

ORIEN

Rapport de projet de programmation fonctionnelle



MEDJADJ Abderraouf

KERMADJ Zineddine



Université Paris-Saclay

LDD3 Magistère d'informatique

2025

Table des matières

1	Introduction	2
1.1	L'Histoire de Orien	2
2	Analyse globale	3
2.1	Interface et Navigation	3
2.1.1	Ecrans de commencement et de fin	3
2.1.2	Option et écran de pause	3
2.1.3	Panneau de contrôle du jeu	3
2.1.4	Gates	3
2.1.5	Génération de niveaux	3
2.1.6	Eternal Sun	4
2.2	Dynamique des personnages et comportements	5
2.2.1	Héro	5
2.2.2	Les ennemis	5
2.2.3	Eclipsar	5
2.3	Gestion des structures	7
2.3.1	Barriers	7
2.3.2	Barrels	7
2.3.3	Trampolines	7
2.3.4	Spikes	7
2.4	Objets ramassables	8
2.4.1	Types d'objets	8
2.4.2	Mécanismes de collecte	8
3	Organisation et Plan de Développement	9
4	Conception générale	10
4.1	Structure du Projet	10
4.2	Composants du Projet	10
4.3	Flux de données et interaction entre modules	10
4.4	Blocs Fonctionnels et Communication	11
5	Résultats	12
5.1	Panneau de contrôle	12
5.2	Héro	12
5.3	Threats	13
5.4	Collectibles	14
5.5	Other Collidables	15
6	Conclusion et perspectives	16
6.1	Perspectives d'amélioration	16
7	Sources	17

1 Introduction

1.1 L'Histoire de Orien

Tout commença un jour paisible, où les habitants du royaume d'Orien menaient une vie tranquille, sans guerre ni conflit. Cependant, cet équilibre fut brutalement rompu lorsque Eclipsar, un être malveillant, vola le Soleil et s'enfuit dans son dangereux donjon, plongeant le monde dans l'obscurité et le chaos.

Le chevalier, héros du royaume, décida de partir à la recherche du Soleil volé. Il se lança dans un périple périlleux à travers le donjon d'Eclipsar, un endroit peuplé de créatures féroces, d'ennemis redoutables, et de pièges mortels. Mais la route vers la victoire ne serait pas facile.

À travers les méandres sombres du donjon, le chevalier devait retrouver les trois fragments du Soleil : **Hope**, **Power**, et **Wisdom**. Chaque fragment était éparpillé dans une zone dangereuse du donjon, gardée par des créatures hostiles et des épreuves difficiles. Ces fragments, une fois réunis, offriraient au héros la puissance nécessaire pour affronter Eclipsar.

Le donjon n'était pas seulement peuplé de dangers, mais offrait également quelques atouts pour le chevalier. Des potions de santé et des armures pouvaient être trouvées en cours de route, permettant au héros de renforcer ses capacités et d'augmenter ses chances de survie.

Enfin, après avoir récupéré les trois fragments et affronté de nombreux ennemis, le chevalier se retrouva face à Eclipsar dans un combat épique pour récupérer le Soleil et restaurer la lumière sur le royaume d'Orien.

Ainsi, l'aventure du héros est marquée par la recherche des fragments essentiels, la lutte contre des adversaires redoutables et le défi ultime d'affronter le voleur du Soleil dans son propre repaire.

Présentation du jeu

Orien est un jeu de simulation stratégique captivant, plaçant le joueur au contrôle d'un chevalier. Votre objectif :

- Survivre en éliminant ou en esquivant les ennemis et menaces qui parcourent les niveaux.
- Explorer et récupérer des fragments essentiels pour améliorer vos capacités et progresser.
- Affronter et vaincre le boss final, Eclipsar, afin de restaurer le Soleil qu'il a dérobé et sauver le monde d'Orien.

Orien propose ainsi une expérience immersive unique, combinant gestion de ressources, stratégie en temps réel, et prise de décisions sous pression. Chaque input compte, faisant de chaque partie une aventure pleine de suspense et de rebondissements.

Cadre de réalisation

Ce jeu a été développé dans le cadre d'un projet universitaire, dans son module de Programmation Fonctionnelle Avancée. L'objectif pédagogique était double : Développer notre maîtrise du langage OCaml et tester notre adaptabilité à une bibliothèque inconnue. En parallèle, ce projet nous a permis de développer des compétences essentielles en gestion de projet et en travail collaboratif, incluant la rédaction technique, l'organisation méthodique et la planification efficace des tâches.

2 Analyse globale

Cette analyse présente les principales fonctionnalités à développer pour le projet « ORIEN » ainsi que leur niveau de difficulté et leur priorité. L’approche se décline en 4 grands pôles : Interface et Navigation, Dynamique des Personnages et Comportements, Gestion des Structures, et Objets Ramassables.

2.1 Interface et Navigation

2.1.1 Ecrans de commencement et de fin

Permet de distinguer le jeu, de lancer ce dernier ou de relancer une partie.

- **Difficulté** : *Facile*
- **Priorité** : *3/10*

2.1.2 Option et écran de pause

Permet au joueur de stopper la partie, la relancer ou lancer une nouvelle partie.

- **Difficulté** : *Moyenne*
- **Priorité** : *4/10*

2.1.3 Panneau de contrôle du jeu

Le panneau de contrôle fournit une vue d’ensemble en temps réel des éléments stratégiques du jeu sa mise à jour fluide et l’interaction intuitive avec l’utilisateur sont essentielles.

- **Difficulté** : *Facile*
- **Priorité** : *5/10*

2.1.4 Gates

Elle assure la continuité du jeu. En la franchissant, le joueur évolue d’un niveau n à un niveau $n+1$, marquant ainsi sa progression dans l’univers d’Orien. Elles ont 2 états

- **Opened** : Le joueur accède au niveau suivant en franchissant cette porte.
- **Closed** : Cette porte ne se laisse pas franchir aisément : La porte permet le passage uniquement si le héros a ramassé une clé. Elle s’ouvre alors au contact du héros, lui permettant de passer du niveau n au niveau $n+1$.
- **Difficulté** : *Moyenne*
- **Priorité** : *8/10*

2.1.5 Génération de niveaux

Bien que les portes ne soient que de simples morceaux de métal permettant de passer d’un niveau à l’autre, elles ne sont que l’élément visible d’un portail quantique bien plus sophistiqué. En coulisses, nous créons les niveaux sous forme de fichiers .txt, où chaque caractère incarne une entité, et défilons ces données en fonction des besoins du jeu.

- **Difficulté** : *Moyenne*
- **Priorité** : *10/10*

2.1.6 Eternal Sun

L'objectif du jeu est de récupérer le "Eternal Sun" volé par Eclipsar. Ce dernier fait office de porte entre le jeu et l'écran de fin.

- **Difficulté :** *Facile*
- **Priorité :** *10/10*

2.2 Dynamique des personnages et comportements

2.2.1 Héro

Le héros constitue le cœur dynamique du jeu. Il est responsable de l'exploration, du combat et de la progression dans l'univers du jeu. Le héros doit se déplacer, s'animer, se blesser, se battre, ramasser des objets, et parfois mourir. Chaque action du héros influence le déroulement de l'aventure.

- **Mouvements et interactions** : Le héros se déplace à l'aide des touches Q, Z, S, D pour se déplacer respectivement à gauche, en haut, en bas et à droite. Il peut également tirer des projectiles avec les touches fléchées, ce qui lui permet d'attaquer à distance. Pour ramasser des objets, il suffit de marcher dessus, ce qui déclenche leur récupération automatiquement.
- **Gestion des dégâts et de la vie** Le héros possède une barre de vie qui se réduit lorsqu'il subit des dégâts, notamment lors de collisions avec des ennemis ou des obstacles. Il peut également utiliser un bouclier, qui lui permet de se protéger temporairement des attaques ennemies. Si la barre de vie du héros atteint zéro, celui-ci meurt et la partie peut redémarrer à partir du dernier point de sauvegarde.
- **Capacités spéciales et objets** Le héros peut utiliser divers objets ramassés lors de ses explorations, tels que des armes, des potions de soin, ou des améliorations temporaires. Ces objets lui permettent d'améliorer ses compétences, comme la vitesse de déplacement, la puissance d'attaque ou la régénération de vie.
 - **Difficulté** : *Difficile*
 - **Priorité** : *10/10*

2.2.2 Les ennemis

Les ennemis jouent un rôle crucial, avec des comportements plus complexes : se déplaçant de manière autonome, interagissant avec le héros, et évoluant constamment d'un état à l'autre (perdant des points de vie, passant en mode attaque ou patrouille, etc.).

- **Darkies** : Des ennemis basiques un peu bêtes qui se baladent sur une même plateforme en infligeant des dégâts si rentre en collision avec le héros qui peut à son tour leur en infliger.
 - **Difficulté** : *Facile*
 - **Priorité** : *6/10*
- **Followers** : Des ennemis plus complexes. Ils ont le même comportement que les darkies s'ils ne repèrent pas le héros. Si ce dernier est repéré, il se fait attaqué jusqu'à la mort de l'un des deux ou la fuite du héros.
 - **Difficulté** : *Facile*
 - **Priorité** : *4/10*
- **Fireball Towers** : Des structures qui attaquent le héros sans relâche en lui tirant dessus avec des boules de feu. Deux options : L'éliminer ou se réfugier derrière un obstacle.
 - **Difficulté** : *Moyenne*
 - **Priorité** : *3/10*

2.2.3 Eclipsar

Le boss final traque le joueur en adaptant ses déplacements à sa position et déchaîne, de façon imprévisible, l'une de ses trois attaques redoutables toutes les secondes. Il perd de la vie au fur et à mesure des attaques du héros, jusqu'à sa mort.

- **Difficulté** : *Moyenne*
- **Priorité** : *10/10*

- **Attaque de mêlée** : Le boss charge violemment le héros, lui infligeant d'importants dégâts lors de l'impact. Cette attaque peut être déclenchée si le Eclipsar est acculé dans un coin, ou de manière aléatoire avec une chance de 30 %.
 - **Difficulté** : *Facile*
 - **Priorité** : *4/10*

- **Spawn followers** : Dans 20 % des cas, Eclipsar invoque ses sbires et fait surgir deux Followers, prêts à déclencher leur fureur sur le héros.
 - **Difficulté** : *Facile*
 - **Priorité** : *5/10*

- **Fireball** : Eclipsar, 20 % du temps, fait pleuvoir sa furie en lançant des boules de feu : l'une directement sur le héros, l'autre 20° plus haut, et la troisième 20° plus bas, créant un piège infernal.
 - **Difficulté** : *Moyenne*
 - **Priorité** : *6/10*

2.3 Gestion des structures

2.3.1 Barriers

Littéralement la brique la plus importante du jeu. La véritable fondation du jeu repose littéralement sur les murs, le sol, et même le plafond, des éléments essentiels qui forment l'ossature du monde.

- **Difficulté** : *Facile*
- **Priorité** : *10/10*

2.3.2 Barrels

Ces barrières mobiles, guidées par la gravité, deviennent des atouts stratégiques : elles peuvent emprisonner des ennemis, offrir une protection contre des attaques puissantes, ou même étendre des plateformes, offrant ainsi une flexibilité tactique sans pareille.

- **Difficulté** : *Facile*
- **Priorité** : *7/10*

2.3.3 Trampolines

Les trampolines, élément dynamique du jeu, permettent de sauter à des hauteurs impressionnantes. Ces objets offrent une flexibilité tactique majeure : ils peuvent être utilisés pour atteindre des zones inaccessibles, éviter des attaques ennemies, ou encore se propulser pour attaquer des ennemis depuis les airs. Grâce à leur interaction avec la gravité, ils ajoutent une dimension supplémentaire à l'exploration et au combat.

- **Difficulté** : *Facile*
- **Priorité** : *2/10*

2.3.4 Spikes

Temp

- **Difficulté** : *Facile*
- **Priorité** : *9/10*

2.4 Objets ramassables

Les objets ramassables jouent un rôle clé dans l'évolution du joueur et la progression dans le jeu. Ces objets peuvent fournir des avantages tels que des ressources, des armes, des clés, ou des améliorations temporaires. En les récupérant, le héros pourra débloquent de nouvelles capacités, accéder à des zones inaccessibles ou survivre plus longtemps face aux ennemis.

2.4.1 Types d'objets

Les objets ramassables sont variés. On distingue :

- **Clés** : Permettent d'ouvrir des portes verrouillées et de passer d'un niveau à l'autre.
- **Potions** : Objets nécessaire à la gestion des ressources du joueur, Une potions de soin qui rétablit un point de vie.
- **Shields** : Le héro peut aussi ramasser un bouclier s'il a perdu le sien.
- **Fragments** : Le jeu comporte trois fragments puissants que le joueur doit impérativement ramasser pour avoir une chance de finir l'aventure.
 - **Hope Fragment** : Augmente la santé totale du joueur à 2 points.
 - **Power Fragment** : Augmente la force du joueur à 2 points d'attaque.
 - **Wisdom Fragment** : Augmente la santé totale du joueur à 3 points et la force du joueur à 3 points d'attaque.

2.4.2 Mécanismes de collecte

Les objets sont disséminés dans les niveaux et peuvent être récupérés en entrant en contact avec eux. Certaines mécaniques, telles que des zones spécifiques ou des ennemis particuliers, peuvent rendre l'accès à ces objets plus difficile, incitant le joueur à explorer et à risquer davantage.

- **Difficulté** : *Facile*
- **Priorité** : *8/10*

L'analyse met en évidence que la priorité doit être donnée aux mécanismes centraux du jeu : l'interface de contrôle, la dynamique du héro et des ennemis, ainsi que la gestion des structures et des ressources. Ces éléments, jugés faciles à mettre en œuvre ou moyennement complexes, constituent la base d'une expérience de jeu fluide et stratégique. À ce socle s'ajoutent des fonctionnalités complémentaires – telles que l'histoire, l'ambiance sonore – qui, bien que non critiques pour la jouabilité, enrichiront l'immersion et la profondeur narrative du jeu.

3 Organisation et Plan de Développement

Durant la réalisation du projet, nous avons employé plusieurs outils afin d'optimiser notre organisation et assurer un suivi efficace des différentes tâches. Parmi ces outils, nous avons utilisé principalement :

- **Un tableau Trello** permettant une gestion visuelle des tâches à réaliser, en cours et achevées.
- **Un serveur Discord** facilitant la communication instantanée et la collaboration entre les membres de l'équipe.
- **Un diagramme de GANTT** indispensable à la planification temporelle du projet.

Le diagramme de GANTT nous a particulièrement aidés à répartir clairement les responsabilités et à suivre l'état d'avancement du projet tout au long de son développement. Cet outil s'est avéré crucial pour respecter les délais impartis.

4 Conception générale

Le projet *Orien* est une simulation en temps réel basée sur l'architecture **ECS** (Entity-Component-System). Il est conçu autour de plusieurs blocs fonctionnels qui communiquent pour gérer la logique de jeu, l'affichage et les interactions.

4.1 Structure du Projet

Le projet *Orien* repose sur l'architecture ECS (Entity-Component-System), mais avec une organisation modulaire où les entités, leurs composants, leurs logiques, ainsi que les systèmes et les états du jeu sont centralisés dans différents dossiers. Cette architecture permet de gérer la logique du jeu de manière flexible tout en conservant un code propre et évolutif.

4.2 Composants du Projet

- **Entities** : Contrairement à une organisation ECS classique avec un dossier spécifique pour les entités, toutes les entités sont déclarées dans le fichier `src/Component/component_defs`. Chaque entité est représentée par un identifiant unique et est associée à des composants qui définissent ses propriétés. Ces entités ne contiennent aucune logique propre mais sont gérées par des systèmes en fonction de leurs composants.
- **Component** : Le dossier `src/Component` contient les définitions de tous les composants ainsi que la logique liée à chaque entité. Le fichier `component_defs` est utilisé pour définir les données des entités, telles que la position, la vitesse, la santé, et l'apparence. Chaque composant contient uniquement des données, sans comportement. Les autres fichiers dans le dossier **Component** contiennent des logiques spécifiques qui définissent le comportement de certaines entités.
- **System** : Le dossier `src.System` contient la logique métier du jeu, sous la forme de systèmes ECS. Chaque système parcourt les entités qui possèdent certains composants et applique des comportements en fonction des données présentes dans ces composants. Par exemple, un système de déplacement met à jour la position des entités en fonction de leur vitesse, ou un système de collision vérifie les intersections entre les entités.
- **Core** : Le dossier `src/Core` gère les constantes du jeu, l'état global du jeu (`game_state`), ainsi que les fonctions utilitaires telles que celles manipulant les vecteurs. Ce module sert de base pour orchestrer les autres composants et systèmes du jeu.
- **Game** : Le fichier `./game.ml` est le point d'entrée principal du jeu. Il centralise la logique générale du jeu, y compris le démarrage, l'initialisation des entités et des systèmes, ainsi que la gestion du cycle principal du jeu (mise à jour et rendu).
- **Ressources** : Le dossier `./ressources` contient tous les fichiers nécessaires au bon fonctionnement du jeu : les niveaux au format `.txt` et les textures utilisées pour le rendu graphique. Les fichiers de niveau définissent la structure des environnements du jeu, tandis que les textures sont utilisées pour l'affichage visuel des éléments du jeu, comme les personnages, les objets et les décors.

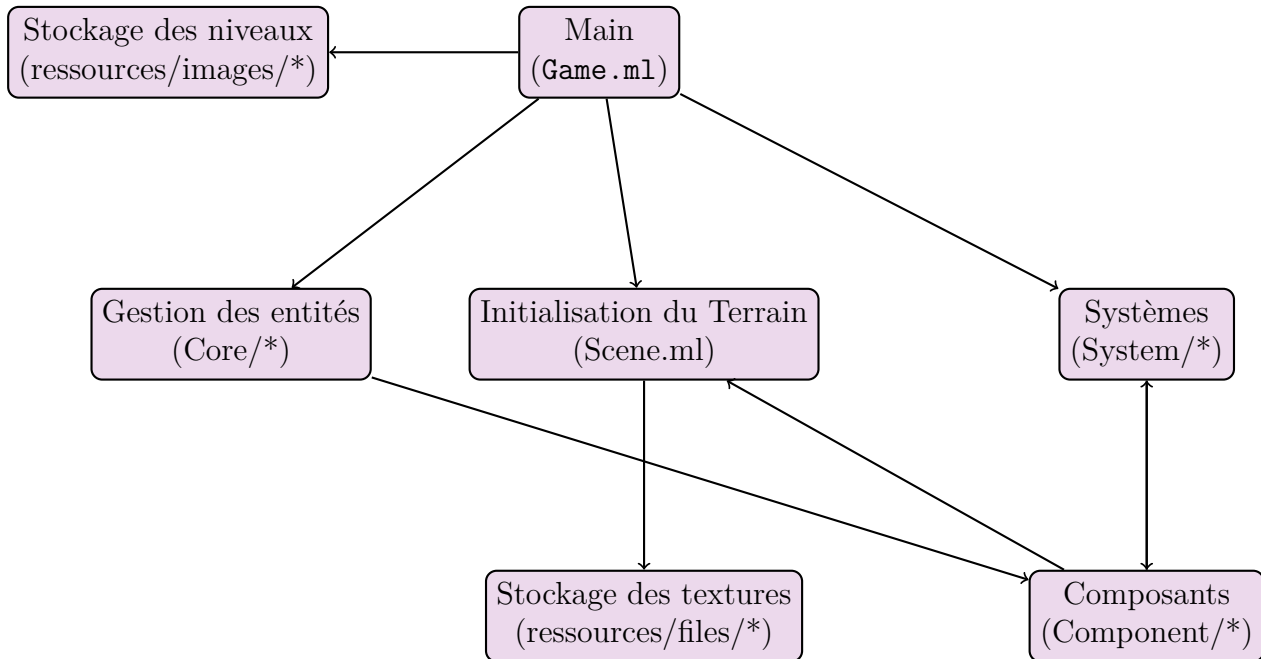
4.3 Flux de données et interaction entre modules

Le flux de données dans le projet suit une approche ECS, où les entités sont définies par des identifiants uniques et associées à des composants décrivant leurs caractéristiques. Les entités elles-mêmes n'ont pas de logique mais sont gérées par les systèmes, qui utilisent les composants pour appliquer des comportements (comme le déplacement, la gestion des collisions,

etc.). Les fichiers de composants (`Component/component_defs` et autres fichiers dans le dossier `Component`) servent à définir les entités et leur état.

Les systèmes dans le dossier `system` parcourent les entités avec des composants spécifiques et appliquent les logiques correspondantes. Le module `core` gère les fonctions utilitaires et les états du jeu, tandis que `game.ml` orchestre l'ensemble du projet en initialisant et en gérant le cycle du jeu.

4.4 Blocs Fonctionnels et Communication



5 Résultats

La version (presque) finale du jeu présente une interface graphique dynamique et immersive, intégrant plusieurs modules essentiels qui se combinent pour offrir une expérience de simulation riche et interactive.

5.1 Panneau de contrôle

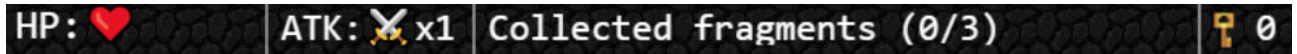


FIGURE 1 – Panneau de contrôle réduit

5.2 Héro

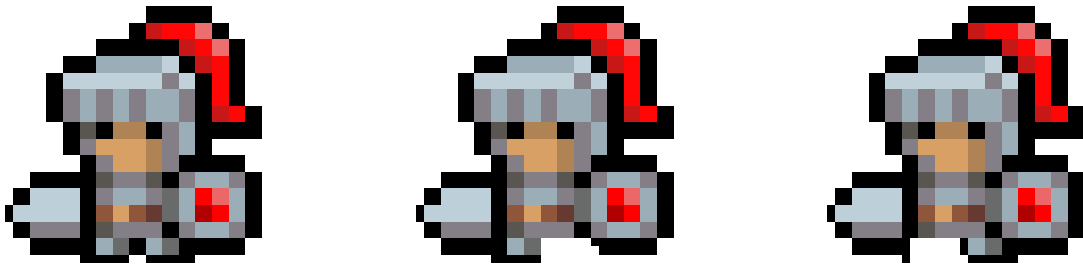


FIGURE 2 – Les trois positions du héros orienté vers la gauche



FIGURE 3 – Les trois positions du héros orienté vers la gauche sans bouclier

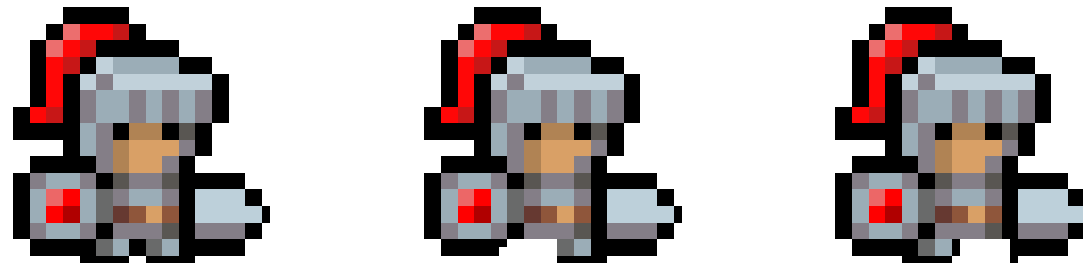


FIGURE 4 – Les trois positions du héros orienté vers la droite



FIGURE 5 – Les trois positions du héros orienté vers la droite sans bouclier

5.3 Threats

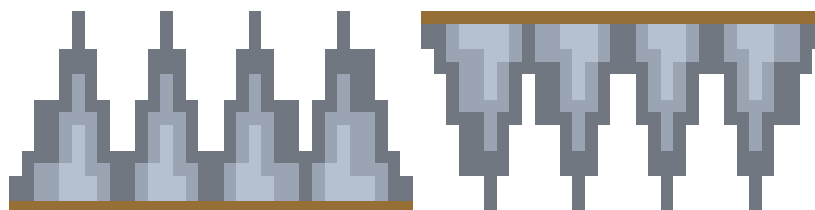


FIGURE 6 – Les piques

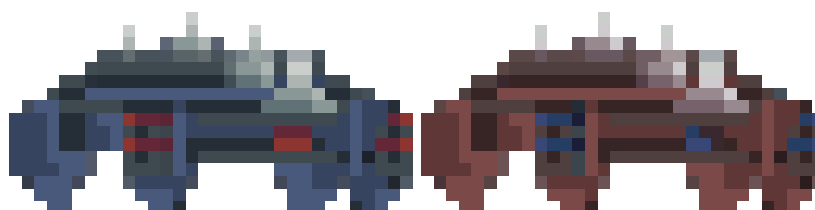


FIGURE 7 – Darkies and Followers



FIGURE 8 – Les tours et les boules de feu



FIGURE 9 – Eclipsar

5.4 Collectibles

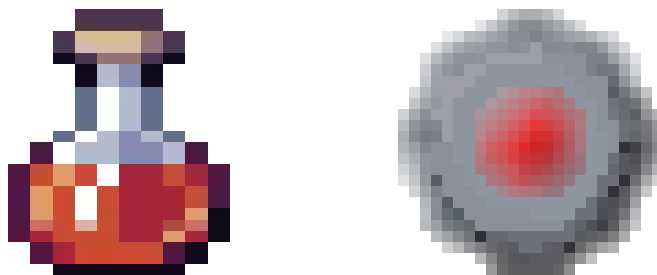


FIGURE 10 – Potion de santé et Bouclier

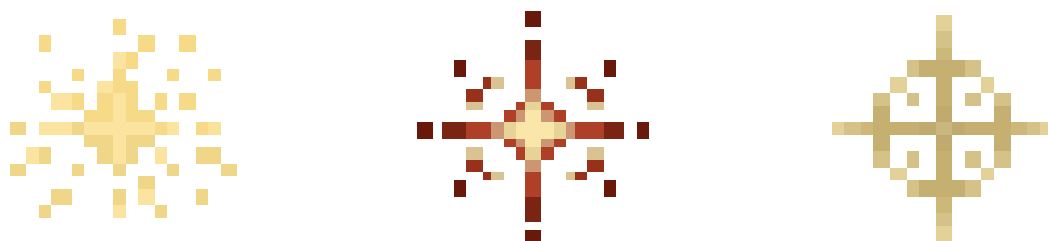


FIGURE 11 – Fragments

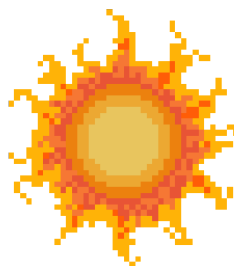


FIGURE 12 – Le soleil eternal

5.5 Other Collidables

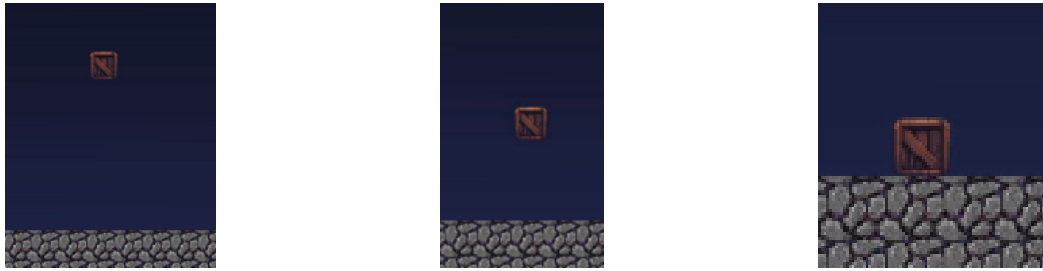


FIGURE 13 – La physique des barrels

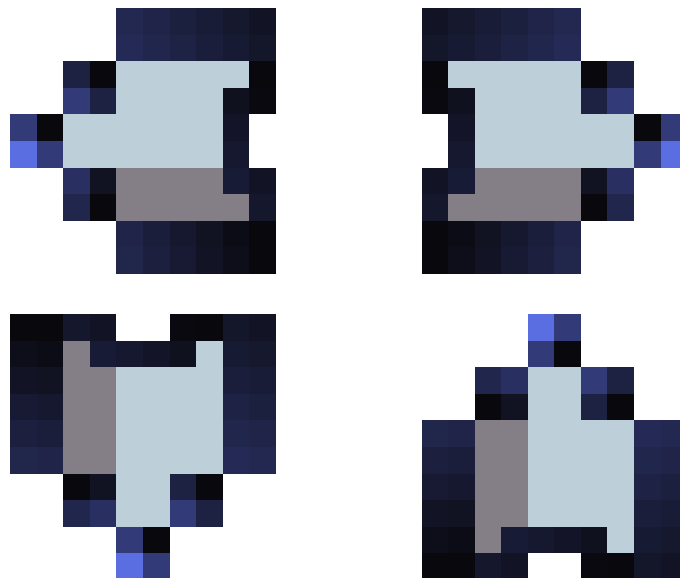


FIGURE 14 – Les quatre orientations du projectile

La mise en œuvre de ces composants repose sur une architecture logicielle robuste, garantissant une synchronisation efficace entre l'interface utilisateur et les processus en arrière-plan.

6 Conclusion et perspectives

Dans le cadre de ce projet, nous avons développé un jeu platformer en temps réel nommé *Orien*, basé sur une architecture ECS (Entity-Component-System). Cette approche nous a permis de modéliser de manière flexible et performante les entités du jeu telles que le joueur, les ennemis, les projectiles ou encore les éléments du décor. Le but principal du jeu est de vaincre un boss final, en progressant à travers des niveaux dynamiques et hostiles.

L'architecture repose sur trois modules principaux :

Entity : représente les objets du jeu sous forme d'identifiants uniques.

Component : encapsule les données des entités (héro, boss, ennemis, blocs..etc).

System : contient la logique métier. Chaque système agit sur les entités possédant un certain ensemble de composants (système de mouvement, collision, IA, rendu...).

Nous avons également veillé à découpler soigneusement le traitement des entrées utilisateur, la logique d'IA et le rendu graphique, assurant ainsi une exécution fluide et réactive du jeu, même durant les phases d'action intense.

Les principales difficultés rencontrées ont concerné la coordination en équipe, la maîtrise du temps imparti, ainsi que la gestion de la complexité liée à l'architecture modulaire et découplée propre à l'ECS. Pour surmonter ces défis, nous avons mis en place une organisation rigoureuse :

- Utilisation d'outils de gestion de version (Git),
- Réunions de suivi régulières pour synchroniser les avancées,
- Structuration claire des composants, systèmes et états de jeu.

Ce projet nous a permis de développer une meilleure compréhension des architectures interactives modernes, de l'optimisation en temps réel, ainsi que des enjeux liés à la collaboration en développement logiciel. Il a renforcé nos compétences en conception modulaire, en gestion des entités dynamiques et en organisation du code à grande échelle.

6.1 Perspectives d'amélioration

- **Amélioration graphique** : Intégration d'un moteur de rendu plus performant, supportant des effets avancés (shaders, particules, éclairages dynamiques) et des animations fluides pour offrir une immersion visuelle accrue.
- **Extension des mécaniques de gameplay** : Développement de nouvelles fonctionnalités comme des capacités spéciales dynamiques, un système d'inventaire complexe, ainsi qu'une IA de boss adaptative avec des patterns d'attaque évolutifs.
- **Optimisation et scalabilité** : Refonte partielle de la boucle principale du jeu et des systèmes ECS pour assurer des performances optimales sur des environnements plus vastes, tout en conservant une architecture modulaire et extensible.

Ces axes d'amélioration visent à faire évoluer *Orien* vers un projet de plus grande envergure, capable de proposer une expérience de jeu toujours plus riche, immersive et technologiquement avancée.

7 Sources