Les tests gameplay ont été réalisés tout au long du développement afin de vérifier que les différentes mécaniques du jeu fonctionnaient correctement.

Nous avons principalement testé :

— les déplacements du joueur,

— les interactions avec les éléments du casino,

— le lancement des mini jeux,

— le retour au casino après une partie,

— la gestion de l'argent du joueur,

— les conditions de victoire et de défaite.

Chaque mini jeu a été testé séparément avant d'être intégré dans le projet principal. Cela nous a permis de vérifier plus facilement si les bugs venaient du mini jeu lui-même ou de son intégration dans le casino.

La roulette a par exemple été testée avec plusieurs types de mises afin de vérifier le calcul des gains et des pertes. Nous avons aussi testé les cas où le joueur n'avait pas assez d'argent pour miser.

Le blackjack a été testé avec plusieurs situations différentes :

— victoire du joueur,

— victoire du croupier,

— égalité,

— dépassement de 21,

— gestion de l'as,

— mode multijoueur.

Le Buckshot Roulette a demandé davantage de tests car il repose sur plusieurs états de jeu différents. Nous avons testé les phases de mise, les tours du joueur, les tours de l'adversaire, la gestion des balles et la fin de partie.

Ces tests ont permis de corriger plusieurs erreurs de logique et d'améliorer la stabilité globale du jeu.

12.2 Tests réseau

Les tests réseau ont été une partie importante du projet car le mode LAN était l'une des fonctionnalités les plus difficiles à stabiliser.

Nous avons testé la connexion entre plusieurs ordinateurs présents sur le même réseau local. L'objectif était de vérifier que l'hôte pouvait créer une partie et que les autres joueurs pouvaient la rejoindre grâce à l'adresse IP.

Les principaux éléments testés étaient :

— la création d'une partie en tant qu'hôte,

— la connexion d'un ou plusieurs clients,

— l'envoi des actions des joueurs,

— la réception des données,

— la synchronisation des états de jeu,

— la gestion des déconnexions.

Plusieurs problèmes sont apparus pendant ces tests, notamment des messages non reçus, des états désynchronisés ou des connexions qui se fermaient de manière inattendue.

Nous avons donc simplifié le fonctionnement du réseau afin de garder un système compréhensible et plus stable. Le serveur reste la source principale des informations, ce qui permet d'éviter que chaque joueur possède une version différente de la partie.

Le mode LAN reste simple, mais il permet aujourd'hui de jouer en réseau local sur les mini jeux compatibles.

12.3 Tests des sauvegardes

Le système de sauvegarde a été testé afin de vérifier que les données du joueur étaient bien conservées entre deux sessions de jeu.

Les données principales sauvegardées sont :

— la position du joueur,

— l'argent du joueur.

Nous avons testé plusieurs situations :

— sauvegarder avant de quitter le jeu,

— relancer le jeu et charger la sauvegarde,

— modifier l'argent après un mini jeu,

— vérifier que les données restent cohérentes après plusieurs parties.

Les sauvegardes étant stockées dans des fichiers JSON, il était également possible de vérifier manuellement leur contenu pendant les tests.

Ce système nous a permis de mieux contrôler la progression du joueur et d'éviter de recommencer une partie depuis le début à chaque lancement du jeu.

12.4 Tests des collisions

Les collisions ont été testées directement dans les différentes maps du casino.

L'objectif était de vérifier que le joueur ne pouvait pas traverser :

— les murs,

— les tables,

— les comptoirs,

— les décorations importantes,

— les limites des salles.

Nous avons aussi testé les zones de transition entre les maps afin de vérifier que le joueur arrivait bien au bon endroit après un changement de salle.

Certains problèmes sont apparus avec les collisions créées dans Tiled. Certaines formes étaient trop grandes, mal placées ou trop complexes. Nous avons donc dû ajuster plusieurs hitbox manuellement.

Un mode debug a été utilisé afin d'afficher les collisions à l'écran pendant les tests. Cela nous a permis de repérer plus facilement les zones incorrectes et de les corriger rapidement.

12.5 Correction des bugs

Pendant le développement, de nombreux bugs ont été corrigés progressivement.
Les bugs les plus fréquents concernaient :

— les erreurs de chemins vers les assets,

— les problèmes d'affichage,

— les collisions mal placées,

— les mini jeux qui ne rendaient pas correctement l'argent,

— les problèmes de synchronisation réseau,

— les erreurs lors du changement de map.

Certains bugs étaient simples à corriger, tandis que d'autres demandaient de revoir une partie plus importante du code.

Par exemple, la gestion de l'argent dans certains mini jeux a demandé plusieurs corrections afin d'éviter que le joueur gagne deux fois la même somme ou que son argent soit mal actualisé.

Le réseau a également demandé plusieurs phases de correction, car une fonction qui semblait fonctionner seule pouvait créer des problèmes une fois intégrée dans le reste du jeu.

Ces corrections ont été faites progressivement, souvent juste après les phases de tests ou avant les soutenances.

12.6 Validation finale du projet

La validation finale du projet a consisté à vérifier que toutes les parties principales du jeu fonctionnaient ensemble.

Nous avons vérifié :

— le lancement du jeu,

— le menu principal,

— les déplacements dans le casino,

— les transitions entre les maps,

— les mini jeux,

— le système de sauvegarde,

— le mode LAN,

— le site web,

— l'exécutable Windows.

Même si certains éléments restent perfectibles, le projet final est jouable et permet de montrer les principales fonctionnalités prévues.

Le jeu possède aujourd'hui une structure complète avec un casino explorable, plusieurs mini jeux, une sauvegarde, un site web, un exécutable et un mode réseau local.

13 Bilan du projet

13.1 Objectifs atteints

Plusieurs objectifs fixés au début du projet ont été atteints.

Nous avons réussi à créer un jeu en 2D jouable avec un univers cohérent et une direction artistique identifiable. Le joueur peut se déplacer dans un casino, interagir avec des éléments du décor et accéder à plusieurs mini jeux.

Les principales fonctionnalités réalisées sont :

— un moteur de jeu fonctionnel,

— des maps avec collisions et transitions,

— plusieurs mini jeux jouables,

— une interface utilisateur cohérente,

— un système de sauvegarde,

— un mode multijoueur LAN,

— un site web,

— un exécutable Windows.

Tous les objectifs n'ont pas été atteints exactement comme prévu au départ, mais le projet final reste complet et présentable.

Certaines fonctionnalités secondaires ont été abandonnées ou simplifiées, mais cela nous a permis de concentrer le travail sur les éléments les plus importants du jeu.

13.2 Compétences acquises

Ce projet nous a permis de progresser dans plusieurs domaines.

Sur le plan technique, nous avons appris à mieux utiliser Pygame pour créer un jeu complet avec des déplacements, des collisions, des interfaces et des mini jeux.

Nous avons également découvert ou approfondi :

— la gestion des maps avec Tiled,

— l'utilisation de PyTMX et PyScroll,

— la création de sauvegardes en JSON,

— la génération d'un exécutable avec PyInstaller et l'installation avec Inno Setup,

— la mise en place d'un réseau LAN simple,

— l'organisation d'un site web.

Le projet nous a aussi appris à mieux structurer notre code. Au début, certaines parties étaient difficiles à modifier car elles n'étaient pas assez séparées. Avec le temps, nous avons compris l'intérêt de mieux organiser les fichiers et de séparer les responsabilités.

Enfin, nous avons beaucoup progressé sur la correction de bugs. Le fait d'intégrer plusieurs systèmes différents dans un même jeu nous a obligés à chercher l'origine des problèmes et à tester régulièrement les nouvelles fonctionnalités.

13.3 Expérience du travail en groupe

Le travail en groupe a été une partie importante du projet.

Nous avons dû apprendre à répartir les tâches, communiquer régulièrement et adapter notre organisation en fonction de l'avancement réel.

Chaque membre avait un rôle principal, mais le projet a souvent demandé de l'entraide. Certaines parties, comme le réseau, les mini jeux ou les interfaces, ont nécessité plusieurs discussions et plusieurs corrections collectives.

L'utilisation de Trello, Discord, Instagram et Google Sheets nous a aidés à suivre l'avancement du projet et à mieux répartir les priorités.

Le projet nous a aussi montré les difficultés du travail collaboratif. Il fallait faire attention aux conflits Git, aux fichiers modifiés en même temps et aux différences de rythme entre les membres du groupe.

Malgré ces difficultés, le groupe a réussi à avancer jusqu'à la soutenance finale avec un projet jouable.

13.4 Retour sur le projet

Avec du recul, le projet a été plus ambitieux que ce que nous avions imaginé au départ.

Créer un jeu avec plusieurs mini jeux, des maps, un site web, un système de sauvegarde et un mode réseau demandait beaucoup d'organisation et de temps.

Certaines fonctionnalités ont pris beaucoup plus de temps que prévu. Le réseau, par exemple, a été particulièrement difficile à stabiliser. Le Buckshot Roulette a également demandé beaucoup de travail à cause de ses nombreux états et de sa logique de gameplay.

Nous avons aussi compris qu'il valait mieux avoir moins de fonctionnalités, mais mieux intégrées, plutôt que beaucoup d'idées incomplètes.

Même si le projet n'est pas parfait, nous sommes satisfaits de son évolution. Entre la première version et la version finale, le jeu a énormément changé, aussi bien visuellement que techniquement.

14 Conclusion

14.1 Résumé du projet

Rot it or Lose it est un jeu vidéo 2D en pixel art réalisé avec Python et Pygame.

Le projet propose au joueur d'explorer un casino rétro et de participer à plusieurs mini jeux comme la roulette, le blackjack, le Buckshot Roulette et des jeux d'arcade.

Le jeu repose sur une monnaie virtuelle, les Rot Coins, qui permet au joueur de miser et de gérer son argent au fil des parties.

Le projet comprend également :

— un système de maps,

— des collisions,

— des transitions,

— des interfaces,

— un système de sauvegarde,

— un mode LAN,

— un site web,

— un exécutable Windows.

L'objectif était de créer un jeu cohérent, jouable et reconnaissable, avec une ambiance casino rétro volontairement caricaturale.

14.2 Évolution depuis la première soutenance

Depuis la première soutenance, le projet a beaucoup évolué.

Au départ, seules les bases du jeu étaient présentes. Le moteur principal, les déplacements, les premières collisions et quelques prototypes de mini jeux étaient encore en construction.

Lors de la deuxième soutenance, plusieurs systèmes étaient déjà plus avancés : les maps étaient plus complètes, certains mini jeux étaient jouables et les interfaces commençaient à prendre une vraie identité visuelle.

Entre la deuxième et la troisième soutenance, nous avons surtout travaillé sur la finalisation et l'intégration globale du projet.

Nous avons amélioré :

— les maps finales,

— les interfaces,

— la roulette,

— le blackjack,

— le Buckshot Roulette,

— le réseau LAN,

— le site web,

— l'exécutable,

— la stabilité générale du jeu.

Le projet final est donc beaucoup plus abouti que les premières versions. Il ne s'agit plus seulement d'un ensemble de prototypes, mais d'un jeu complet avec un univers, une structure et plusieurs fonctionnalités reliées entre elles.

14.3 Perspectives futures

Même si le projet est aujourd'hui jouable, plusieurs améliorations pourraient être envisagées dans le futur.

Nous pourrions par exemple améliorer le mode réseau afin de le rendre plus stable et plus complet. Une meilleure gestion des déconnexions, des salons ou du nombre de joueurs pourrait rendre le multijoueur plus agréable.

Les mini jeux pourraient également être enrichis avec davantage d'animations, de sons et de variantes de gameplay.

Le casino pourrait aussi être agrandi avec de nouvelles salles, de nouveaux personnages et davantage d'interactions.

D'autres améliorations possibles seraient :

— ajouter plus de dialogues,

— améliorer les animations du joueur,

— ajouter davantage de musiques,

— créer de nouveaux mini jeux,

— améliorer les menus,

— rendre le site web plus complet.

Ces améliorations permettraient de rendre l'univers du jeu encore plus vivant et d'approfondir l'expérience proposée au joueur.

Malgré ces pistes d'amélioration, nous sommes satisfaits du résultat obtenu. Le projet nous a permis d'apprendre beaucoup de choses et de mieux comprendre les difficultés liées à la création d'un jeu vidéo complet en équipe.

A Annexes

A.1 Captures d'écran du jeu

FIGURE 1 – Écran d'accueil principal du jeu

FIGURE 2 – Salle principale du casino

FIGURE 3 – Salle du bar et ambiance intérieure

FIGURE 4 – Salle dédiée au Buckshot Roulette

A.2 Interfaces utilisateur

FIGURE 5 – Menu pause

FIGURE 6 – Menu principal du Blackjack

FIGURE 7 – Menu principal du Buckshot Roulette

FIGURE 8 – Interface de sélection de la Roulette

FIGURE 9 – Menu des mini-jeux d'arcade

FIGURE 10 – Système de dialogues avec les PNJ

FIGURE 11 – Dialogue et mise en scène dans le Buckshot Roulette

A.3 Gameplay des mini-jeux

FIGURE 12 – Interface de mise du Blackjack

FIGURE 13 – Gameplay du Buckshot Roulette

FIGURE 14 – Interface de paris de la Roulette

FIGURE 15 – Mini-jeu arcade Turret

FIGURE 16 – Mini-jeu Pile ou Face

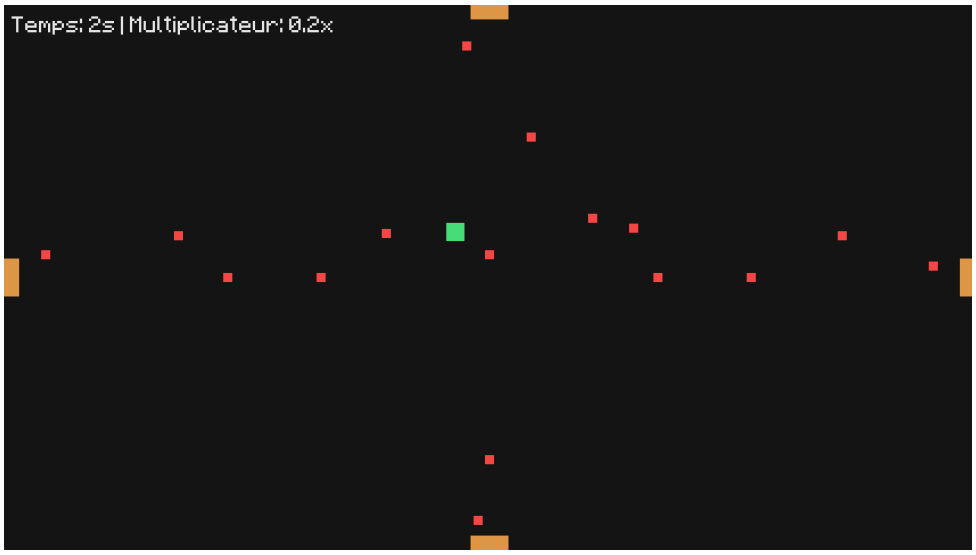

FIGURE 17 – Mini-jeu arcade type bullet hell

A.4 Exemples de tests réseau

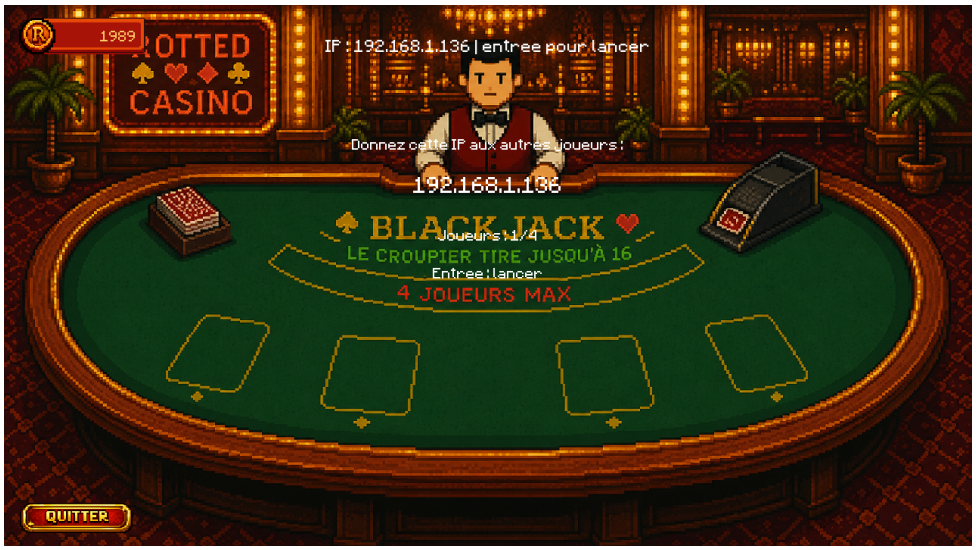

FIGURE 18 – Connexion LAN du Blackjack

FIGURE 19 – Système de connexion réseau du Buckshot Roulette

A.5 Planning du projet

Planning (21 décembre → Soutenance 1 : 13 janvier)

Tâche	Responsable (R)	Suppléant (S)	Début	Fin (rendu)	Objectif % (S1)	Livrable attendu pour atteindre l'objectif
Bases (Main, Game, Screen, Mapscroll, Player, animations, collisions)	Flavien	Oscar	21 déc.	29 déc.	90%	Boucle de jeu stable + déplacement + scroll + collisions (player & events) + anim player. Base pour brancher les mini-jeux.
Menu principal	Oscar	Flavien	21 déc.	24 déc.	100%	Écran titre + boutons (Jouer / Explications / Quitter) + navigation fonctionnelle + intégration visuelle cohérente.
Création des maps (bar + salle principale + zones)	Lucas	Oscar	21 déc.	28 déc.	60%	Map jouable du casino (au moins bar + salle principale) avec collisions/portails de base.
Tileset pour map	Oscar	Équipe	21 déc.	30 déc.	40%	Tiles essentiels (sols, murs, portes, éléments décor) suffisants pour rendre la map cohérente.
Roulette	Nathanael	Amadou	21 déc.	02 janv.	60%	Prototype jouable : mise simple + tirage + calcul gain/perte + affichage basique.
Blackjack	Nathanael	Oscar	26 déc.	06 janv.	20%	Noyau règles : distribution + hit/stand + calcul main + résultat (sans polish).
Jeu de cartes personnalisé	Lucas	Amadou	26 déc.	03 janv.	10%	Design + base technique (classes/cartes) + mini démo (ex: piocher/jouer une carte) pour prouver la faisabilité.
Arcade (mini-jeu)	Flavien	Nathanael	27 déc.	04 janv.	20%	Logique de code
Various UI/UX	Oscar	Amadou	24 déc.	08 janv.	40%	UI de base pour les mini-jeux (textes, boutons, retour lobby) + cohérence typographie/couleurs.
Network	Flavien	Nathanael	30 déc.	10 janv.	10%	Connexion/échange minimal (hello + synchro simple) : preuve de concept réseau.
Savemanager	Flavien	Lucas	24 déc.	07 janv.	50%	Sauvegarde/chargement minimal (argent, paramètres) + fichier local + tests basiques.
Soundtrack	Amadou	Lucas	21 déc.	06 janv.	20%	1 thème principal + 1 ambiance casino (boucles) + intégration lecture audio in-game.
Buckshot	Oscar	Lucas	02 janv.	13 janv.	0%	Pas attendu pour Soutenance 1 (objectif 0%). Préparer doc règles/écran maquette si temps.

FIGURE 20 – Organisation de la première soutenance

Tâche	Responsable (R)	Suppléant (S)	Fin (rendu)	Objectif % (S2)	Livrable attendu pour atteindre l'objectif	Avancement (%)
Bases (Main, Game, Screen, Mapscroll, Player, animations, collisions)	Flavien	Oscar	18 mars	100%	Architecture finale fonctionnelle, gestion player + collisions stable	100%
Menu principal	Oscar	Flavien	18 mars	100%	Menu interactif final relié aux différentes zones du jeu	100%
Buckshot	Flavien	Oscar	18 mars	45%	Gameplay jouable avec logique de tir et gestion du tour	90%
Roulette	Lucas	Oscar	18 mars	90%	Roulette jouable avec système de mise et résultat aléatoire	90%
Blackjack	Oscar	Amadou	18 mars	40%	Logique des cartes fonctionnelle avec règles principales	50%
Arcade	Flavien	Lucas	18 mars	80%	Mini-jeu arcade jouable intégré dans la map	60%
Tileset pour map	Oscar	Équipe	18 mars	100%	Tileset complet pour casino et environnement	100%
Création des maps (bar + salle principale + zones)	Lucas	Oscar	18 mars	100%	Map finale navigable avec zones interactives	100%
Soundtrack	Amadou	Lucas	18 mars	60%	Musique principale et sons de base intégrés dans le jeu	60%
Various UI/UX	Oscar	Amadou	18 mars	70%	Interfaces secondaires et améliorations visuelles	90%
Network	Flavien	Amadou	18 mars	40%	Connexion réseau stable pour partie multijoueur	0%
Savemanager	Flavien	Lucas	18 mars	80%	Sauvegarde simple des données joueur	80%
Jeu de cartes personnalisé	Lucas	Amadou	18 mars	70%	Prototype jouable avec règles principales	0%

FIGURE 21 – Organisation de la deuxième soutenance

Tâche	Responsable (R)	Suppléant (S)	Fin (rendu)	Objectif % (S3)	Livrable attendu pour atteindre l'objectif	Avancement (%)
Bases (Main, Game, Screen, Mapscroll, Player, animations, collisions)	Flavien	Oscar	21 mai	100%	Architecture finale stable et totalement intégrée avec gestion complète des déplacements, collisions, maps et interactions.	100%
Menu principal	Oscar	Flavien	21 mai	100%	Menu final entièrement fonctionnel avec navigation complète et accès aux différentes zones du jeu.	100%
Buckshot	Flavien	Oscar	21 mai	100%	Buckshot Roulette jouable avec gameplay complet, états de jeu, gestion des vies, mises, sprites et logique réseau LAN.	100%
Roulette	Lucas	Oscar	21 mai	100%	Roulette jouable avec système de mises, calcul automatique des gains, interface graphique complète et intégration dans le casino.	100%
Tileset pour map	Oscar	Équipe	21 mai	100%	Tileset complet et intégré pour l'ensemble des environnements du casino.	100%
Arcade	Flavien	Lucas	21 mai	100%	Mini-jeu d'arcade finalisé, jouable et intégré dans la map principale.	100%
Création des maps (bar + salle principale + zones)	Lucas	Oscar	21 mai	100%	Toutes les maps finales jouables avec collisions, transitions et zones interactives.	100%
Soundtrack	Amadou	Lucas	21 mai	100%	Musiques	100%
Various UI/UX	Oscar	Amadou	21 mai	100%	HUD, menu pause, interfaces secondaires et améliorations	100%
Savemanager	Flavien	Lucas	21 mai	100%	Système de sauvegarde finalisé avec chargement et gestion fiable des données joueur.	100%
Network	Flavien	Amadou	21 mai	100%	Connexion LAN fonctionnelle permettant de jouer en réseau sur les mini-jeux compatibles.	100%
Blackjack	Oscar	Lucas	21 mai	100%	Blackjack jouable avec logique des cartes, règles principales, mode multijoueur LAN et interface intégrée.	100%
Site web	Oscar	Lucas	21 mai	100%	Site vitrine finalisé avec présentation du projet, téléchargement du jeu et exécutable fonctionnel.	100%
Rapport final (50 pages)	Équipe	-	21 mai	100%	Rapport final complet avec architecture, gameplay, contributions, annexes et captures d'écran.	100%
Exécutable du jeu	Flavien	Oscar	21 mai	100%	Build exécutable stable permettant de lancer le jeu sans environnement de développement.	100%
Jeu de cartes personnalisé	Lucas	Amadou	Abandonné	0%	Fonctionnalité abandonnée suite au départ d'un membre du groupe et recentrage des priorités.	0%

FIGURE 22 – Organisation de la soutenance finale

A.6 Screenshots du site web

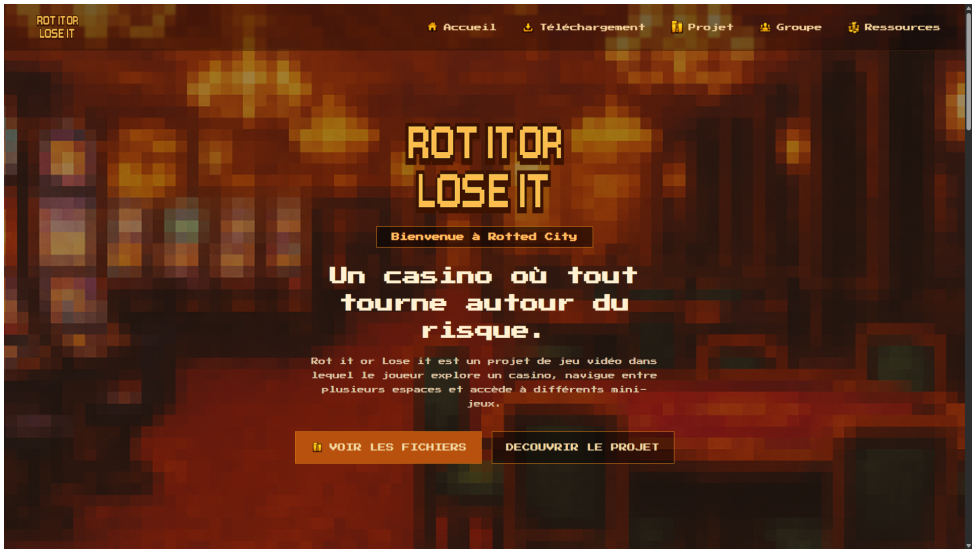

FIGURE 23 – Aperçu de la page d'accueil du site web