KILL DEM'ON

ican
institut de création et
animation numériques

Sommaire

Équipe

Denis Hottin
Programmation
Game Design
DA Visuel/VFX
Gestion de projet

Arthur Lunais
Game Design
Documentation
Sound Design
VFX

Loïc Ding
Programmation
Prog Tool

Yoann Barbosa
Concept Art
Game Art 2D/3D
UI
Prog Tool

Raphael Alun
Game Design
Documentation
VFX

Léo Chevance
Game Design
Level Design

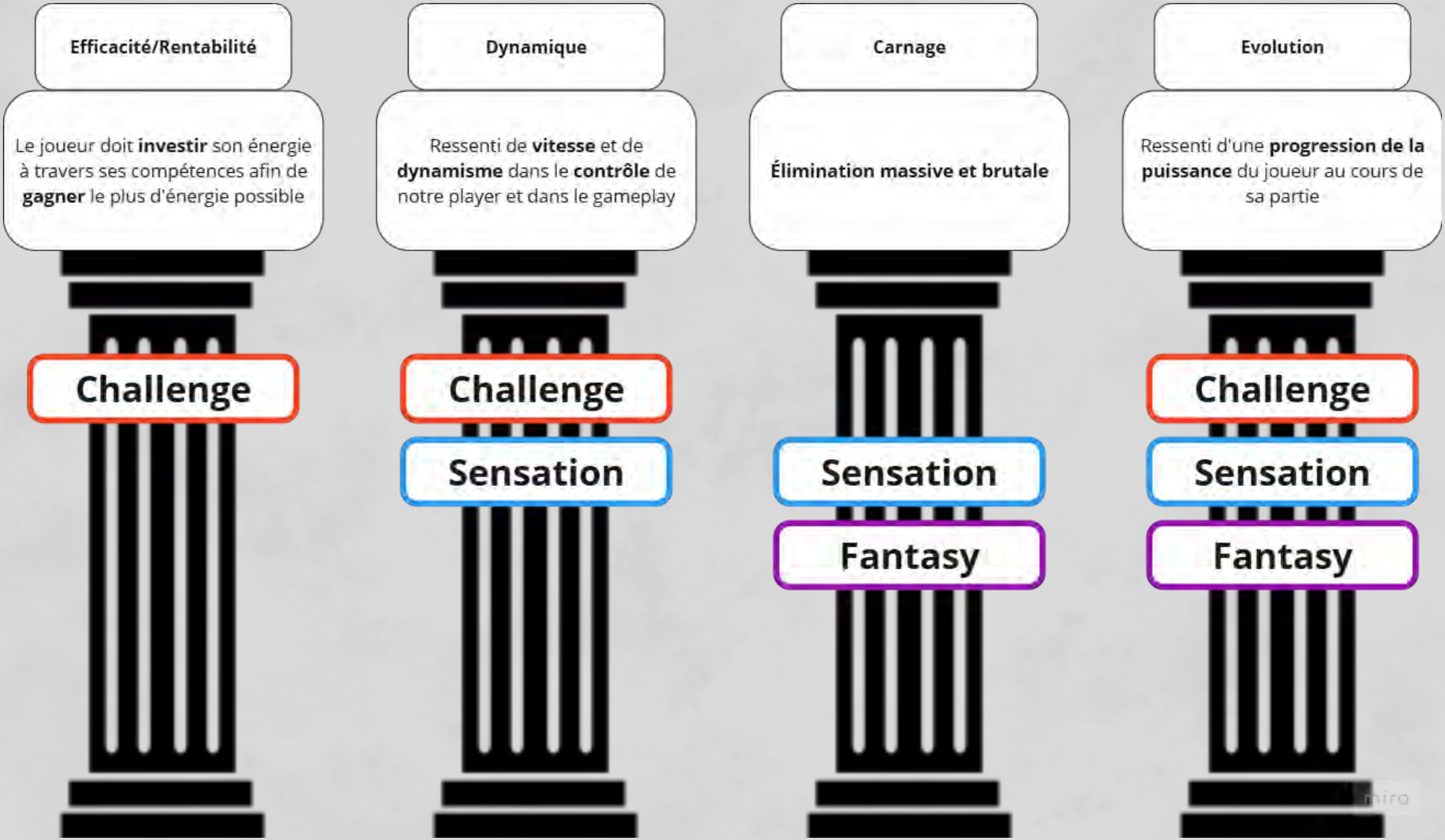
Introduction

Contexte du Projet :
Kill Dem'On est le projet de fin d'études de notre Bachelor en Game Design à l'ICAN. Il représente l'aboutissement de trois années de formation, et à ce titre, il occupe une place toute particulière dans notre parcours. Ce dernier projet nous a offert une opportunité unique de mettre en pratique l'ensemble des compétences acquises, mais aussi de montrer ce dont nous sommes réellement capables après trois ans d'apprentissage, d'expérimentations, et de productions diverses.

Pour la première fois, nous avons travaillé sur un même projet pendant une durée aussi longue : une année entière. Ce format étendu constituait un véritable défi en termes d'organisation, de motivation et de gestion d'équipe. Il nous a fallu apprendre à planifier sur le long terme, à structurer nos phases de travail et à maintenir un rythme constant, sans perdre en ambition ni en cohérence.

Ce projet n'est pas seulement l'aboutissement d'un cursus : c'est aussi un terrain d'expérimentation créative, un espace pour repousser nos limites et affirmer notre identité de jeunes game designers.

Outils de travail:
Pour structurer efficacement notre production sur la durée, nous avons utilisé une combinaison d'outils professionnels. Unity a été notre moteur principal, GitHub pour la gestion de version, FMOD pour le son. L'organisation et la documentation ont été assurées via Trello, Miro, Google Docs, Slides, Drive et InDesign. Pour l'UI, nous avons travaillé avec Figma, et pour le balancing ainsi que la QA, nous avons utilisé Google Sheets. Ces outils nous ont permis de travailler de manière claire, collaborative et professionnelle tout au long de l'année.


Efficacité/Rentabilité
Le joueur doit **investir** son énergie à travers ses compétences afin de **gagner** le plus d'énergie possible
Challenge
Dynamique
Ressenti de **vitesse** et de **dynamisme** dans le **contrôle** de notre player et dans le gameplay
Challenge
Sensation
Carnage
Élimination massive et brutale
Sensation
Fantasy
Evolution
Ressenti d'une **progression de la puissance** du joueur au cours de sa partie
Challenge
Sensation
Fantasy

L’éfficacité et la rentabilité :

Chaque action doit être un investissement. Le joueur gère une ressource précieuse (l’énergie), qu’il dépense pour se déplacer, attaquer ou survivre. À travers un système de compétences évolutives et de gestion stratégique, il est incité à utiliser ses capacités de manière optimale afin de maximiser ses gains d’énergie. L’objectif est clair : plus il est efficace dans ses éliminations, plus il devient rentable et puissant.

La dynamique :

Kill Dem’ On se veut nerveux, rapide et fluide. Le contrôle du personnage, les mécaniques de déplacement et les feedbacks visuels renforcent une sensation constante de mouvement. Cette dynamique est essentielle à l’immersion dans un fast-FPS : le joueur doit ressentir la vitesse, la tension et l’adrénaline à chaque instant.

Carnage :

La brutalité est au cœur de l’expérience. Les affrontements sont intenses, violents et sans concession. L’élimination des ennemis se fait de manière rapide, percutante, et souvent spectaculaire. Ce pilier vise à transmettre au joueur une sensation de puissance destructrice, presque jouissive, au sein d’un univers chaotique et oppressant.

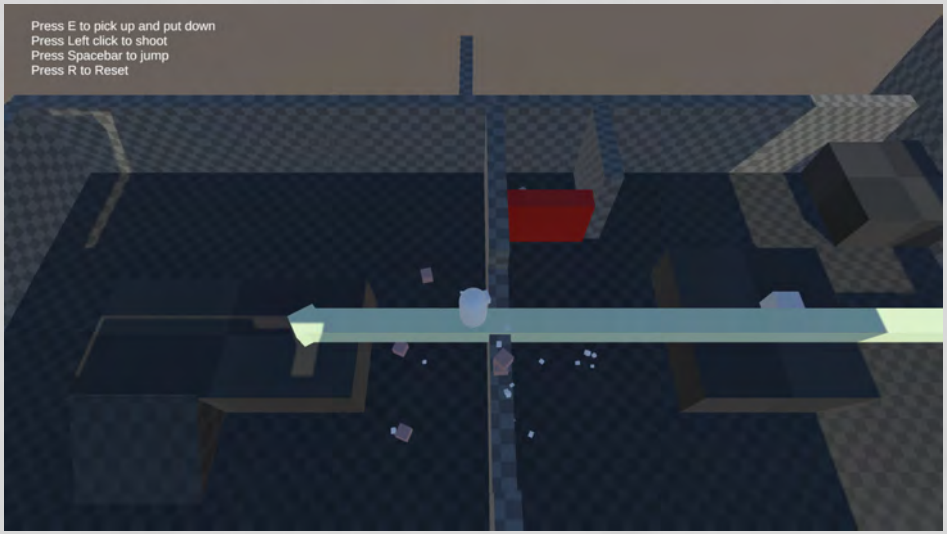
Évolution :

Le joueur commence vulnérable, mais peut progressivement devenir une force absolue. Grâce à un système de progression par paliers, chaque action réussie lui permet de monter en puissance, de débloquer de nouvelles compétences, d’améliorer ses statistiques et d’élargir ses possibilités. Ce pilier est essentiel pour renforcer l’engagement du joueur sur la durée, en lui offrant une sensation tangible de montée en puissance à chaque session.

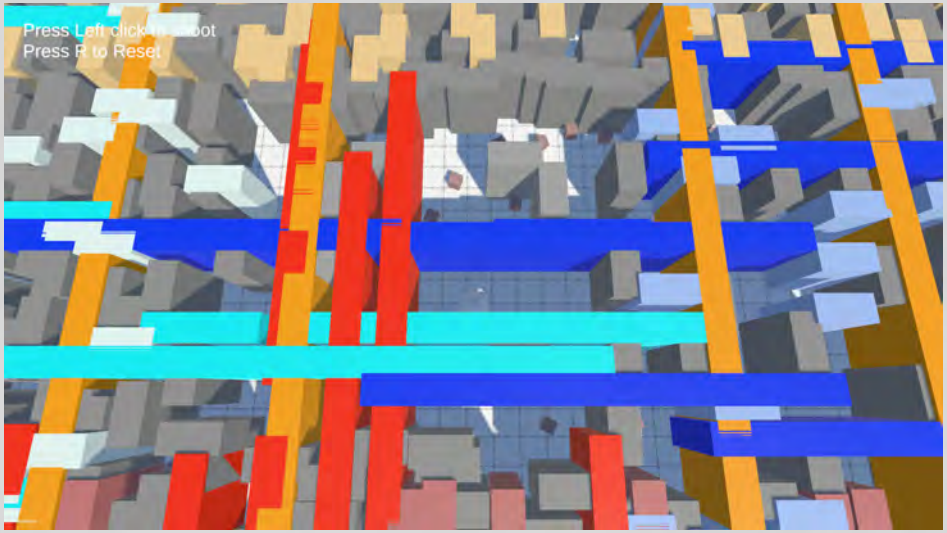
Projet Next Round :

Au départ, nous nous sommes appuyés sur le thème imposé du projet pour imaginer un concept de gameplay à la fois simple en apparence, mais ambitieux dans sa mise en œuvre. L'idée initiale reposait sur un système d'actions à répercussions différées, dans lequel chaque action du joueur avait des conséquences dans un "temps suivant". Le joueur évoluait dans un système en tour par tour où il maîtrisait le rythme du temps, avec des interactions persistantes.

Par exemple, casser un mur à un tour pouvait faire apparaître une munition à cet emplacement au tour suivant, ouvrant ainsi de nouvelles possibilités pour continuer à explorer ou interagir avec l'environnement.



Prototype de puzzle avec notre projet next round



Prototype de labyrinthe avec notre projet next round

Le concept nous semblait original, prometteur et stimulant à prototyper. Cependant, nous nous sommes rapidement heurtés à des difficultés de clarté et de cohérence dans la mise en œuvre. Malgré nos nombreuses itérations, nous peinions à garder une logique interne claire et à maîtriser pleinement notre propre "jouet". À force de variations et d'expérimentations, le système s'est complexifié au point que nous avons eu du mal à en garder le contrôle, et il devenait difficile pour le joueur de comprendre les règles implicites du jeu.

Face à ce constat, et après de nombreuses phases de test ainsi que des discussions approfondies au sein de l'équipe, nous avons pris la décision de faire un reboot complet du projet. Cette décision, loin d'être un échec, a été un moment clé dans la maturation de notre démarche de conception.
Nous vous présentons ci-dessous quelques images de "Next Round", notre première version, ainsi que certaines de ses principales itérations.

Génèse du projet :

Suite à ce reboot, nous avons entamé une nouvelle phase de brainstorming en équipe. Très rapidement, un constat s'est imposé à nous : nous voulions créer un jeu qui nous ressemble davantage, un projet plus en accord avec nos sensibilités, notre humour et notre manière de jouer.

(Je reviendrai plus loin dans le document sur la composition de notre équipe, mais disons simplement que nous sommes plutôt du genre «bourrins» : on aime l'action intense, le fun immédiat, et les jeux nerveux qu'on pourrait qualifier de "sauce tomate". C'est donc tout naturellement que l'idée d'un fast FPS s'est imposée.)
Au départ, nous avons envisagé un système dans lequel des zones influenceraient le personnage en lui attribuant des particularités et des compétences spécifiques selon l'endroit où il se trouvait. Mais très vite, cette idée s'est révélée trop complexe à appréhender pour le joueur, et nuisait à la lisibilité du gameplay.

Nous avons alors recentré notre concept autour d'un système beaucoup plus clair et lisible : un gameplay fondé sur la gestion d'une énergie centrale. Toutes les actions du joueur – courir, tirer, sauter – consomment cette ressource unique, directement liée à sa puissance. Ce choix nous a permis de conserver une profondeur stratégique tout en simplifiant la compréhension des mécaniques. Cette base est devenue le cœur de Kill Dem'On.

Version de Kill Dem'On à la première soutenance (31/01/2025)

Version de Kill Dem'On deux semaines avant la seconde soutenance

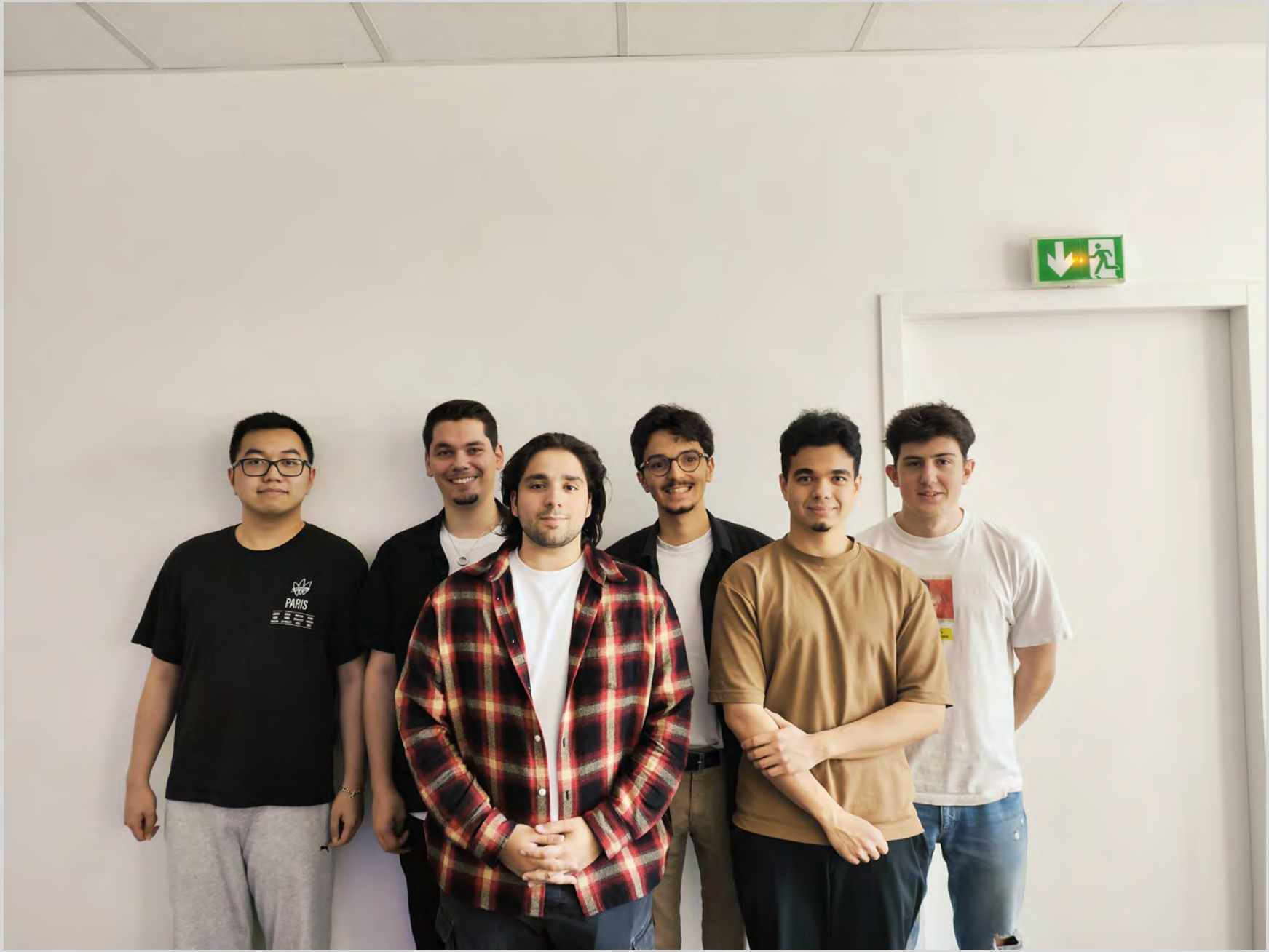
L’Équipe :

Pour développer Kill Dem’On, nous étions une équipe de six personnes. Le reboot du projet a été un véritable déclencheur : il nous a permis de mieux cerner nos forces en tant qu’équipe, et surtout de trouver une direction qui nous ressemblait vraiment.

Dès les premières discussions, une évidence s’est imposée : nous sommes tous, à notre manière, des profils assez “bourrins” — dans le bon sens du terme. On aime l’action directe, les sensations fortes, les jeux nerveux, généreux, où ça explose dans tous les sens. Ce tempérament commun a fini par se refléter naturellement dans notre gameplay : nous avons pris le parti du fun immédiat et de la destruction, plutôt que celui d’un gameplay basé sur l’analyse ou la stratégie posée, comme c’était le cas dans notre premier prototype. Ce choix a donné une véritable identité à notre jeu, que ce soit dans le rythme, les mécaniques ou même le ton général du projet.

Un autre point fondamental de notre fonctionnement d’équipe réside dans notre communication directe et honnête. Nous avons toujours su nous dire les choses franchement, sans tourner autour du pot, ce qui a permis de faire avancer les idées rapidement. Cela ne veut pas dire qu’il n’y a jamais eu de désaccords ou de frictions — évidemment, il y en a eu — mais nous avons toujours su trouver un terrain d’entente et régler les conflits sans que cela nuise au projet ou à l’ambiance.

Le fait que nous nous connaissions tous avant le projet a également beaucoup joué. Nous étions déjà amis en dehors du cadre scolaire, ce qui a rendu la collaboration plus fluide, plus naturelle, et surtout très agréable. Cette ambiance conviviale et pleine d’humour a été un vrai moteur tout au long de l’année. Elle nous a permis de travailler efficacement sans jamais perdre le plaisir de créer ensemble, ni la capacité à nous remettre en question quand c’était nécessaire.


PARIS

Game Design

Intentions Globale :

Avec Kill Dem'On, notre ambition est de proposer une expérience de fast-FPS à la fois accessible et exigeante, où le plaisir de jeu repose sur la fluidité, la maîtrise et l'intensité des actions. Nous avons conçu chaque mécanique pour qu'elle soit immédiatement compréhensible mais profondément exploitable, en misant sur des sensations fortes, une progression organique et une liberté d'approche.

Une gestion d'énergie constante et des choix tactiques rentables

Au cœur de Kill Dem'On, chaque action du joueur est conditionnée par une ressource unique : l'énergie. Tirer, sauter, sprinter ou utiliser une capacité consomme cette ressource vitale. Le joueur est donc en permanence confronté à des choix : dépenser maintenant pour un avantage immédiat, ou conserver son énergie pour survivre plus loin ? Cette mécanique impose une tension continue et pousse le joueur à réfléchir en termes de rentabilité d'action, à chaque seconde. L'efficacité devient une compétence, et la survie une affaire d'anticipation, de rythme et de gestion éclairée.

Une jouissance de la destruction et du mouvement

Nous avons conçu Kill Dem'On comme un exutoire d'action pure. Chaque tir, chaque explosion, chaque mouvement a été pensé pour procurer un plaisir immédiat, intense et gratifiant. Le gameplay repose sur une sensation de puissance permanente, renforcée par des feedbacks visuels et sonores généreux. Le jeu n'encourage pas la prudence, mais l'audace. Il célèbre le chaos contrôlé, l'élan destructeur et la prise de risque. C'est un jeu qui invite à tout casser avec style, et qui récompense le joueur non pas pour sa retenue, mais pour sa capacité à embrasser pleinement la violence du système.

Une montée en puissance claire et ressentie par le joueur

La progression dans Kill Dem'On ne repose pas sur des upgrades automatiques ou des statistiques passives, mais sur une compréhension profonde du gameplay. Le joueur devient plus fort non pas parce que le jeu lui donne plus de pouvoir, mais parce qu'il apprend à mieux jouer. Cette montée en puissance se traduit par des paliers de sensations : au début, le joueur découvre et survit. Ensuite, il anticipe, enchaîne, domine. Cette structure permet à chacun de ressentir très clairement son propre progrès, avec des moments de bascule où l'on sent que l'on a passé un cap. C'est cette évolution organique, directement liée à la maîtrise, qui rend chaque session plus intense que la précédente.

Fiche d’identité :

Pitch :

Kill Dem’On est un fast-FPS brutal et frénétique, où chaque action du joueur consomme une ressource unique : de l’énergie. Pour continuer à se mouvoir, attaquer et survivre, le joueur doit enchaîner les éliminations. La montée en puissance ne dépend pas d’objets ou d’upgrades passifs, mais de la maîtrise progressive du gameplay. Le jeu propose une sensation de domination croissante, à travers des combats nerveux, viscéraux et tactiques, dans un monde infernal et chaotique.

Cible :

Kill Dem’On s’adresse à tous les types de joueurs, en offrant une prise en main facile et des mécaniques lisibles.
Mais ce sont surtout les fans de FPS rapides et techniques qui en tireront le meilleur : leur maîtrise des réflexes, de la visée et du flow leur permettra d’exploiter pleinement le système, de grimper dans les scores, et de maintenir les paliers les plus élevés.

Le jeu reste accessible, mais il récompense fortement la persévérance, l’efficacité et l’agressivité stratégique.

Type de jeu :

- Genre : Fast-FPS solo à composante Beat them all
- Mécaniques clés : gestion de ressource (énergie), montée en puissance par paliers, ennemis évolutifs, score et leaderboard
- Style de jeu : rapide, agressif, sans interruption – le joueur est un prédateur, pas une proie

Univers :

Le jeu se déroule dans un enfer organique, crasseux et oppressant, où la chair et le chaos sont omniprésents. Cet univers sans pitié ne cherche pas à terrifier, mais à donner au joueur un terrain de carnage. L’ambiance visuelle, entre esthétique déstructurée (dithering, posterize, pixellisation) et tons rougeoyants, renforce l’impression d’être une entité destructrice surnaturelle, plus proche du démon que de l’humain.

Plateforme :

Kill Dem’On est développé exclusivement pour PC, avec une jouabilité pensée pour le combo clavier-souris.
En tant que fast-FPS, le jeu repose sur une visée précise, des réflexes instantanés et un contrôle nerveux — des sensations que seule cette configuration permet d’atteindre.

Déroulement de jeu

Palier 1 : L'entrée dans l'arène

Le joueur est projeté dans un monde brutal, sans pitié.
Il débute avec un kit minimal :
- Déplacement, saut, tir (hitScan), et un coup de poing gratuit.

Il comprend vite que chaque action coûte de l'énergie, sauf le coup de poing.
Cette énergie ne se régénère pas : elle s'obtient uniquement en tuant des ennemis ou via des FeedSeed.
Tuer -> recharger -> rejouer

Ennemis présents :
Cuby lvl 1 : simples, agressifs, cherchent le contact, parfaits pour apprendre à viser et bouger.
Le joueur commence à ressentir la boucle du jeu :
«Je tire -> je dépense. Je tue -> je récupère. Je gaspille -> je meurs.»

Palier 2 : La montée en puissance commence

Le joueur débloque le Sprint, qui lui offre :
- Des déplacements plus rapides,
- Des options offensives et défensives plus dynamiques.

Les compétences s'intensifient, mais la tension aussi :
- Cuby lvl 2 : plus rapides, plus résistants.
- Dashoot lvl 1 : tire à distance et se téléporte.

Le joueur doit désormais :
- Gérer son énergie plus finement,
- Adapter sa mobilité à des attaques à distance,
- Savoir quand fuir, attaquer ou attendre.

Palier 3 : Mobilité aérienne et impact

Deux nouvelles compétences majeures s'ajoutent :
- Multi-saut (2 sauts) -> mobilité verticale avancée.
- Pilonnage -> attaque de zone activable en vol.

Les actions déjà acquises deviennent plus efficaces :
- Le sprint est plus rapide, le tir plus puissant.

Ennemis présents :
- Cuby lvl 3, Dashoot lvl 1
- Banshee lvl 1 : vole et bombarde à distance, impossible à atteindre sans mobilité verticale.

À ce stade, chaque mouvement doit être calculé.
Le joueur alterne entre sol et air, enchaîne, esquive, et maximise ses enchaînements avec style et efficacité.

Palier 4 : Tout est permis, rien n'est gratuit

Le joueur n'apprend plus de nouveaux outils, mais :
Chaque compétence atteint son niveau maximal.
Il a 3 sauts, un sprint ++, un pilonnage dévastateur, et une mobilité totale.

Mais l'arène devient plus dangereuse que jamais :
- Cuby lvl 4 : erratiques et résistants.
- Dashoot lvl 2 : plus agressifs.
- Banshee lvl 2 : attaques plus rapides.
- OunOun : lent, massif, cadence de tir énorme -> force à jouer plus stratégique.

Ici, le joueur doit tout combiner.
Pas de place pour l'erreur. Il peut dominer s'il maîtrise... ou tomber s'il surestime sa puissance.

Objectif final

Une fois 1000 ennemis éliminés, le boss apparaît.
Ce dernier bouscule les habitudes : un seul adversaire, mais complexe, dangereux et résistant.

Il incarne le test ultime de la run : le joueur doit utiliser tout ce qu'il a appris.
Si le boss tombe, la partie se conclut par un affichage du score total, lié à l'efficacité et au style.
Le leaderboard final permet de comparer les runs et pousse les joueurs à toujours viser mieux.

Palier 1	Palier 2	Palier 3	Palier 4
Déplacement	Déplacement +	Déplacement ++	Déplacement +++
Coup de poing	Coup de poing +	Coup de poing ++	Coup de poing +++
Sprint	Sprint	Sprint +	Sprint ++
1 saut	1 saut	2 sauts	3 sauts
Multi-saut	Multi-saut	Multi-saut	Multi-saut +
Pilonnage	Pilonnage	Pilonnage	Pilonnage

évolution des compétences au cours de la partie

Objectif : Eliminer 1000 ennemis			
Palier 1	**Palier 2**	**Palier 3**	**Palier 4**
Cuby lvl 1	Cuby lvl 2	Cuby lvl 3	Cuby lvl 4
Dashoot lvl 1	Dashoot lvl 1	Dashoot lvl 1	Dashoot lvl 2
Banshee	Banshee lvl 1	Banshee lvl 2	Banshee lvl 2
OunOun	OunOun	OunOun	OunOun

évolution du spawn des ennemis au cours de la partie

Jauge d’énergie :

Un indicateur de progression dynamique

La jauge évolue de manière fluide selon l’énergie accumulée ou perdue par le joueur. Plus elle se remplit, plus elle se rapproche du palier supérieur ; plus elle diminue, plus le joueur risque de rétrograder au palier précédent. Elle devient donc un outil de gestion constante, où chaque action impacte directement la stabilité du niveau de puissance.

Quatre paliers distincts, quatre couleurs claires

La jauge est divisée en quatre sections, correspondant chacune à un palier de puissance.
Chaque section possède une couleur unique, allant du jaune clair (palier 1) à un rouge vif et intense (palier 4).
Cette codification visuelle permet au joueur de savoir d’un coup d'œil dans quel palier il se trouve, sans avoir à lire de texte. Ce système est renforcé par la musique dynamique, qui évolue elle aussi selon le palier atteint, ajoutant une couche auditive à cette perception.

Un feedback vivant et physique

Pour rendre les gains d’énergie plus palpables, nous avons intégré une vibration subtile du cristal (l’élément visuel central de la jauge) à chaque gain d’énergie. Cette animation renforce la sensation de progression et agit comme une récompense sensorielle immédiate.
Le joueur voit, entend et sent qu’il devient plus fort — une triple confirmation qui renforce l’immersion et la lisibilité de l’interface.

Palier 1


Palier 2


Palier 3


Palier 4

Rune de compétences :

Pour accompagner le joueur dans la lecture de ses capacités actives, nous avons conçu un système de runes de compétence situé sur les côtés de l'écran. Ce système visuel permet de communiquer clairement quelles compétences sont disponibles, leur état d'activation, ainsi que leur niveau de puissance.

Visualisation des compétences actives

Chaque compétence majeure du jeu — Sprint, Pilonnage, et Saut — possède sa propre rune. Ces runes sont placées de manière fixe sur l'interface, afin de devenir des repères visuels immédiats pour le joueur, sans perturber la lisibilité centrale de l'écran.

Feedback d'état : verrouillé, actif, en cours d'utilisation

- Lorsqu'une compétence n'est pas encore débloquée (par exemple parce que le joueur n'a pas atteint le bon palier), la rune apparaît grisée, signalant son indisponibilité.

- Une fois la compétence débloquée, la rune s'illumine, indiquant sa disponibilité immédiate.

- Lors de l'utilisation d'une compétence, la rune effectue un léger tremblement, un feedback visuel discret mais expressif, renforçant la sensation d'action.

Indicateur de puissance par niveaux

Le système de runes ne sert pas uniquement à signaler la disponibilité : il indique aussi la puissance actuelle de chaque compétence.
Plus une compétence est renforcée (via l'atteinte d'un palier supérieur), plus la rune affiche des «+» visibles sur ses côtés, symbolisant son niveau d'amélioration.
Ce système permet au joueur de ressentir sa montée en puissance, tout en lui offrant une lecture simple et non verbale de l'état de ses outils.

Palier 1

Palier 4

Saut

Sprint

Pilonnage

Références Game Design

ULTRAKILL

ULTRAKILL pousse le genre du fast-FPS à son paroxysme, avec un gameplay basé sur la mobilité extrême et l'agression constante. Le jeu repose sur une mécanique centrale : se soigner en infligeant des dégâts. Cette dynamique oblige le joueur à rester constamment dans l'action et à maîtriser ses déplacements pour survivre. Bien que le bestiaire et les types d'armes soient limités, chaque combat est renouvelé par la richesse des mouvements et la précision du game feel. Le jeu valorise la fluidité, la réactivité et le style, tout en conservant une excellente lisibilité. Cette approche, centrée sur la maîtrise du flow et la pression permanente, résonne pleinement avec les intentions de Kill Dem' On.

Devil Daggers

Avec sa direction minimaliste, Devil Daggers propose une expérience de survie brutale dans une arène fermée. Le joueur dispose d'un seul type d'attaque, et doit affronter des vagues d'ennemis de plus en plus oppressantes. L'intérêt réside dans la compréhension des patterns, la gestion de l'espace et la réactivité. Il n'y a ni HUD, ni tutoriel : tout passe par l'apprentissage et l'observation. Cette économie de moyens met en valeur l'essence du gameplay pur, sans éléments superflus. Cette tension constante et lisible, ainsi que l'exigence de performance, sont des éléments que nous reprenons dans Kill Dem' On pour renforcer la clarté et l'intensité du système de jeu.

Bulletstorm

Bulletstorm se distingue par sa capacité à transformer le combat en terrain d'expérimentation. Grâce à son système de score basé sur la créativité des éliminations, il encourage le joueur à combiner armes, capacités et environnement pour maximiser son efficacité. Cette valorisation du style et de l'ingéniosité renforce l'engagement dans les affrontements, qui deviennent plus dynamiques et imprévisibles. Dans Kill Dem' On, nous reprenons cette idée de gameplay modulaire et stratégique, où chaque compétence peut être utilisée de manière contextuelle pour enrichir le flow du joueur et lui offrir une liberté d'approche sans nuire à l'intensité de l'action.

SUPERHOT

SUPERHOT revisite les codes du FPS avec une mécanique centrale : le temps ne progresse que lorsque le joueur bouge. Cette idée transforme chaque action en choix réfléchi, et place la prise de décision au cœur de l'expérience. Le jeu impose une lecture claire de l'espace et des menaces, tout en maintenant une forte pression tactique. Ce fonctionnement trouve un écho dans Kill Dem' On, où chaque action a un coût en énergie, forçant le joueur à anticiper et optimiser ses décisions en temps réel. L'inspiration ici réside dans cette capacité à proposer un gameplay tendu, stratégique et parfaitement lisible, même dans le chaos le plus total.

DOOM

DOOM est une référence majeure pour Kill Dem' On, notamment dans sa manière d'imposer un rythme de jeu extrêmement agressif et constant. Le joueur est encouragé à rester mobile en permanence, à enchaîner les éliminations pour survivre, et à utiliser l'environnement de manière dynamique. Le système de Glory Kill, qui permet de récupérer des ressources en éliminant des ennemis au corps-à-corps, transforme chaque combat en opportunité tactique. Ce gameplay renforce une boucle d'action offensive où la survie dépend directement de la capacité à rester dans l'action. Cette philosophie, mêlant efficacité, lisibilité et intensité, constitue une base solide pour notre propre approche du fast-FPS.

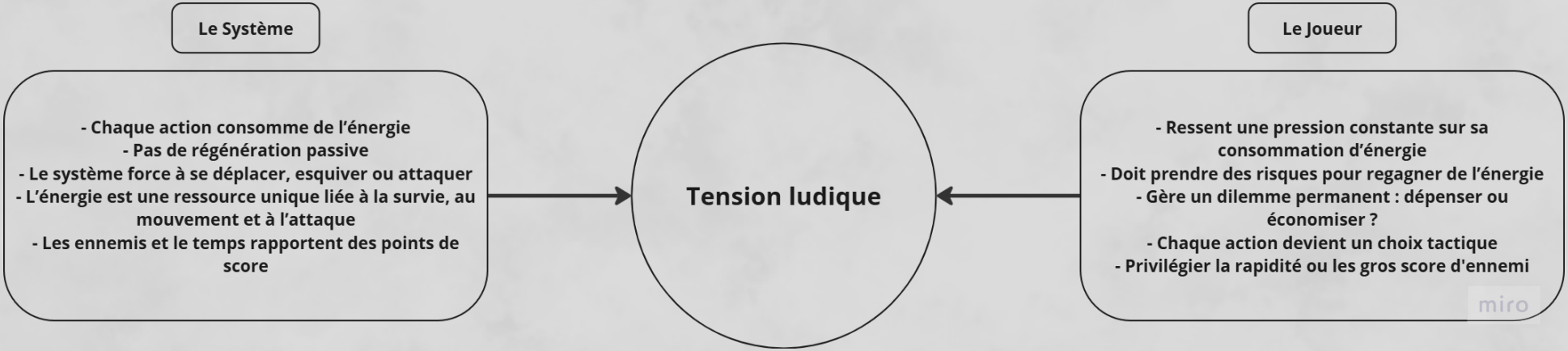
Core Système

Tension ludique :
Dans Kill Dem' On, la tension ludique naît d'un déséquilibre volontaire entre dépense et survie. Chaque action qu'elle soit offensive, défensive ou simplement liée au déplacement consomme de l'énergie. Et cette énergie est la seule ressource du jeu.

Le système, quant à lui, n'accorde aucune régénération passive.
Il contraint le joueur à agir pour survivre, mais agir lui coûte.

Ce paradoxe volontaire place le joueur dans une pression constante :
- S'il agit trop, il s'épuise
- S'il agit trop peu, il meurt
- Chaque action devient un choix tactique
- Chaque inaction devient un risque de ne plus pouvoir agir ensuite

Le schéma montre cette dualité :
- Le système impose une pression énergétique.
- Le joueur réagit par des décisions risquées, parfois urgentes, mais toujours calculées.



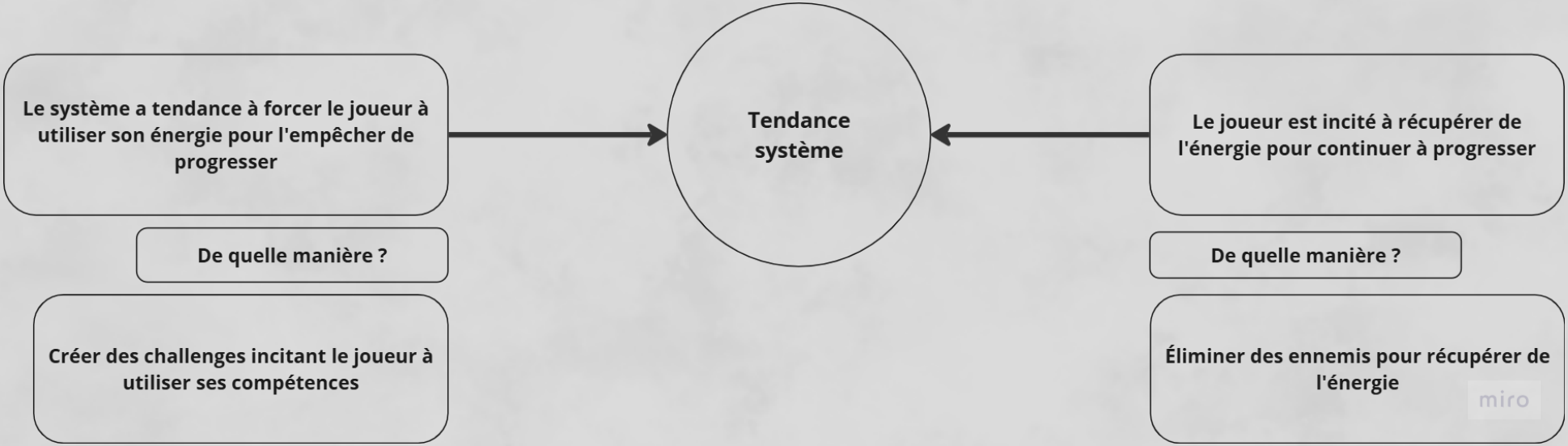
Tendance Système :

Dans Kill Dem' On, le système ne cherche pas à équilibrer la partie.
Il est conçu pour forcer la consommation d'énergie à travers chaque élément du gameplay. Le but n'est pas de maintenir un état stable, mais de mettre le joueur sous pression permanente, en l'obligeant à puiser dans sa ressource vitale pour continuer à jouer.

Le simple fait de se déplacer, d'esquiver, de sauter ou d'attaquer a un coût.
Le système n'accorde aucun répit, aucun mécanisme de régénération passive.

Mais dans ce déséquilibre, le joueur dispose d'une contre-tendance :
- Il peut tuer des ennemis pour récupérer de l'énergie
- Et plus il joue avec efficacité, plus il crée lui-même les conditions de sa survie.

Cette opposition forme une boucle dynamique :
- Le système met des obstacles énergétiques
- Le joueur répond par l'élimination



Boucle Macro Energie - Compétence :

La boucle centrale du jeu est une dynamique continue entre dépense d'énergie pour agir et élimination d'ennemis pour la récupérer. Chaque action du joueur qu'elle soit offensive ou défensive coûte de l'énergie. Et cette énergie ne peut être récupérée qu'en éliminant des ennemis et en absorbant les orbes qu'ils laissent derrière eux.
Cette boucle se divise en deux axes :

1. Récupération de l'énergie

La partie gauche de la boucle illustre la séquence de survie :
- Le joueur entre dans un état de besoin énergétique
- Il recherche des ennemis capables de lui fournir cette ressource
- Il les élimine, puis récupère l'orbe générée
- Ce qui lui permet de soutenir ses futures actions

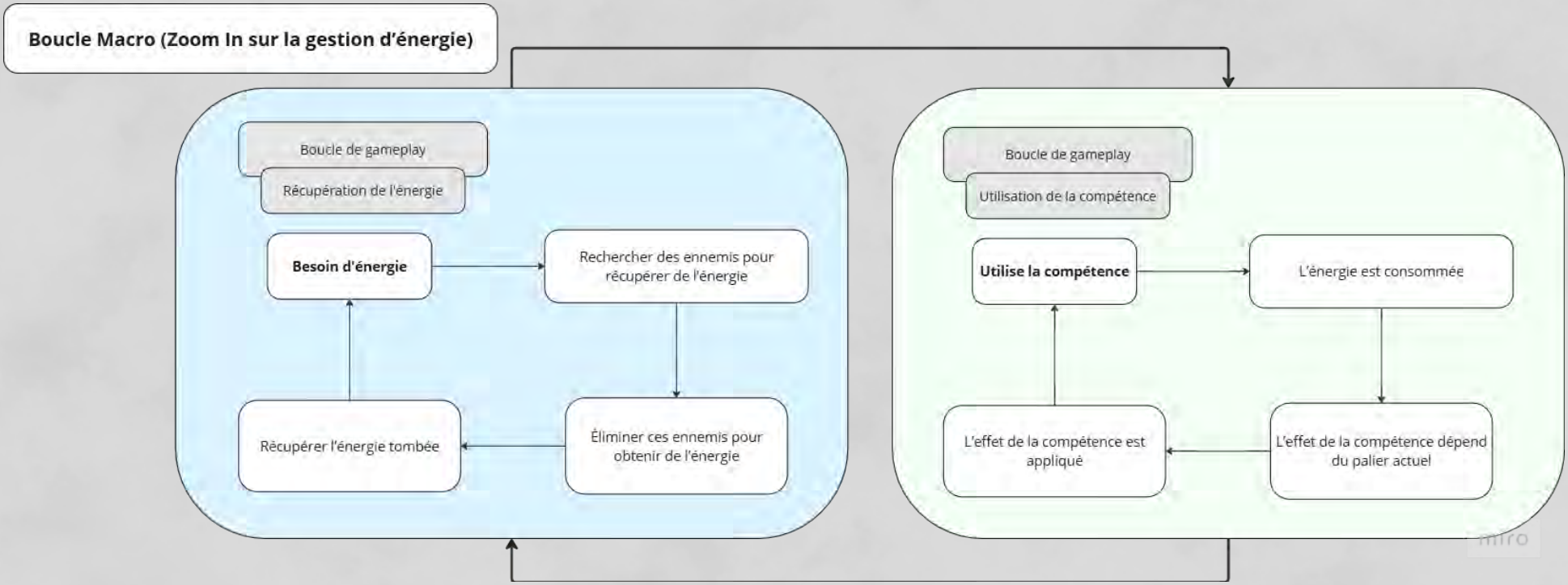
Cette moitié de la boucle pousse à l'engagement offensif, et récompense l'initiative.

Une fois l'énergie récupérée, le joueur peut l'investir dans des actions plus puissantes. C'est dans cette logique de dépense maîtrisée que s'inscrit la deuxième moitié de la boucle.

2. Utilisation de la compétence

La partie droite montre la mise en action :
- Le joueur utilise une compétence (saut, tir, poing, sprint, pilonnage)
- Ce qui entraîne une consommation immédiate d'énergie
- L'effet de la compétence dépend directement du palier atteint.

Plus le joueur monte, plus ses compétences gagnent en puissance, portée et impact mais leur coût énergétique reste élevé, renforçant la nécessité d'une bonne gestion.
Cette moitié structure l'idée que plus on gagne d'énergie, plus on peut l'investir dans des actions fortes mais à un coût.


Boucle Macro (Zoom In sur la gestion d'énergie)
Boucle de gameplay
Récupération de l'énergie
Besoin d'énergie
Rechercher des ennemis pour récupérer de l'énergie
Éliminer ces ennemis pour obtenir de l'énergie
Récupérer l'énergie tombée
Boucle de gameplay
Utilisation de la compétence
Utilise la compétence
L'énergie est consommée
L'effet de la compétence dépend du palier actuel
L'effet de la compétence est appliqué

Boucle de gameplay générale :

La boucle de gameplay de Kill Dem' On repose sur un enchaînement d'actions maîtrisées, où chaque compétence suit un cycle de décision, d'exécution et de résolution.

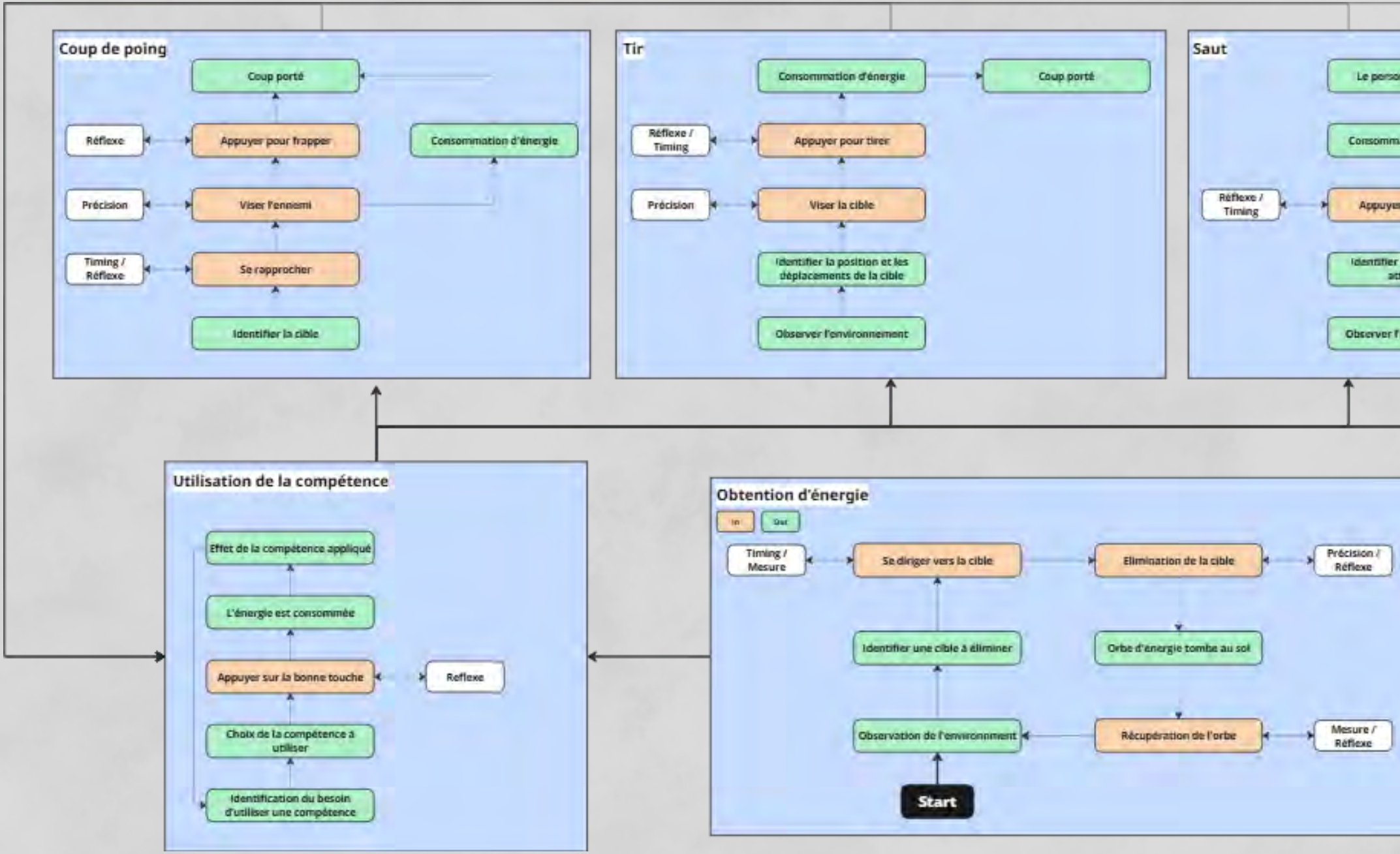
Chaque action est rattachée à un système énergétique commun, et chaque compétence est conçue pour s'inscrire dans une dynamique de rythme et de gestion des ressources.

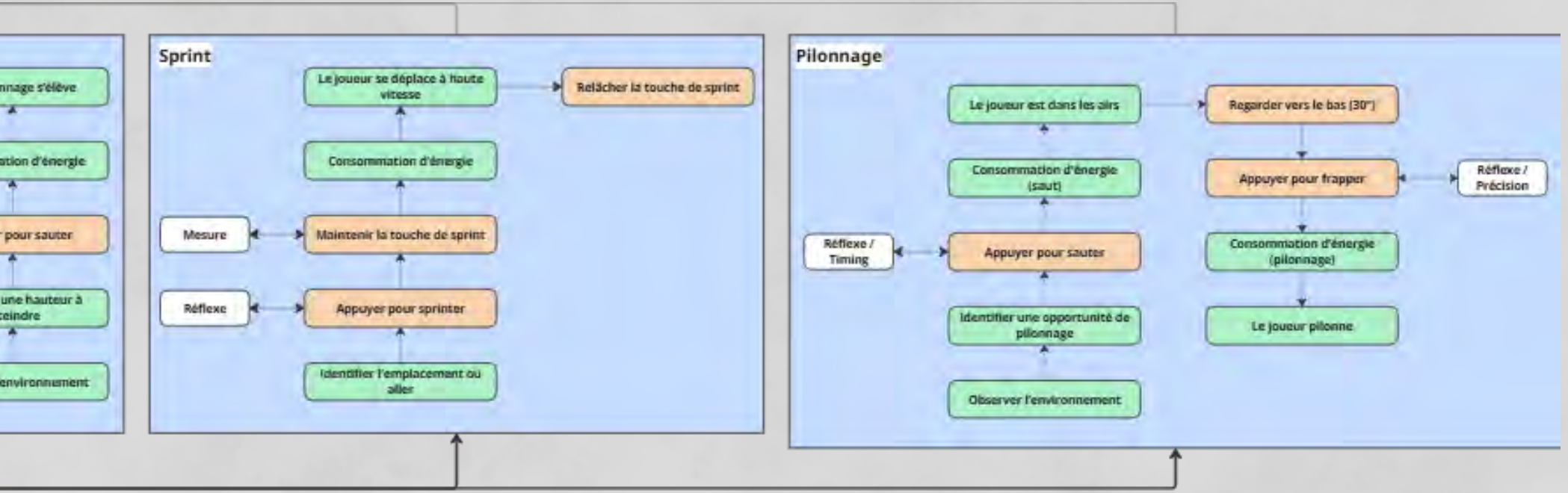
Un cycle global au cœur de chaque compétence :
Chaque action qu'elle soit offensive ou utilitaire suit une boucle commune :
- Le joueur identifie une intention (frapper, sauter, fuir, viser...)
- Il utilise une compétence en réponse à cette intention
- Cela entraîne une consommation d'énergie
- Si l'action aboutit, l'ennemi meurt et libère une orbe
- Le joueur la récupère, relançant ainsi le cycle

Ce cycle peut démarrer depuis une envie de dépenser (exécuter une action forte) ou depuis un besoin de récupérer (chercher une orbe).
Compétences intégrées à la boucle
Le schéma montre que chaque compétence est un segment spécifique d'un système global, et que toutes se rejoignent au sein d'un cadre énergétique unique.

Elles diffèrent par leur usage, leur timing, leur portée ou leur effet, mais elles partagent :
- un coût énergétique
- une exigence d'efficacité (viser, frapper, réussir)
- une finalité commune : favoriser l'élimination et la récupération

Exemples :
- Le tir permet d'abattre rapidement une cible à distance pour sécuriser une orbe
- Le saut permet d'atteindre une plateforme stratégique ou d'éviter un danger
- Le pilonnage permet d'achever plusieurs ennemis d'un coup mais à condition d'avoir pris de la hauteur et de viser juste
- Le sprint offre une mobilité temporaire au prix d'une consommation continue
- Le coup de poing est risqué mais économique, et peut être très rentable s'il est bien placé (ex : sur un point faible)


Coup de poing
Coup porté
Réflexe
Appuyer pour frapper
Consommation d'énergie
Précision
Viser l'ennemi
Timing / Réflexe
Se rapprocher
Identifier la cible
Tir
Consommation d'énergie
Coup porté
Réflexe / Timing
Appuyer pour tirer
Précision
Viser la cible
Identifier la position et les déplacements de la cible
Observer l'environnement
Saut
Réflexe / Timing
Utilisation de la compétence
Effet de la compétence appliqué
L'énergie est consommée
Appuyer sur la bonne touche
Reflexe
Choix de la compétence à utiliser
Identification du besoin d'utiliser une compétence
Obtention d'énergie
In
Out
Timing / Mesure
Se diriger vers la cible
Elimination de la cible
Précision / Réflexe
Identifier une cible à éliminer
Orbe d'énergie tombe au sol
Observation de l'environnement
Récupération de l'orbe
Mesure / Réflexe
Start



Flow Chart des actions du joueur

FlowChart Objectifs :

Ce flowchart illustre le cycle principal d'une partie dans Kill Dem'On.
Le joueur débute la partie et commence immédiatement à éliminer des ennemis.

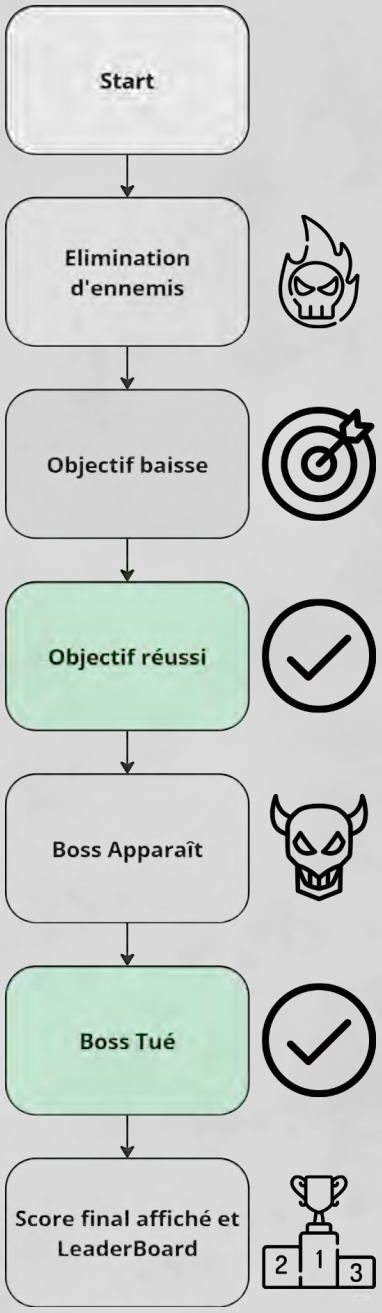
Chaque élimination fait baisser un compteur d'objectif : l'élimination de 1000 ennemis est nécessaire pour progresser.

Une fois cet objectif atteint, il est validé automatiquement, ce qui déclenche l'apparition du boss final.

Le joueur doit alors mobiliser toutes ses compétences, acquises et maîtrisées pendant la partie, pour le vaincre.

Une fois le boss éliminé, le jeu se conclut avec l'affichage du score final et l'enregistrement dans le leaderboard, où la performance peut être comparée à celle des autres joueurs.

Ce système repose sur une logique simple mais exigeante : "Tu avances si tu élimines. Tu gagnes si tu maîtrises."



3Cs

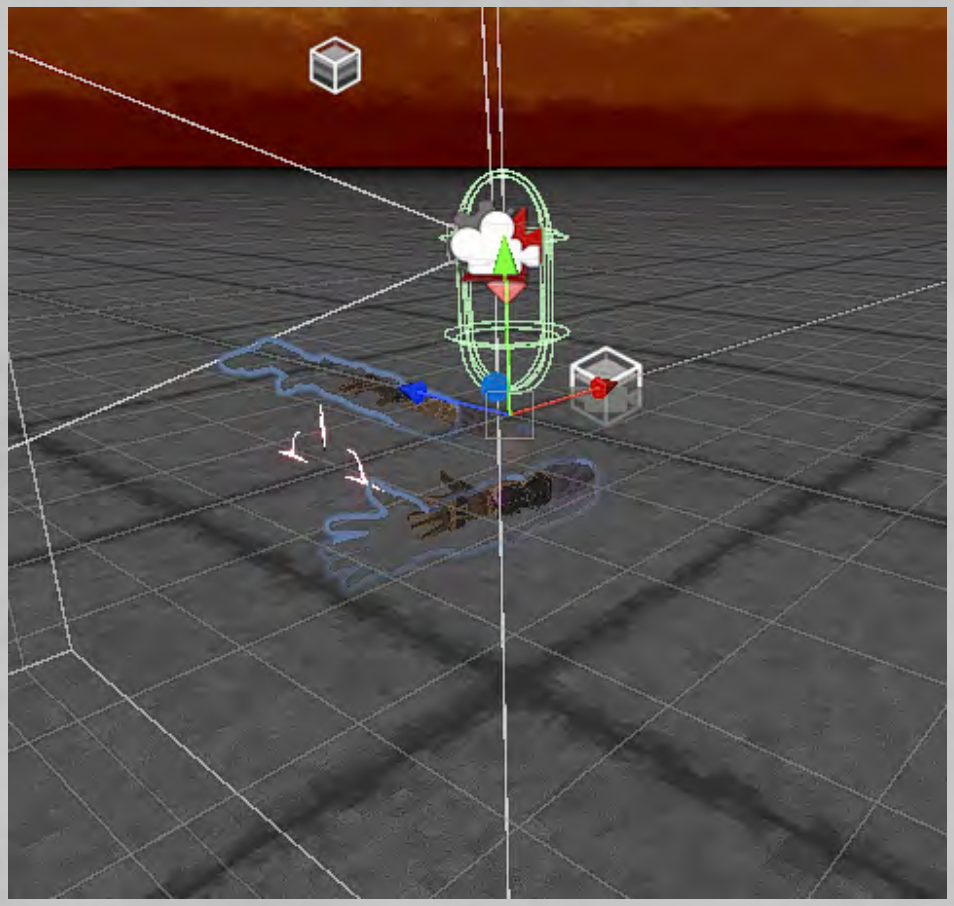
Géneral :

Character :

Dans Kill Dem'On, le personnage n'est pas un simple avatar que l'on contrôle : il est la caméra elle-même. Ce choix de design renforce l'immersion et place le joueur au cœur de l'action, sans distance ni interface superflue. Tout passe par lui : la vitesse, la violence, la puissance... et surtout, la montée en tension.

Le personnage incarne à lui seul les sensations fondamentales du fast FPS : fluidité, réactivité, brutalité. Il se déplace avec précision, enchaîne les actions sans rupture, et répond instantanément à chaque impulsion du joueur. Mais il n'est jamais figé. Sa nature évolue en fonction de l'état du jeu, et plus précisément de son niveau d'énergie.
Au début d'une partie ou d'un niveau, le personnage paraît relativement modeste. Rapide, certes, mais encore fragile, encore "humain". C'est à travers la montée en paliers que le joueur ressent une transformation physique et symbolique : à chaque seuil franchi, il devient une menace plus imposante, plus rapide, plus destructrice. Le gameplay ne change pas seulement : la posture du joueur change, son rapport au monde aussi. Ce n'est plus lui qui fuit les ennemis, ce sont eux qui devraient le craindre.
Ce système est directement lié à la barre d'énergie, qui ne sert pas uniquement à activer des compétences : elle est le reflet de l'état du joueur. Chaque action consomme de l'énergie, chaque palier atteint débloque une nouvelle puissance. Ainsi, on ne joue pas de la même manière selon son niveau d'énergie : en haut de la barre, on est un prédateur ; en bas, une cible vulnérable.

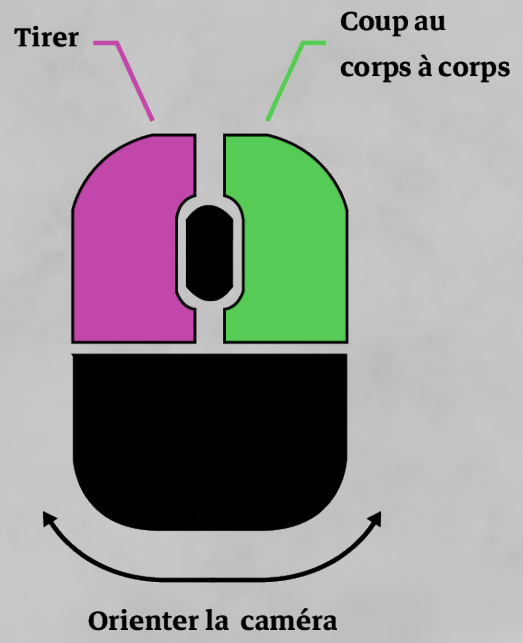
Le joueur ressent dans son corps virtuel cette tension entre domination et survie. Il devient littéralement ce que le jeu lui permet d'être : une force incontrôlable... ou un survivant à bout de souffle.



représentation du player dans la scène unity

Controls :

Le jeu est conçu pour se jouer exclusivement avec un clavier et une souris, une configuration intuitive et adaptée aux fast FPS. En effet, la souris offre une précision bien supérieure à celle d'une manette pour viser les ennemis ou utiliser ses compétences de manière plus efficace.
Cette disposition permet aux joueurs de maîtriser rapidement les combinaisons de touches,favorisant un gameplay fluide et instinctif.



Caméra :

Dans notre jeu, nous avons choisi une caméra à la première personne pour maximiser l'immersion et la réactivité, essentielles dans un fast FPS.

Précision et fluidité

La caméra offre un contrôle direct et instantané, aligné sur les mouvements du joueur, permettant une visée intuitive et une parfaite réactivité.

Immersion totale

En incarnant le point de vue du personnage, le joueur voit et ressent chaque action, renforçant le lien avec l'environnement et l'intensité dugameplay.

Sensations

Les variations dynamiques du champ de vision (FOV) et les effets visuels, comme le vignettage ou les indicateurs de direction des dégâts, amplifient la vitesse, l'urgence et le danger à chaque instant.

screenshoot de Kill Dem'on

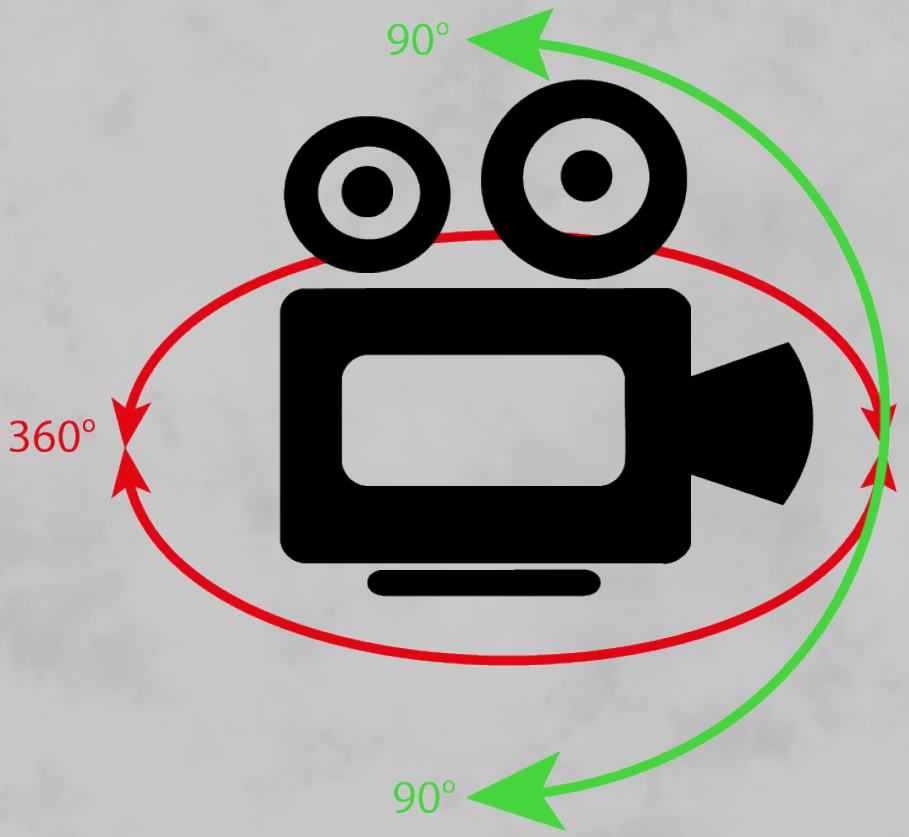
Cette caméra n'est pas seulement un outil de vision, mais un élément central du gameplay, conçu pour transmettre chaque action en une expérience rapide et viscérale.

Rotation horizontale illimitée

La caméra permet au joueur de pivoter librement sur 360° sur l'axe horizontal, lui donnant la possibilité d'observer son environnent dans toutes les directions. Cette liberté est indispensable dans un FPS rapide où la réactivité et l'anticipation sont essentielles.

Limitation verticale

Sur l'axe vertical, la caméra est limitée entre -90° et 90°. Ces limites empêchent le joueur de dépasser des angles extrêmes qui peuvent nuire à la lisibilité et à l'expérience visuelle. Cette restriction garantit que la caméra reste intuitive et toujours orientée vers les zones d'action.
La caméra s'adapte précisément aux mouvements de la souris, assurant une réponse immédiate et fluide. Cela permet au joueur de rester concentré sur l'action tout en préservant une certaine ergonomie, même dans les moments de forte intensité.



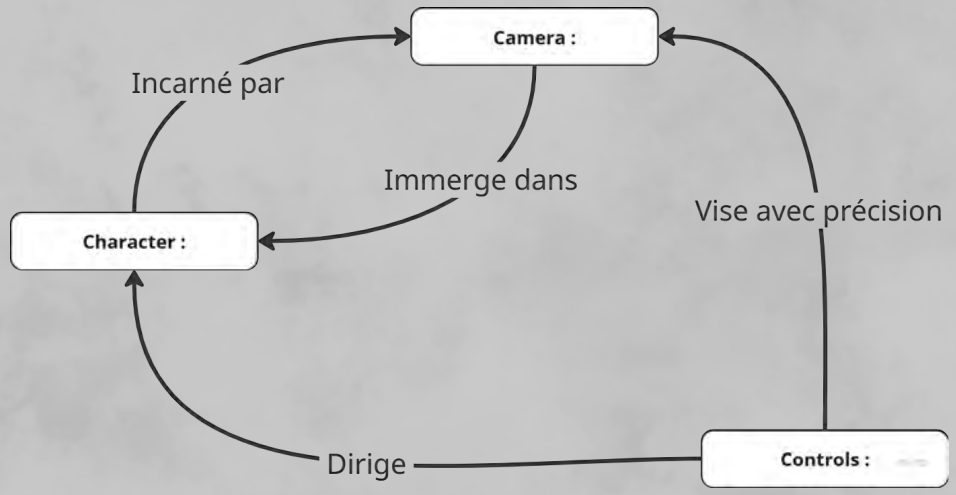
Lien 3Cs :

Dans Kill Dem'On, le character, la caméra et les contrôles ne sont pas des éléments séparés, mais les trois piliers d'une expérience fusionnelle, conçus pour transmettre des sensations de vitesse, de puissance et de fluidité. Ensemble, ils forment un système unifié au service de l'immersion et du gameplay.

Le personnage n'est pas une entité distante, mais le prolongement direct du joueur : il est la caméra. Ce choix renforce le sentiment d'incarnation, transformant chaque action en un ressenti viscéral. La caméra à la première personne, extrêmement réactive, traduit chaque mouvement du joueur avec une précision millimétrée. Les variations de champ de vision (FOV), les effets visuels et les limitations pensées pour la lisibilité participent tous à créer une perception dynamique du monde, où chaque saut, chaque tir et chaque dash devient une sensation physique.

Ce ressenti ne serait rien sans des contrôles parfaitement calibrés. Clavier et souris sont les outils idéaux pour exprimer la nervosité et la précision attendues dans un fast FPS. Ils permettent de réagir au quart de seconde, d'enchaîner des actions complexes de manière fluide, et de garder toujours un contrôle total, même dans le chaos.

Ensemble, ces éléments incarnent la philosophie de Kill Dem'On : un jeu où l'on ressent ce qu'on fait, où chaque mouvement est une décision, et où la maîtrise passe autant par le corps que par l'esprit.



Dans Kill Dem'On, la vignette de dégâts est un élément essentiel de l'interface sensorielle. Elle a pour rôle de traduire visuellement l'état de danger du joueur, en s'appuyant sur une représentation corporelle immersive et organique.

Un effet rouge veineux en réponse aux dégâts

Lorsqu'un ennemi inflige des dégâts au joueur, une vignette rouge apparaît progressivement sur les bords de l'écran. Ce n'est pas un simple filtre : elle est conçue pour évoquer des veines qui gonflent, comme si le corps du personnage réagissait à la douleur ou au stress. Ce choix visuel renforce le sentiment de tension physiologique, tout en étant immédiatement lisible.

Une intensité évolutive selon la gravité

La vignette n'est pas fixe. Plus le joueur perd de vie, plus elle s'étend vers le centre de l'écran, créant un effet d'étouffement visuel. Cela permet d'alerter le joueur sans afficher de texte, en générant une sensation de suffocation ou de danger imminent. L'invasion progressive de l'image renforce la pression, et oblige le joueur à réagir vite pour éviter la mort.

Une alerte renforcée en situation critique

Lorsque le joueur atteint un état de mort imminente, la vignette passe à un niveau supérieur : elle devient plus intense, plus envahissante, et son opacité augmente légèrement. Cela déclenche une alerte visuelle forte, qui signale que la prochaine erreur peut être fatale. Ce système permet de garder le joueur dans un état d'alerte constant, tout en respectant l'immersion.


999

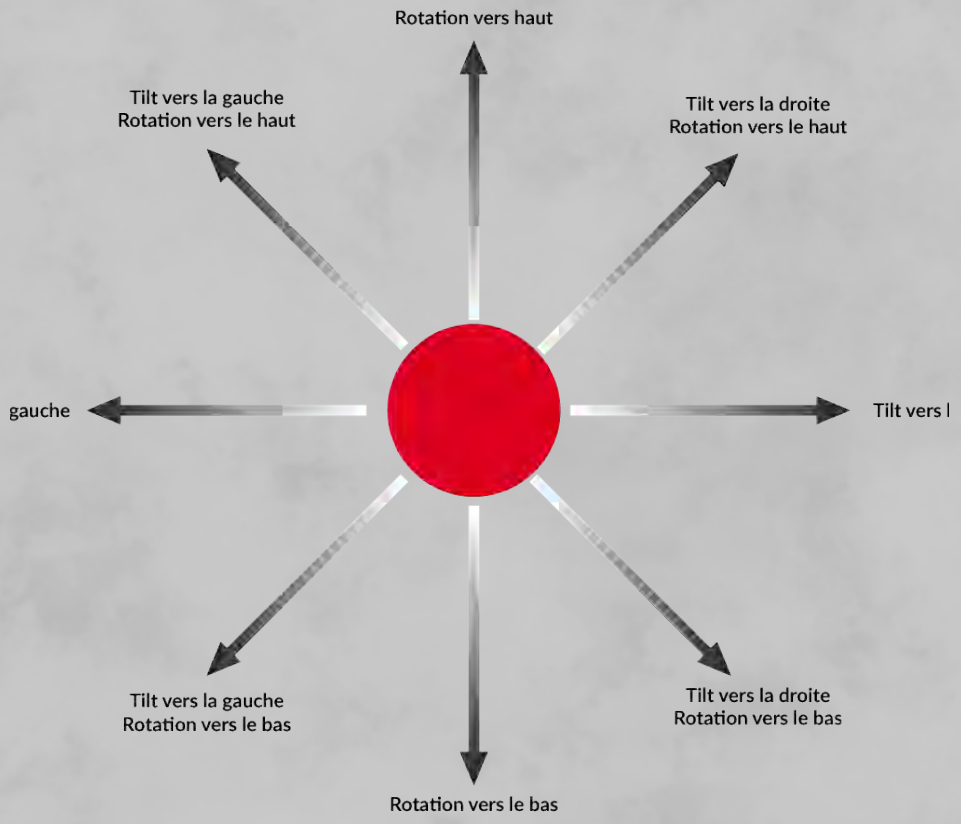

1000


1000

Déplacement :

Control :

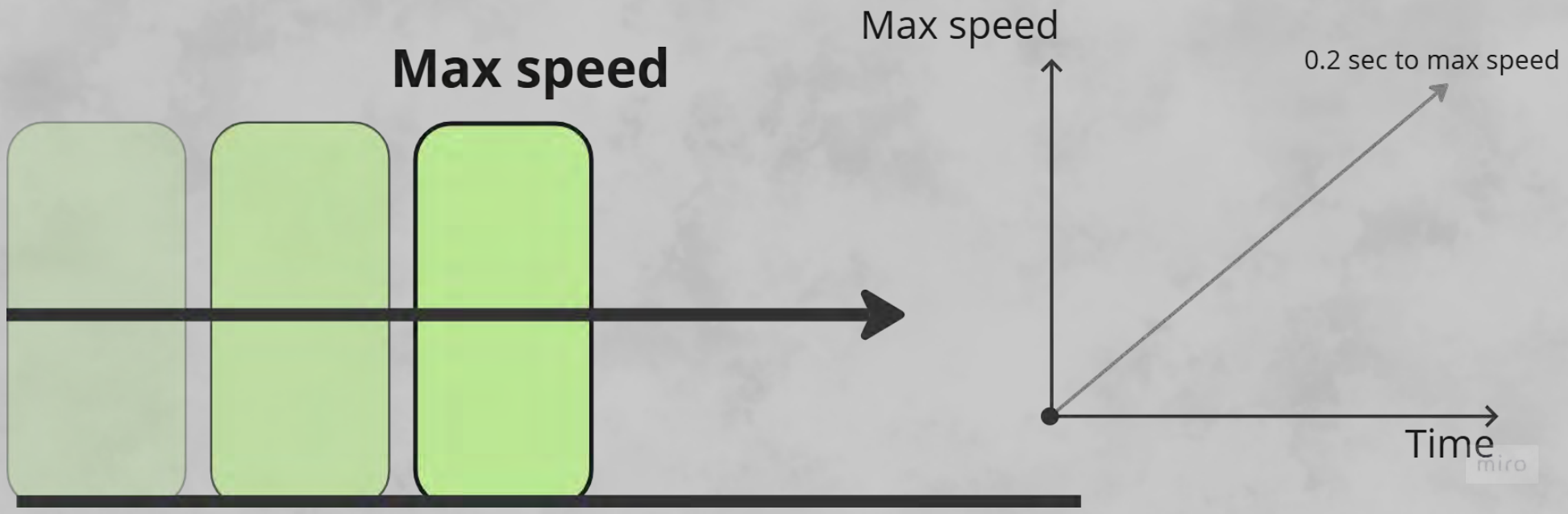
Le système de déplacement dans Kill Dem'On repose sur un fonctionnement volontairement simple et intuitif. Le joueur peut se déplacer sur huit axes, en combinant les directions de base (avant, arrière, gauche, droite) avec les mouvements diagonaux. Ce système classique, entièrement géré via les touches du clavier (ZQSD), permet une prise en main rapide et efficace, parfaitement adaptée à un gameplay rapide et nerveux. L'objectif est de laisser au joueur une liberté totale de mouvement, sans ajouter de complexité inutile, afin qu'il puisse se concentrer sur la précision, l'esquive et le rythme des actions.



Fonctionnement :

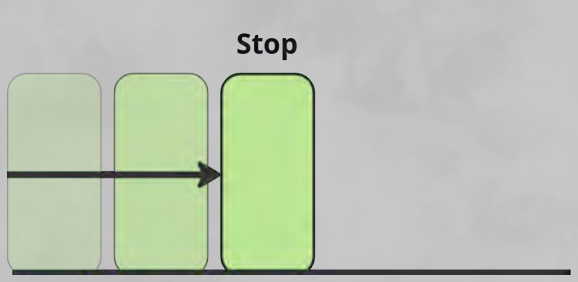
Intention :
Le système de déplacement est conçu pour offrir une expérience fluide et réaliste, tout en garantissant une réactivité optimale pour un jeu à haute intensité. Il repose sur des phases d'accélération, de décélération et de gestion des changements de direction, tout en assurant une uniformité sur différents types de terrains.

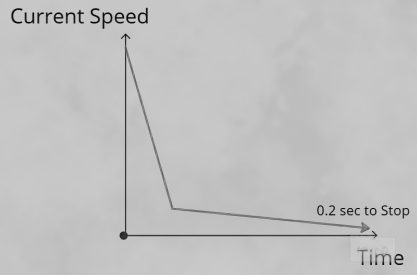
Accélération et démarrage progressif
- Lorsque le joueur commence à se déplacer, il ne démarre pas instantanément à sa vitesse maximale.
- Une courbe d'accélération est appliquée, simulant une montée en vitesse progressive.
- Cette mécanique donne un ressenti plus naturel au mouvement, tout en empêchant un démarrage trop abrupt.



Décélération et arrêt

- Lorsque le joueur cesse de se déplacer, il ne s'arrête pas instantanément.
- Un temps de décélération fixe est appliqué, quelle que soit sa vitesse au moment de l'arrêt.
- Ce temps de freinage est conçu pour offrir un arrêt net, essentiel dans un jeu rapide nécessitant des ajustements de mouvement constants.
- Cependant, une courbe de décélération progressive permet d'apporter une sensation d'inertie naturelle.

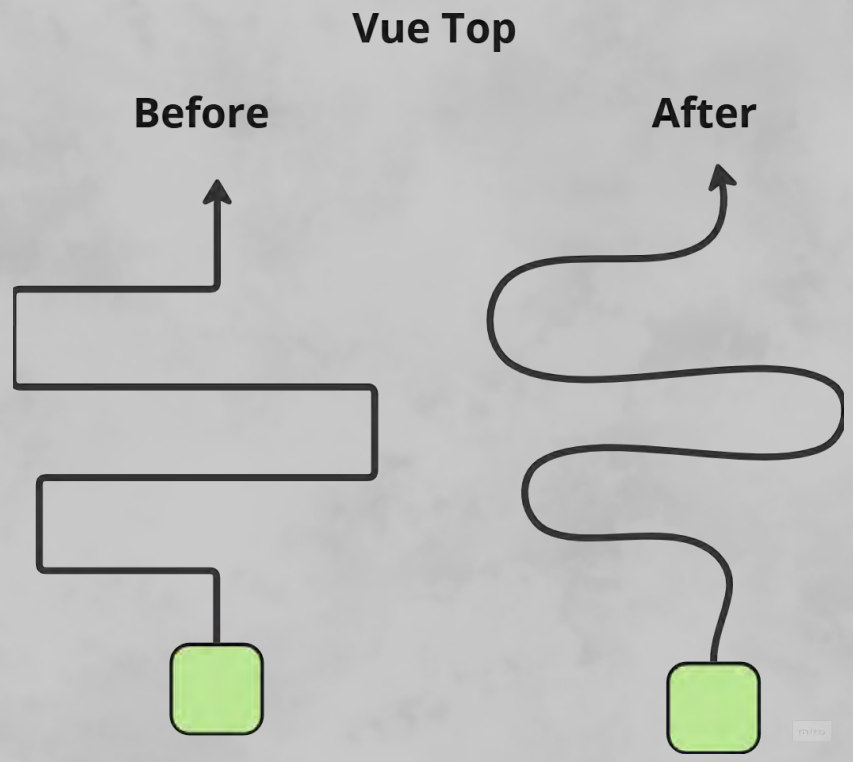




Changement de direction et rotation fluide

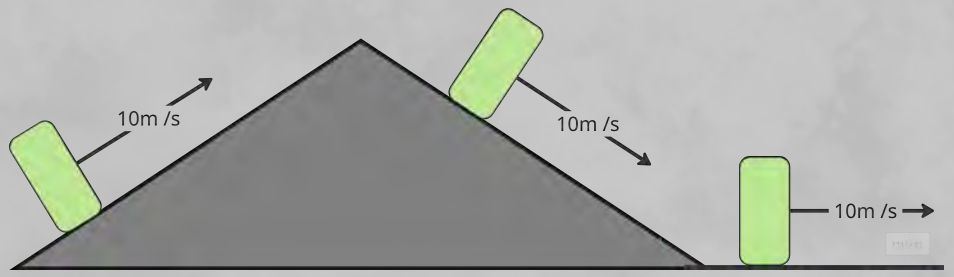
- Le joueur ne tourne pas instantanément lorsqu'il modifie sa direction.
- Un léger délai de rotation est appliqué pour adoucir les virages et éviter une sensation de contrôle trop rigide.

Ce système améliore l'expérience de déplacement en première personne en réduisant les à-coups lors des changements brusques de direction.



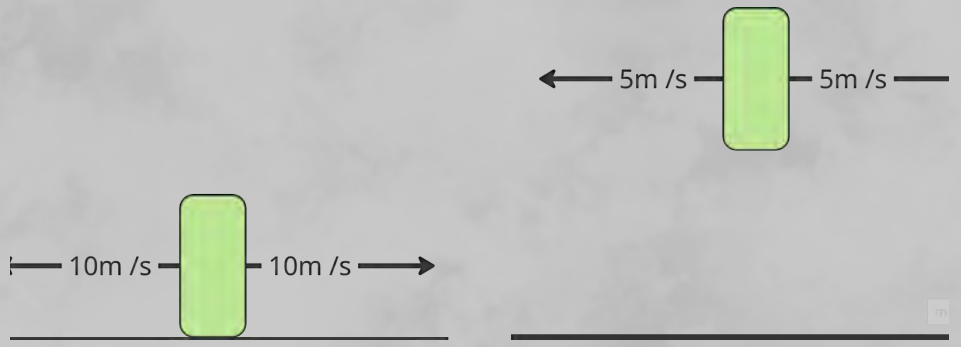
Uniformité du déplacement sur les surfaces

- La vitesse du joueur reste constante, qu'il se déplace en montée ou en descente.
- Ce choix vise à éviter les interruptions de rythme et à garantir une expérience uniforme, sans favoriser ou pénaliser certaines zones du terrain.
- Ainsi, les joueurs ne sont pas influencés négativement par la topographie du niveau et peuvent se concentrer sur le combat et les déplacements stratégiques.



Mobilité dans les airs

- En l'air, la capacité de mouvement est légèrement réduite par rapport aux déplacements au sol.
- Cela empêche le joueur d'avoir une liberté de déplacement aérien excessive dès le début du jeu.
- Dans les paliers avancés, les améliorations de vitesse permettent d'augmenter la maniabilité aérienne, offrant ainsi une évolution progressive du ressenti du déplacement.



Caméra :

Pour renforcer l'immersion du joueur, nous avons conçu les déplacements en caméra de manière à imiter les mouvements du corps humain. L'objectif était de ne pas seulement voir le monde, mais le ressentir à travers des ajustements subtils de la caméra. Lors des déplacements latéraux (gauche/droite), un léger tilt de la caméra accompagne le mouvement, simulant l'inclinaison naturelle du corps lors d'un appui dynamique. De même, avancer provoque un léger recul et une rotation vers l'avant, tandis que reculer génère une sensation d'instabilité et de tension par un mouvement inverse. Ces micro-animations ne sont pas simplement esthétiques : elles participent à l'ancrage du joueur dans l'action, en créant un lien sensoriel fort entre les déplacements et la perception de l'environnement.

Caméra Tilt :

L'inclinaison de la caméra, accompagne les mouvements du joueur pour traduire visuellement la dynamique des déplacements.

Inclinaison latérale :

Lorsque le joueur se déplace latéralement, la caméra s'incline légèrement sur l'axe Z, simulant le basculement naturel du personnage. Le basculement s'effectue progressivement jusqu'à se bloquer à un certain degré.

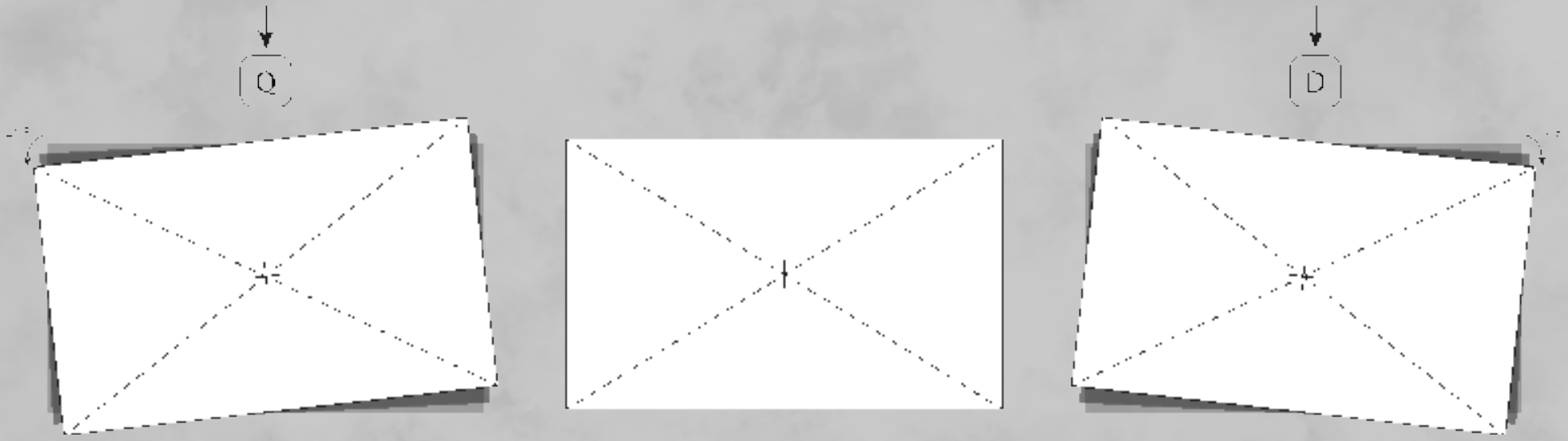
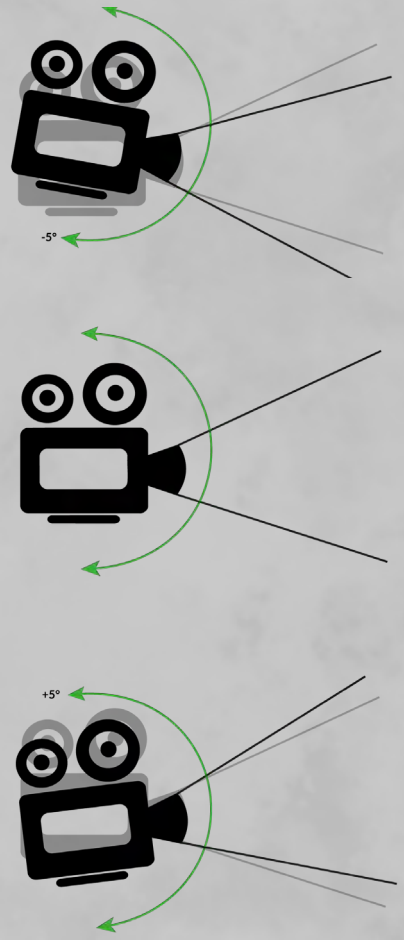


schéma du tilt de la caméra

Rotation Caméra sur Z :

Afin d'accentuer la sensation de mouvement et d'immersion dans Kill Dem'On, nous avons intégré une légère rotation de la caméra sur l'axe Z lorsque le joueur avance. Ce basculement minime, presque imperceptible à l'arrêt, devient perceptible en mouvement, et contribue à simuler une instabilité légère et dynamique du corps en course, proche de ce que l'on ressentrait en sprintant.

Cet effet visuel subtil amplifie la perception de vitesse sans altérer la lisibilité ou la précision de la visée. Il renforce le caractère nerveux du gameplay tout en maintenant un certain confort visuel. Ce type de micro-animation participe à rendre la caméra vivante, comme si elle réagissait aux mouvements du corps du personnage, et donc du joueur lui-même.



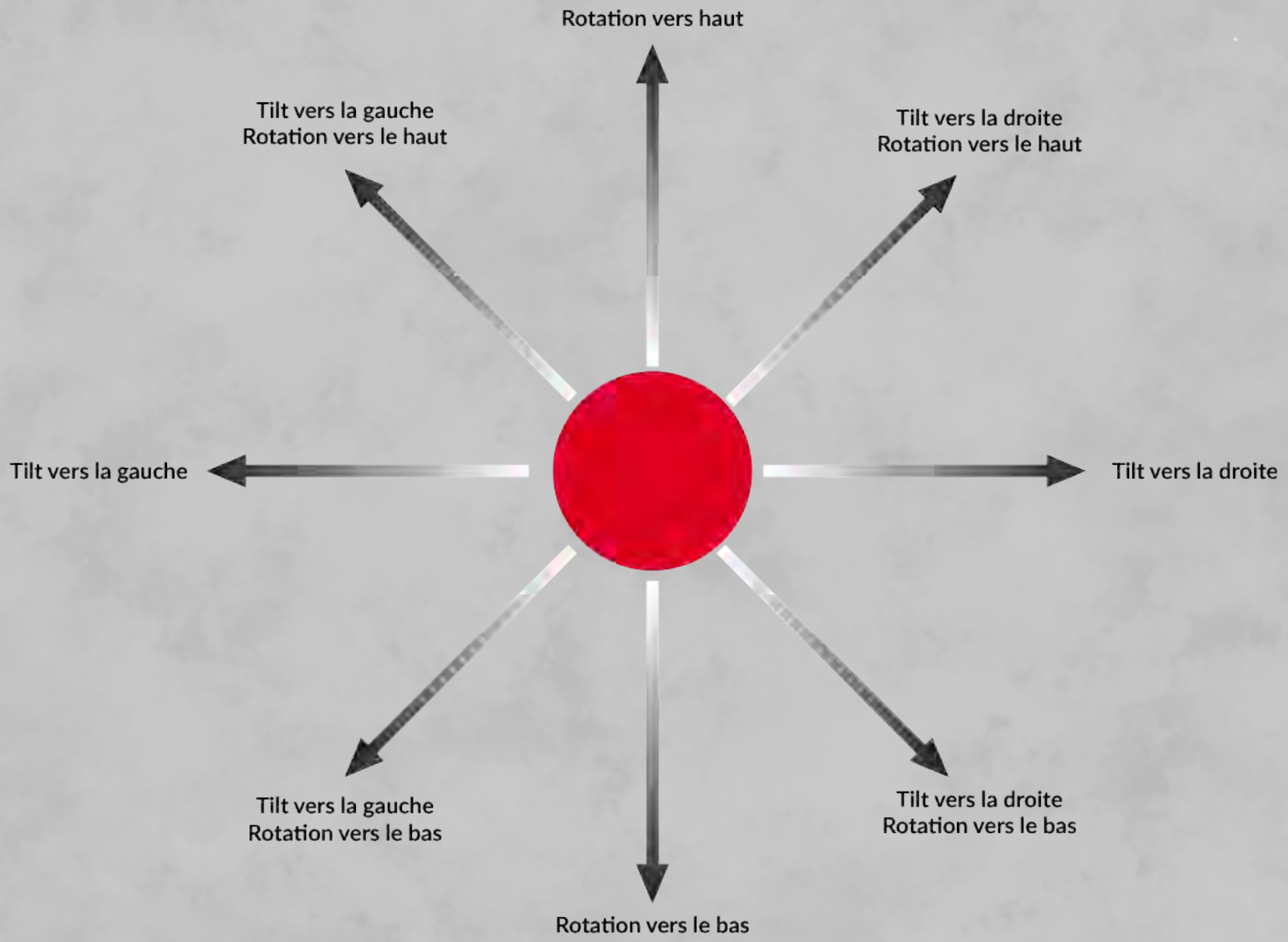
Les bras : un repère spatial et sensoriel essentiel :

Dans Kill Dem'On, les bras du personnage ne sont pas de simples éléments décoratifs ou esthétiques : ils jouent un rôle fondamental dans le ressenti global du joueur, notamment en ce qui concerne la perception de l'espace, du mouvement et de l'impact de ses actions.

Placés stratégiquement à l'écran, les bras servent avant tout de repère visuel. Ils ancrent le joueur dans le monde du jeu, lui rappellent constamment sa position corporelle virtuelle, et facilitent l'évaluation des distances ou de la direction des déplacements. Ce point d'ancrage visuel est d'autant plus important dans un fast FPS, où les mouvements de caméra sont rapides et constants : sans ce repère, le joueur pourrait perdre la sensation de stabilité ou de cohérence dans l'espace.

Au-delà de leur position, un travail d'animation précis a été réalisé sur les bras pour renforcer le dynamisme de l'expérience. Ils réagissent non seulement aux mouvements naturels de la caméra (sprint, saut, pivot), mais sont aussi animés de manière plus poussée à certains moments clés du gameplay. Ces animations «forcées» (par exemple lors de l'utilisation d'une capacité, d'un impact ou d'un changement d'état) viennent enrichir le feedback visuel et renforcer la puissance ressentie des actions.

Ce soin apporté aux bras contribue à rendre le personnage plus présent, plus crédible, et surtout plus lisible. Il renforce le lien entre le joueur et son avatar, en traduisant chaque action par une réaction physique claire, rythmée, et ancrée dans l'univers du jeu.


Rotation vers haut
Tilt vers la gauche
Rotation vers le haut
Tilt vers la droite
Rotation vers le haut
Tilt vers la gauche
Tilt vers la droite
Tilt vers la gauche
Rotation vers le bas
Tilt vers la droite
Rotation vers le bas
Rotation vers le bas

Le saut :

Le saut est une mécanique essentielle du gameplay, influençant la mobilité, l'esquive et l'exploration. Son efficacité évolue avec les paliers, modifiant la hauteur maximale atteinte et la consommation d'énergie.

Mécanique :

Activation et Contrôle

- Le joueur saute en appuyant sur la touche d'action.
- La hauteur du saut évolue en fonction du palier atteint.

Chaque saut consomme de l'énergie, mais cette consommation diminue progressivement avec la progression du joueur.

Paramètres ajustables

- Hauteur du saut : Plus le joueur monte en paliers, plus il peut atteindre des hauteurs élevées.
- Consommation d'énergie : Diminue avec les paliers, rendant les sauts plus efficaces énergétiquement.

Le multi saut :

Le multi-saut est une amélioration du saut classique, permettant au joueur d'enchaîner plusieurs sauts consécutifs. Il est disponible à partir des paliers avancés, offrant une meilleure mobilité, plus de possibilités d'esquive et un accès facilité aux zones en hauteur.

Mécanique :

Activation et Exécution

- Le joueur peut enchaîner plusieurs sauts consécutifs en appuyant plusieurs fois sur la touche d'action.
- Chaque saut supplémentaire augmente proportionnellement la hauteur et la consommation d'énergie.
- Les sauts consécutifs se réinitialisent dès que le joueur touche le sol.

Paramètres et caractéristiques

Nombre de sauts maximum :

Palier 1 & 2 : Pas de multi-saut.

Palier 3 : 2 sauts consécutifs possibles.

Palier 4 : 3 sauts consécutifs possibles.

Consommation d'énergie

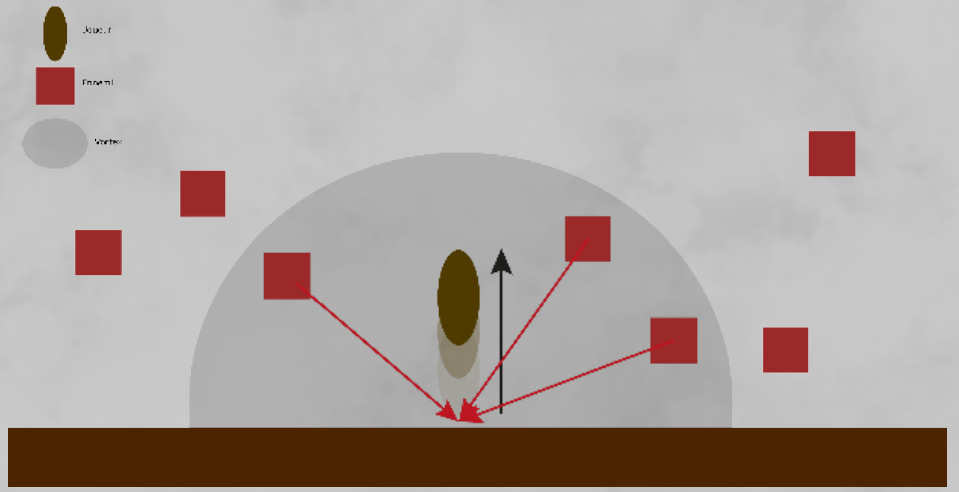
- Chaque saut supplémentaire coûte autant qu'un saut normal.
- Enchaîner 3 sauts consécutifs consomme 3 fois le coût d'un saut unique.

Le vortex :

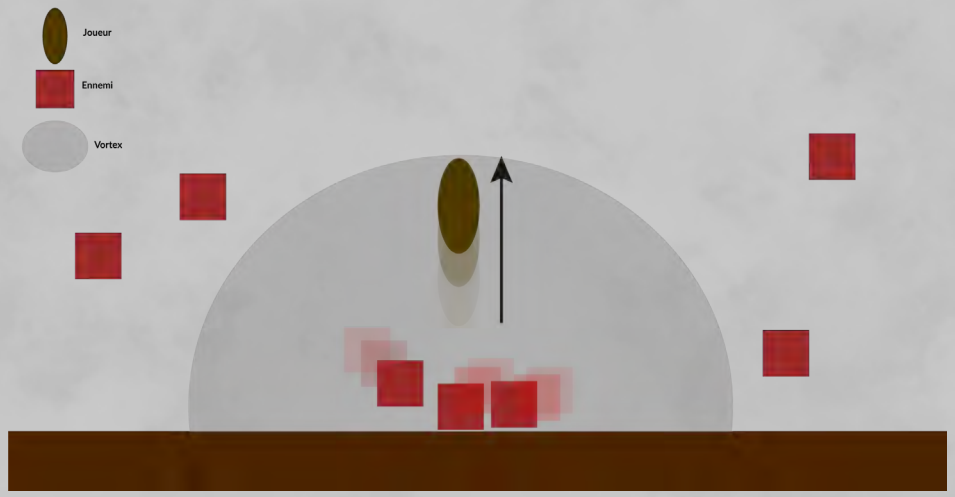
Au cours du développement, nous avons constaté que le saut, bien qu'essentiel au départ, perdait progressivement son intérêt une fois que le dash offensif (lié au coup de poing) avait été intégré au gameplay. Le dash apportait une mobilité horizontale agressive, souvent plus utile que le saut dans un fast FPS. Il devenait donc nécessaire de redonner au saut une fonction propre, claire et stratégique.
C'est dans cette logique qu'est née la mécanique du vortex.

Lorsqu'un saut est effectué, une sphère d'attraction est brièvement générée au point d'appel du saut : le vortex. Cette sphère attire les ennemis présents dans son rayon vers son centre. Ce comportement simple visuellement crée pourtant plusieurs opportunités tactiques : le vortex permet de regrouper les ennemis pour mieux les enchaîner, facilite le ciblage (en les stabilisant temporairement), et peut également servir de moyen d'évasion, en perturbant leur position au moment de fuir.
Mais sa synergie la plus forte reste avec la mécanique de pilonnage. Le vortex devient un outil de mise en scène : on attire les ennemis... pour mieux les pulvériser. Cela transforme un simple saut en une action à fort potentiel d'impact, redonnant au mouvement vertical une vraie valeur ajoutée dans le système de jeu.
Cette mécanique a eu l'effet escompté : elle a rééquilibré les dynamiques de mobilité, renforcé la diversité d'approches possibles, et donné au saut une place unique et indispensable dans le flow du gameplay.

Ici nous avons un state du jeu ou le joueur est entouré d'ennemi

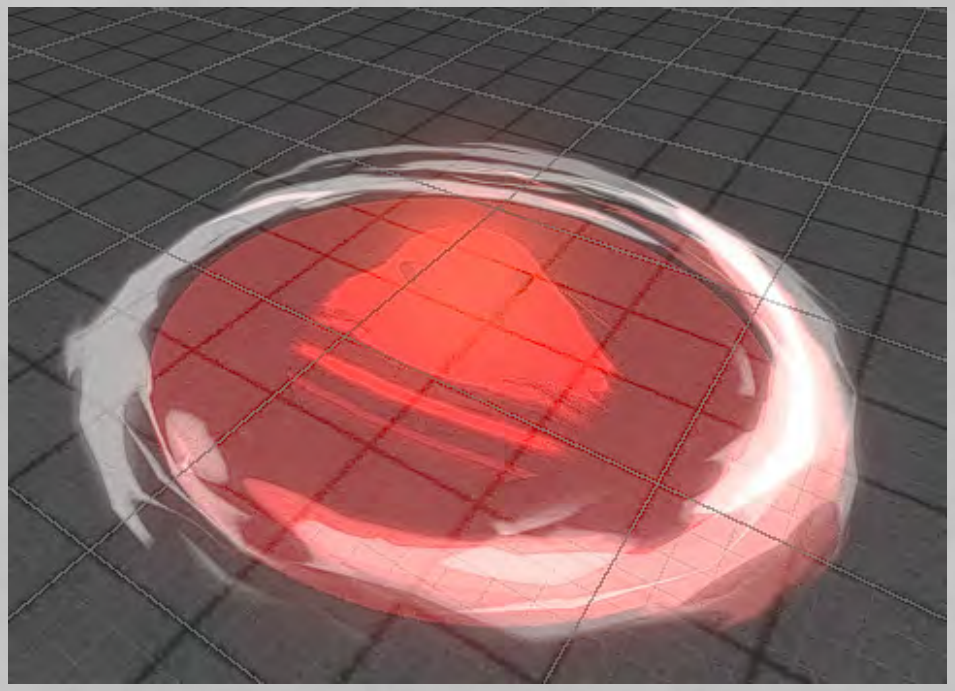


Ainsi il saute et le vortex est créé. Les ennemis vont donc être aspiré au centre de ce dernier



Les ennemis sont aspirés et ce situe en dessous du joueur.

Jump	Jump Height	8	18	25	35
	Max Jump count	1	1	2	3
	Vortex Range	0	30	30	60
	Consumption per jump	20	100	200	450



Visuel du vortex

Feedback du nombre restant de saut

Le Coup de Poing :

Fonctionnement :

Pensé à l'origine comme une mécanique de corps à corps brutale, le coup de poing dans Kill Dem'On s'est rapidement affirmé comme un outil central du gameplay, combinant domination rapprochée, exploitation des faiblesses ennemies, et mobilité offensive.

Lié au point faible :

Cette attaque est particulièrement efficace lorsqu'elle est utilisée sur un point faible : si le joueur déclenche le coup de poing au bon moment et dans la bonne direction, l'ennemi est one-shot et génère une grande quantité d'énergie, créant une récompense forte pour la précision et le timing.

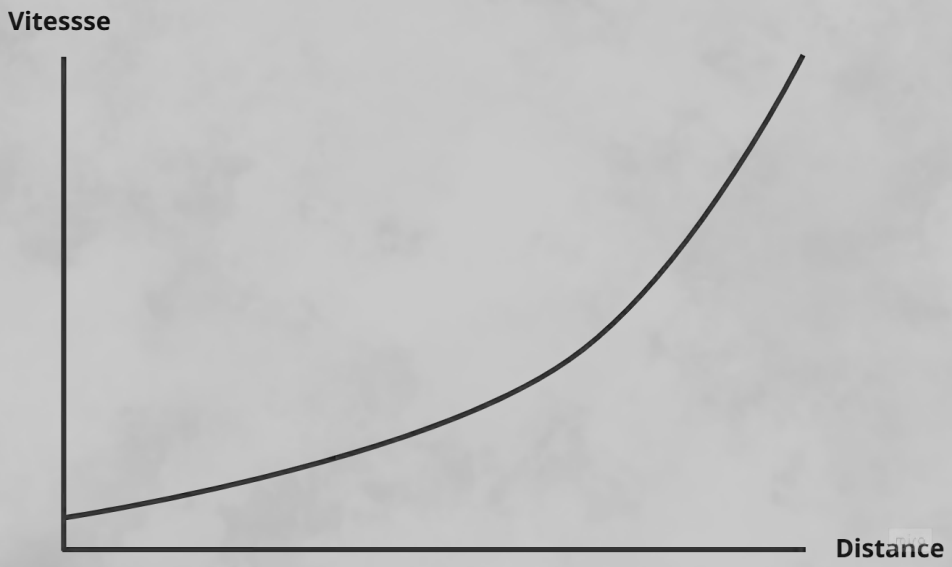
Le Dash :

Mais c'est l'ajout du dash offensif qui a véritablement donné toute son envergure à cette mécanique. Lorsqu'un ennemi est présent dans un cône de détection devant le joueur, et que ce dernier active le coup de poing, un dash automatique est déclenché en direction de la cible la plus proche du centre de la visée. Ce dash est directionnel, accélère progressivement jusqu'à l'impact, et ne possède aucune limite de distance, offrant une grande liberté d'approche, même à travers de vastes espaces.

Une detection en fonction de la distance :

Afin de faciliter l'engagement à distance, le cône de détection a été volontairement élargi : plus l'ennemi est loin, moins la précision requise est stricte. Ce choix permet au joueur de visuellement "verrouiller" une cible sans avoir à être parfaitement aligné, ce qui rend l'usage du dash à longue portée plus intuitif et satisfaisant.

Schéma du cone de détection



La vitesse du dash en fonction de la distance avant de toucher l'énnemi

Un échec qui a des conséquences :

Cependant, si aucune cible n'est détectée dans le cône au moment de l'activation, le dash ne se déclenche pas. Le joueur est alors puni par une animation vide, qui lui fait perdre un temps précieux. Ce risque renforce l'importance de la lecture de l'espace et encourage les joueurs à utiliser cette mécanique avec justesse, pas en spam aveugle.

En combinant attaque puissante, mobilité tactique et feedback de récompense, le coup de poing s'est imposé comme une mécanique centrale de Kill Dem'On, à la fois satisfaisante, expressive, et stratégique.

Caméra Feedback :

Distorsion :

Dès que le Coup de poing est lancé, un effet de distorsion se déclenche : les bords de l'image se courbent doucement vers l'intérieur pour créer un tunnel visuel. La déformation croît de façon continue pendant toute la ruée, puis disparaît dès l'impact. L'objectif est double :

- Accentuer la sensation de vitesse en forçant le regard vers le centre de l'écran.
- Préserver la lisibilité : l'effet reste assez subtil pour éviter fatigue visuelle et motion sickness.,

L'intensité de la distorsion est proportionnelle au temps restant à parcourir (en fonction de la distance) : plus le dash est long, plus la valeur appliquée est élevée ; sur un trajet très court, l'effet est léger.

Camera sans effet

Camera avec effet

Variation du FOV :

En parallèle, la caméra élargit progressivement son champ de vision (FOV) pendant la ruée : on part de la valeur de référence, on l'augmente juste assez pour renforcer l'impression d'accélération, puis on la réduit en douceur tout de suite après le contact, afin que le cadrage redevienne stable avant que le joueur reprenne le contrôle complet.

Comme pour la distorsion, le FOV est pilotée par la durée effective de la ruée : plus le joueur doit couvrir de distance, plus l'écartement maximal du FOV est important, créant une sensation de vitesse soutenue. Sur un dash très court, la variation reste minimale. Shake de caméra,

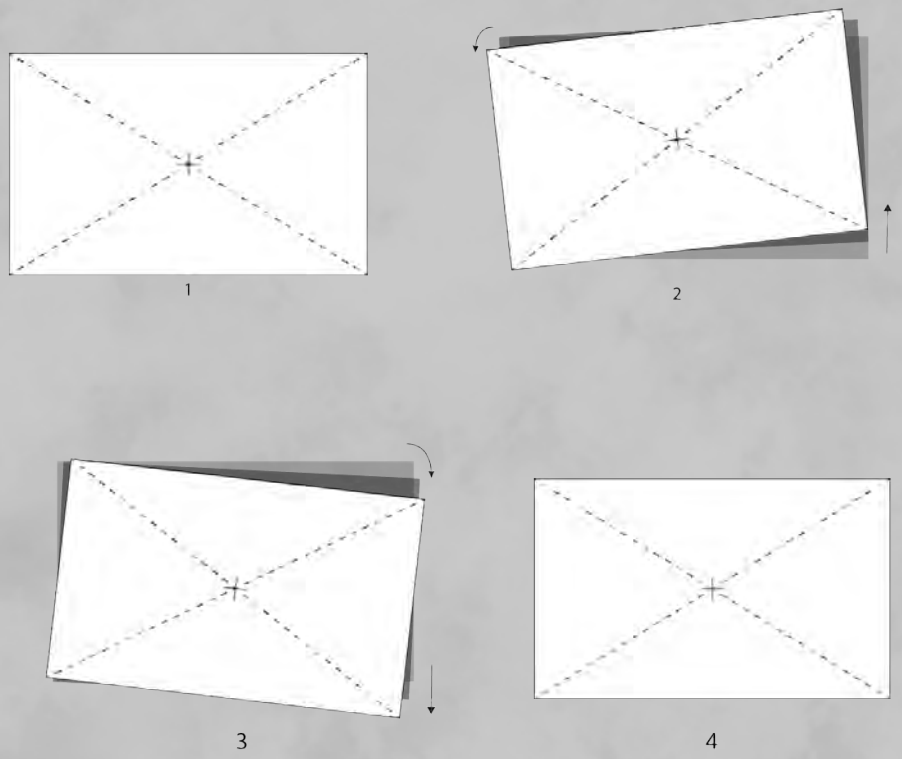


schéma du camera shake quand on hit un ennemi

Caméra Shake :

Au moment où le poing touche l'ennemi, un Cinemachine Impulse est émis pour produire une secousse courte mais énergique : la caméra vibre légèrement en position et en rotation afin de matérialiser l'impact.

Deux profils de shake :

- Coup standard : secousse modérée, rappelant le coup physique sans perturber le suivi de l'action.
- Coup critique (point faible) : même courbe mais gain supérieur, pour amplifier visuellement la réussite et offrir un feedback plus satisfaisant.

Cercle Magique :

Au déclenchement du Coup de poing, un cercle magique luminescent apparaît brièvement autour du poing du joueur ; il est composé de motifs concentriques et de lignes géométriques qui se dessinent en quelques frames, comme s'ils étaient « gravés » dans l'air.
L'animation sert de repère visuel clair :
Phase de formation : Sitôt le clic droit détecté, les tracés s'illuminent de l'extérieur vers l'intérieur, évoquant la « charge » d'une énergie mystique. Cette apparition instantanée signale au joueur que la compétence est bien déclenchée et que la ruée commence.

Résultat de l'action :

- **Touché réussi** : au moment de l'impact, le cercle ne disparaît pas d'un coup ; il se rétracte vers le centre, comme aspiré par l'énergie libérée, puis s'estompe. Ce resserrement donne un sentiment de contact précis et renforce l'idée que la magie s'est transférée dans la cible.
- **Échec** (aucune cible atteinte) : si le poing ne touche rien, le cercle se brise en éclats de lumière ; les segments se fragmentent et s'éteignent rapidement, traduisant la dissipation brutale de l'énergie et l'échec de l'action.

En combinant ces deux fins d'animation, on obtient un feedback immédiat : le joueur identifie en un clin d'œil la réussite ou l'échec de son attaque, sans devoir attendre un autre indice auditif ou chiffré. Visuellement, le cercle magique renforce également le thème arcane du personnage ; il agit comme une signature identifiable, cohérente avec la distorsion, l'ouverture de FOV et le shake de caméra déjà décrits, pour composer un ensemble de retours sensoriels homogène et immersif.

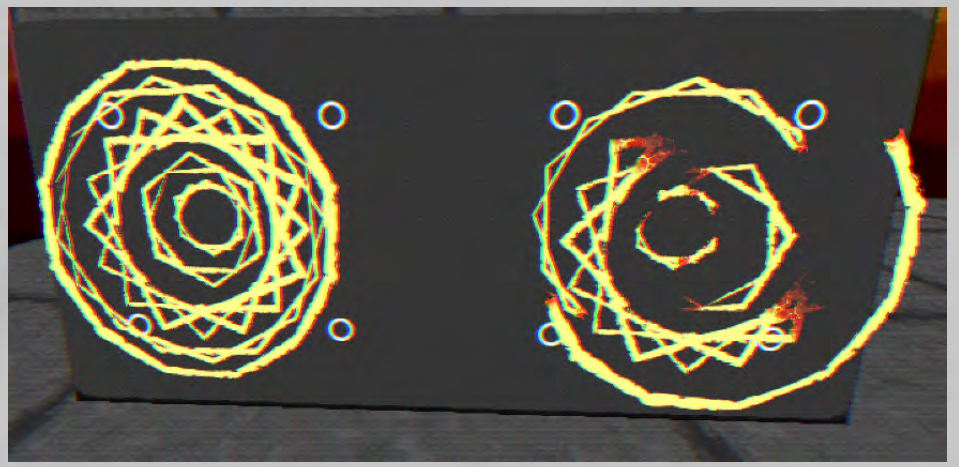


image du cercle du CDP entier à gauche et en cours d'évaporation à droite

Animation du bras :

Lorsqu'un joueur déclenche la compétence, l'animation démarre toujours de la même façon : le coude se replie légèrement pour charger le geste, puis le poing jaillit vers l'avant dans un mouvement explosif. C'est la phase qui suit cette frappe initiale qui change selon le résultat de l'action.

Si le coup réussit, c'est-à-dire qu'une cible est captée et que le dash s'enclenche, l'avant-bras prolonge naturellement son élan : il reste tendu, aligné dans l'axe de la ruée, et ne fléchit plus jusqu'au contact. Le poignet demeure verrouillé, l'épaule avance juste ce qu'il faut pour accompagner l'ensemble, donnant l'impression d'un membre transformé en « lance » qui guide le personnage vers l'ennemi. Tant que ce bras demeure tendu, le joueur sait d'un coup d'œil que la translation est en cours ; l'impact et le retour caméra se produisent au moment précis où la main atteint la cible, puis l'animation bascule vers la réaction d'ennemi ou la reprise de contrôle normale.

Si le coup échoue, aucune cible n'étant détectée, l'extension initiale est immédiatement suivie d'un repli énergique : le coude revient vers le buste, la main se replace devant la poitrine et le personnage **retrouve sa garde.** Ce retour rapide ferme visuellement le geste, marque l'échec, et remet le joueur en posture neutre sans délai superflu, afin qu'il puisse enchaîner aussitôt une autre action.

La distinction entre bras tendu et bras replié s'accorde avec les autres retours visuels : cercle magique qui se resserre ou se brise, variation de FOV et distorsion plus ou moins prononcées, secousse de caméra présente ou atténuée. Tout l'ensemble contribue à informer le joueur, en une fraction de seconde, de la réussite ou de l'échec de son Coup de poing.

Image de l'animation du dash du coup de poing

Sprint :

Fonctionnement :

Lorsque le joueur maintient la touche de sprint :

Accélération progressive

La vitesse croît pendant x secondes selon une courbe douce (ease-in) jusqu'à atteindre la valeur « Vitesse de Sprint ».

Vitesse maximale variable

Cette vitesse de pointe dépend du palier du personnage ; plus le palier est élevé, plus la vitesse finale (et l'énergie cinétique associée) est importante, ce qui augmentera les dégâts infligés lors d'un impact.

Maintien et gestion d'endurance

Tant que la touche reste enfoncée et que les conditions d'endurance sont remplies, la vitesse reste verrouillée à son plafond.

Décélération progressive

À la relâche de la touche (ou quand l'endurance est épuisée), la vitesse décroît selon la même courbe, ramenant le personnage à sa vitesse de déplacement normale sans brusquer le contrôle.

Cette montée/descente graduelle garantit un ressenti fluide : le sprint s'amorce naturellement, conserve la réactivité du gameplay et offre un feedback clair sur la montée en puissance du personnage selon son palier.

Sphère d'impact en sprint

L'accèlération du sprint en fonction du temps

Dégâts au contact :

Pendant toute la durée du sprint, une **sphère de collision invisible** (hitbox) entoure le joueur.
Tout ennemi pénétrant dans la sphère subit immédiatement des dégâts. Ces dégâts sont multipliés par un coefficient lié au palier ; un palier supérieur signifie un coup plus brutal.

Effet de knockback (recul / projection) :

En plus des dégâts, la cible est repoussée violemment dans la direction du déplacement. L'intensité de ce recul s'échelonne elle aussi avec le palier :

Palier bas : poussée modérée, l'ennemi perd légèrement l'équilibre.
Palier élevé : projection nette, l'ennemi est éjecté sur une plus longue distance, ouvrant une fenêtre plus large pour le joueur.

Synchronisation
L'application des dégâts et du knockback est instantanée, exactement au frame où la sphère entre en collision avec la hitbox ennemie, afin de maintenir la sensation d'impact direct et de cohérence avec la vitesse atteinte.

Ce système rend le sprint offensif : plus le joueur progresse dans les paliers, plus sa charge devient dévastatrice, transformant le déplacement rapide en véritable arme de zone.

Schéma du sprint ou le joueur commence à sprinter

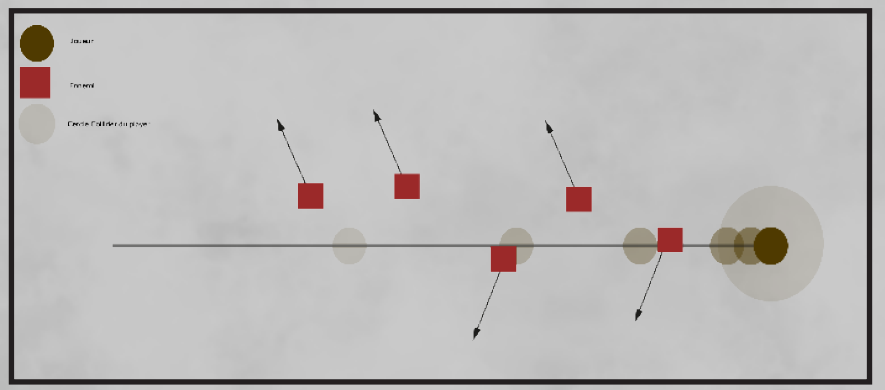


Schéma du sprint, le joueur c'est déplacé et les cubys sont repoussés

Schéma du sprint, les cubys se sont déplacées

Camera sans effet

Camera avec effet

Caméra Feedback :

Ouverture du FOV :

Le champ de vision s'élargit pendant l'accélération. L'amplitude exacte dépend de la vitesse maximale autorisée par le palier : plus le palier est élevé, plus l'ouverture devient large. Lorsque le joueur relâche la touche, le FOV se referme avec la même douceur, garantissant une transition sans à-coups.

Distorsion radiale :

En parallèle, une distorsion apparaît sur les bords de l'écran. Elle suit la même courbe que le FOV : à un palier bas, la déformation reste peu imperceptible ; à un palier haut, elle se renforce pour créer un effet de tunnel et accentuer la vitesse, tout en restant assez subtile pour ne pas gêner la lisibilité.

Camera Dutch :

Cet effet n'apparaît qu'au moment où le joueur percute un ennemi. La caméra bascule brièvement d'un côté puis se redresse aussitôt. L'angle reste dans une plage modérée, indépendante du palier ; pour éviter la monotonie, la bascule alterne aléatoirement entre la droite et la gauche. Chaque impact génère ainsi une inclinaison subtilement différente, soulignant le choc et le knock-back sans devenir prévisible ni fatiguer le joueur.

Combinaison :

En combinant ces trois paramètres — ouverture de FOV, distorsion radiale et dutch ponctuel — le sprint transmet clairement la montée en puissance : plus le personnage progresse dans les paliers, plus la caméra s'éloigne, se déforme et bascule, offrant au joueur un ressenti de vitesse et d'impact toujours plus intense, mais sans sacrifier le confort ni la lisibilité.

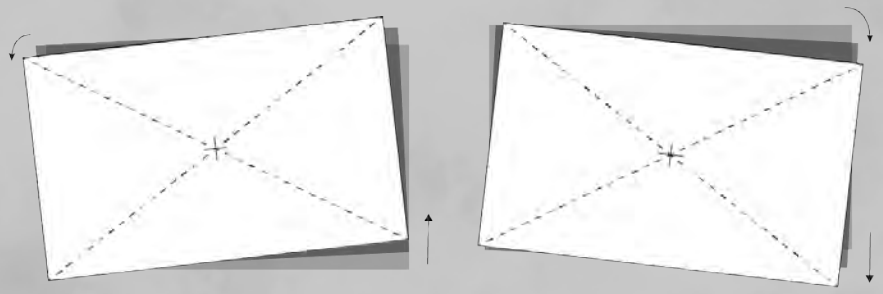


Schéma du mouvement de la caméra du dutch

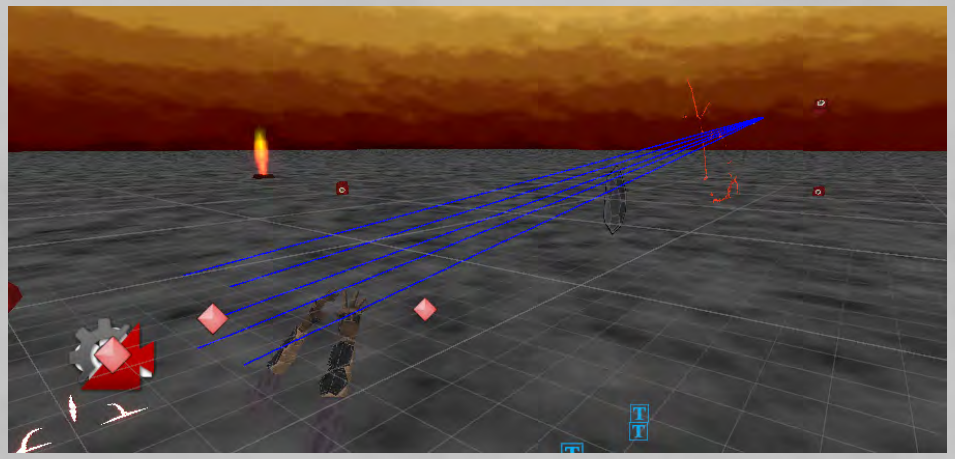
Sprint	Sprint Speed	0	45	60	80
	Consumption per sec	0	250	800	4500
	Sprint Damage per sec	0	65	300	2000
	Movable Time at max energy (sec)	0	60	63	22
	Drop bonus	0	0	0	-10
	Movable Time at 10% energy (sec)	0	6	6	2

Le Tir :

Fonctionnement :

Hitscan :

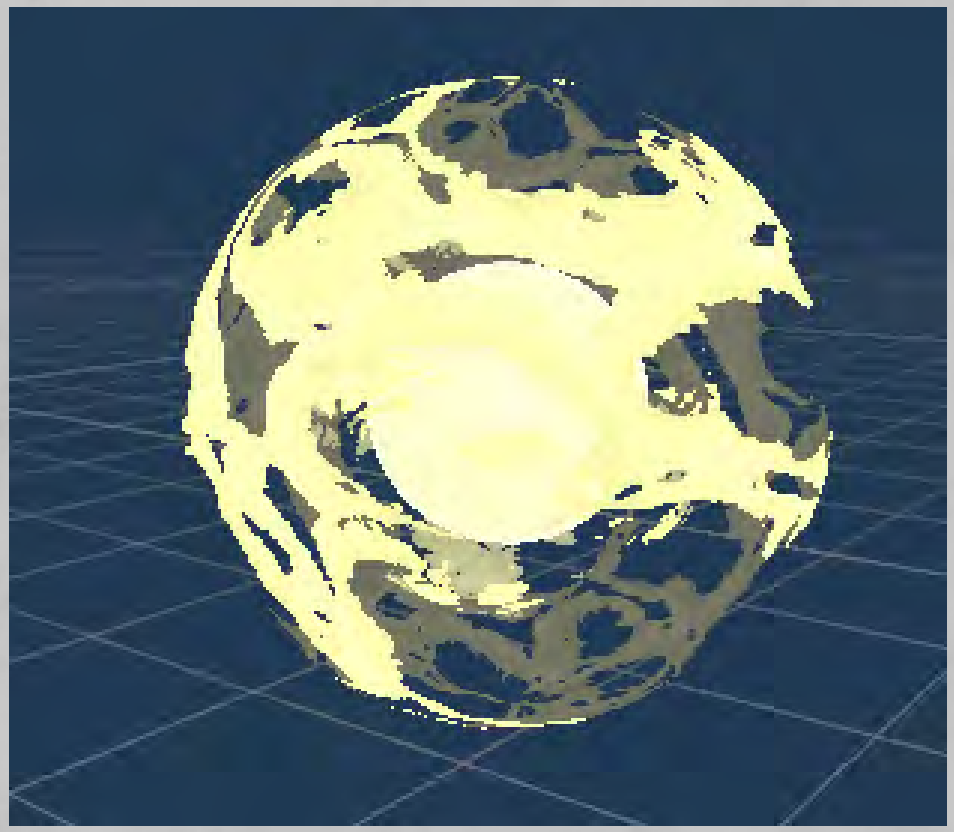
Le tir repose sur un système hitscan : au déclenchement, un rayon invisible part instantanément de la paume du joueur vers la direction visée et détermine l'impact à la frame. Pour élargir légèrement la tolérance et rendre la visée moins punitive, ce rayon est entouré d'un cylindre de collision ; toute cible entrant dans ce volume est considérée comme touchée, même si le rayon strict passe à côté.



Screen représentant la hitbox du hitscan du tir en bleu

Projectiles :

Bien que le calcul soit immédiat, de petits projectiles visuels sont générés le long de la trajectoire. Ils n'interviennent pas dans le gameplay ; ils servent uniquement de support visuel et sonore, afin que chaque tir donne un retour clair de départ et de trajectoire.



Screen du projectile

Lié au Palier :
La mécanique évolue avec le palier du personnage : plus le palier est élevé, plus la cadence de feu augmente et moins chaque tir consomme d'énergie, dégâts élevé. Les paliers inférieurs conservent l'effet inverse (cadence plus lente, dégâts plus faible et coût augmenté).

CrossHair :

État 1 : Idle
Affiché en permanence quand aucune cible valide n'est dans le cylindre de hitscan.
Écartement par défaut : pointes latérales à distance fixe du centre, segments verticaux maximum.

État 2 : Target Lock
Condition : la hit-box d'un ennemi entre dans le cylindre d'assistance.
Action UI :
Les deux lames se rapprochent d'un cran (≈ 10 % du rayon visuel).
Le segment vertical raccourcit à l'identique.
Fonction : confirme que le tir hitscan touchera si le joueur appuie maintenant.

État 3 : Hit Confirm
Condition : le raycast touche une cible.
Action UI :
Apparition instantanée d'un X rouge au centre, calé sur une frame (fade-out très bref ≤ 0,15 s – libre à l'UI de régler).

Screen du crosshair en mode idle

Screen du crosshair quand on regarde un ennemi

Screen du crosshair quand on hit un ennemi

Shoot	Fire rate	4	6	10	20
	Damage	25	45	85	200
	WeakPoint Multiplier	0.3	0.2	0.2	2
	Dps	100	270	850	4000
	Dps with weak point	30	54	170	8000
	~~Hits to Expose Weakpoint~~	~~-1~~	~~0~~	~~1~~	~~3~~
	Time to expose WeakPoint	-0.28	-0.02	0.08	0.11
	Hits to Kill	2	2	3	3
	Time to Kill	0.40	0.30	0.24	0.13
	Time to kill with WeakPoint	1.33	1.48	1.18	0.06
	Consumption per shoot	2.5	15	80	200
	Consumption per sec	10	90	800	4000
	Shoot Time (Max Energy)	300	167	63	25
	Shoot Time (10% Energy)	30	17	6	3
	Drop bonus	5	0	10	-5

Animation :
L'attaque est déclenchée depuis la paume de la main droite :
Le bras se tend vers l'avant, la main s'ouvre légèrement pour libérer le tir.

À chaque salve, le membre subit un recul discret synchronisé avec le claquement sonore, avant de revenir aussitôt en position prête.

Cette boucle tendue -> recul -> retour s'adapte automatiquement à la cadence : à haut palier le mouvement est plus rapide, conservant la cohérence avec le rythme de feu augmenté.

Image de l'animation du tir

Pillonage :

Fonctionnement :

Activation, conditions et déroulement :

Pour déclencher le Pilonnage, trois pré-requis sont obligatoires :
- Palier minimum 3 : assure que le joueur possède déjà l'énergie et la maîtrise nécessaires.
- Position aérienne : l'input doit être lancé en vol (saut, rebond ou propulsion).
- Angle d'attaque ≤ 30 ° par rapport au sol.

Une fois validés, la trajectoire verticale s'enclenche : le personnage accélère brutalement, poing en avant, jusqu'au point d'impact.

schéma de l'angle d'activation du pillonage

Onde de choc, génération et propagation :

Au contact, un front sphérique d'énergie est libéré ; il se dilate, balaie la surface autour du point d'impact, puis se dissipe. Tout ennemi touché encaisse des dégâts instantanés, directement liés à l'ampleur de l'onde.

Variation de taille selon la hauteur :

Le rayon maximal de l'onde est calculé à partir de la hauteur de la plongée : plus la descente est haute, plus la sphère s'étend loin et plus les dégâts sont sévères. À basse altitude, l'effet reste contenu ; à grande hauteur, l'onde peut neutraliser des groupes entiers de cibles faibles et amputer lourdement la santé des ennemis robustes.

Bounceback :

Au même instant, un recul vertical propulse le joueur vers le haut ; la force de ce rebond dépend, elle aussi, de la vitesse d'impact. Les adversaires pris dans l'onde sont projetés en l'air, ouvrant une fenêtre de combo ou de repositionnement. Ce rebond remet le joueur en hauteur, lui offrant une vue claire sur la déflagration et le vfx qu'il vient de provoquer. Challenge technique et artistique,

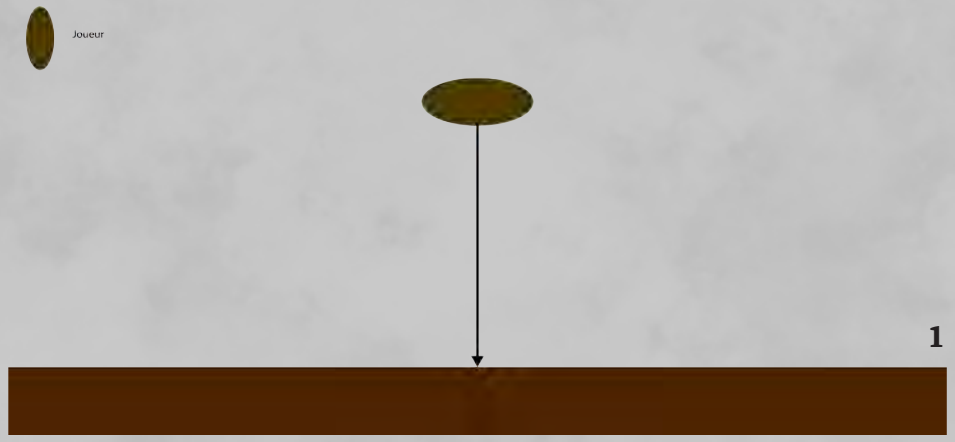
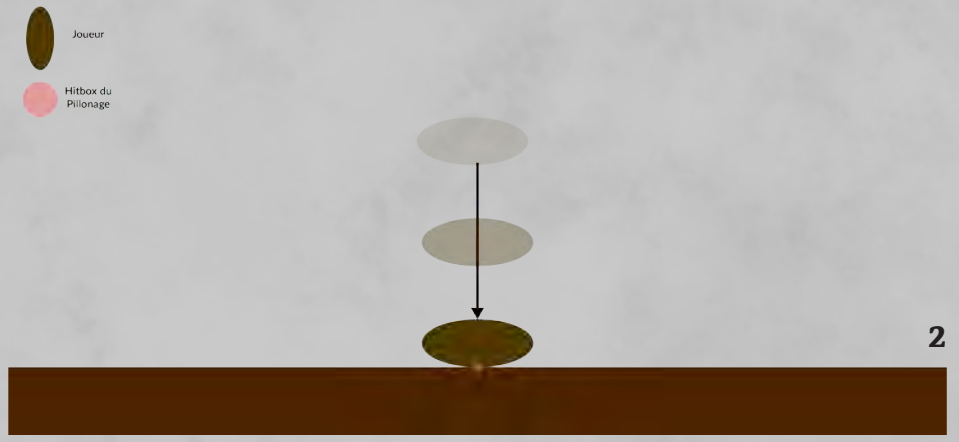
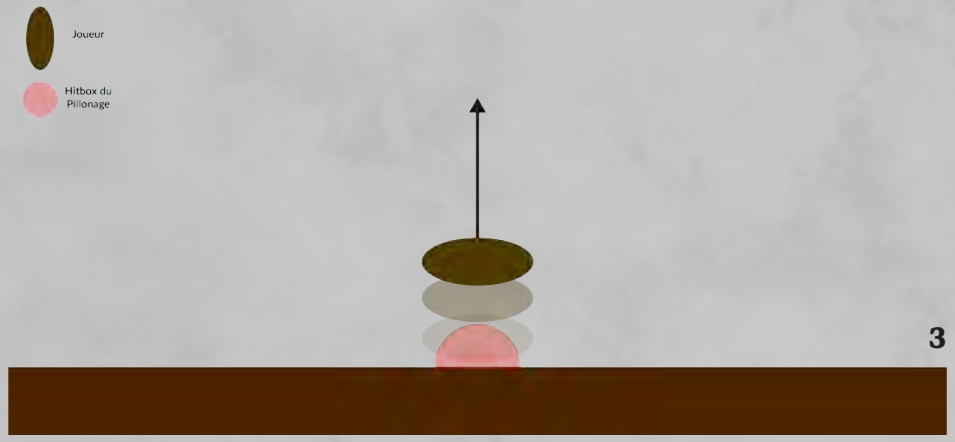


schéma montrant l'activation et le fonctionement du pillonage/le joueur active sa compétence

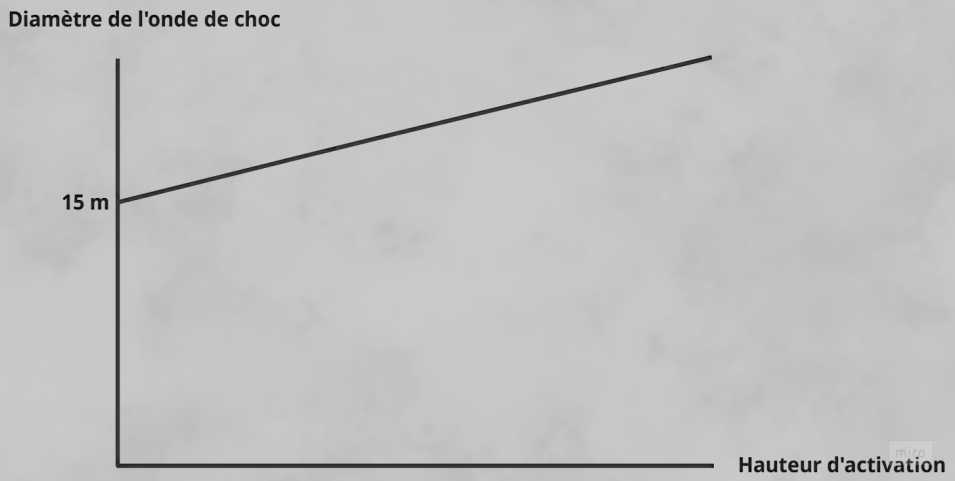


le joueur touche le sol et la zone du pillonage spawn



le joueur est bounce back et la zone du pillonage augmente

le joueur a atteint sa hauteur du bounce back et la zone du pillonage c'est totalement créé



Courbe du diamètre de l'onde de choc en fonction de la hauteur d'activation du pillonage

Challenge de créer un Pillonage en FPS :

Concevoir un VFX de pilonnage en vue FPS s'est révélé atypique : aucun shooter ne proposait jusque-là un slam de cette ampleur, visible depuis l'intérieur de l'action. Le principal défi a été de combiner :
- une onde de choc volumétrique lisible sans masquer la cible,
- une échelle dynamique variant en temps réel avec la hauteur,
- un rebond synchronisé qui remet le joueur au bon endroit pour admirer l'effet.

Cet empilement de contraintes visuelles, techniques et ergonomiques fait du pilonnage l'un des VFX les plus complexes du projet.

Référence Pillonage :

screen d'Ultrakill lors d'un pilonnage

UltraKill :

- vue FPS
- trait lumineux vers le bas suivant la descente
- Shake quand on est sur le sol
- particule quand on touche le sol
- rebond du joueur

Borderlands 3 :

- Prise de FOV dans la descente
- Animation de la main
- Arrivé au sol on saute un peu
- La caméra se remet à la base
- Shake de la caméra à l'arrivé
- Flou de l'image sur les côtés
- Fx de zone plate quand on arrive au sol
    - Sur le point d'impact (étincelle)
    - Sur la zone d'impact (flat)

screen de Borderlands 3 lors d'un pilonnage

screen de Halo Infinite lors d'un pilonnage

Halo Infinite :

- Passage troisième personne
- Caméra Shake before and after
- TimeSlow and Stuck before
- No GroundFeedback

EchoPoint Nova :

- au sol halo bleu qui se déploit autour du joueur
- écran flou

screen d'EchoPoint Nova lors d'un pilonnage

Feedback Pillonage :

Lors de la conception des feedbacks de notre pillonage, nous les avons divisés en trois parties. Cela nous a permis de mieux les différencier et d'avoir des feedbacks précis et utiles pour chacune de ces étapes.

Activation :

La première étape de notre pillonage est l'activation. Cette dernière s'effectue lors de l'input du joueur. Nous avons uniquement utilisé une animation afin de simuler une prise de recul, car il s'agit de l'effet le plus court.

L'animation :

Le joueur effectue une animation où le poing revient vers lui afin de simuler un effet de recul. Elle se distingue de l'animation du coup de poing, qui est plus directe.

Image de l'animation de départ du ground pound

Fall :

Le fall est la partie où le joueur descend à grande vitesse vers le sol. Le joueur doit ressentir une véritable sensation de vitesse, comme s'il tombait le plus vite possible, tout en voyant qu'il ne se téléporte pas au sol. Pour cela, nous avons utilisé de nombreux feedbacks.

Dutch/Tremblement :

Pour simuler un tremblement et une prise de vitesse, comme si le joueur entrait dans l'atmosphère, nous avons ajouté un effet de dutch/tremblement. Ce dernier fait pivoter l'écran de manière aléatoire de droite à gauche, de façon répétée jusqu'à l'arrivée au sol. Il s'accélère progressivement en fonction du temps de descente.

Effet sur la caméra :

Afin de renforcer la sensation de vitesse, nous utilisons une variation croissante du FOV et un effet de distorsion sur la caméra. Cette variation est continue : plus on pillonne de haut, plus l'effet devient visible — avec une limite définie pour éviter tout bug.

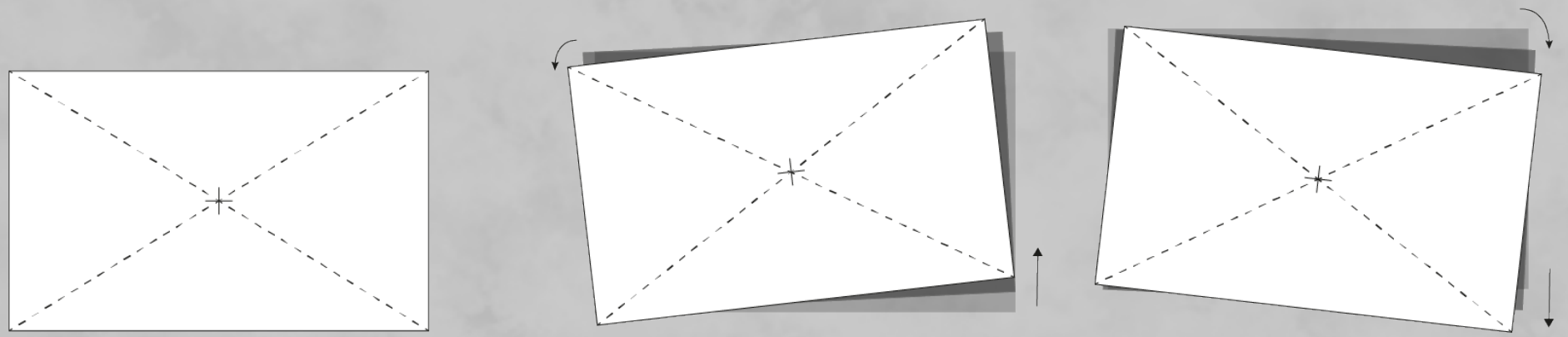


Schéma du dutch/tremblement de la caméra.

Camera sans effet

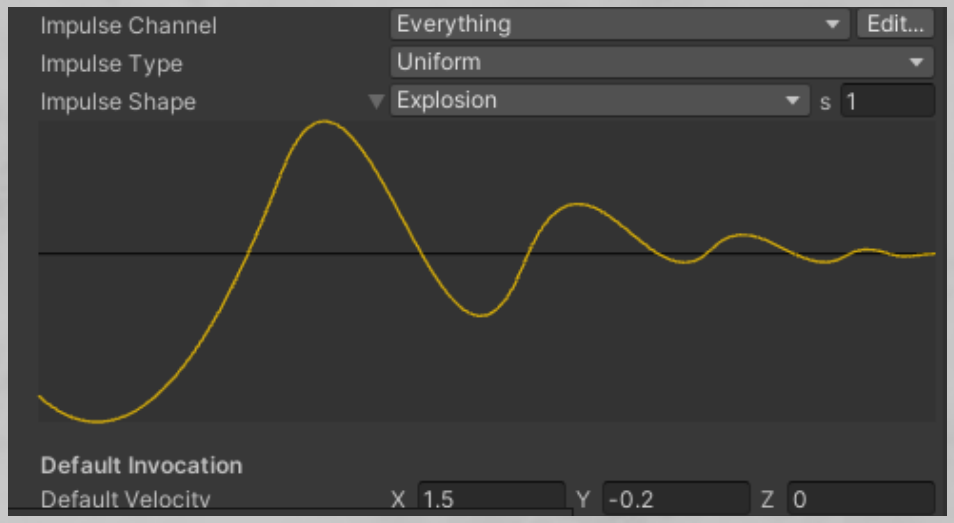
Camera avec effet

Explosion :

La dernière étape du pillonage est l'explosion, qui survient lorsque le joueur touche le sol. Les feedbacks sont ici très importants pour montrer que l'explosion se propage et pour renforcer la sensation de puissance de l'impact.

Camera Shake :

Pour accentuer l'impact, nous avons ajouté un effet d'impulsion d'explosion (Cinemachine Impulse) sur la caméra. Cet effet simule une explosion lointaine et correspond parfaitement à ce que nous voulions.



Forme graphique de l'effet d'impulse sur la caméra

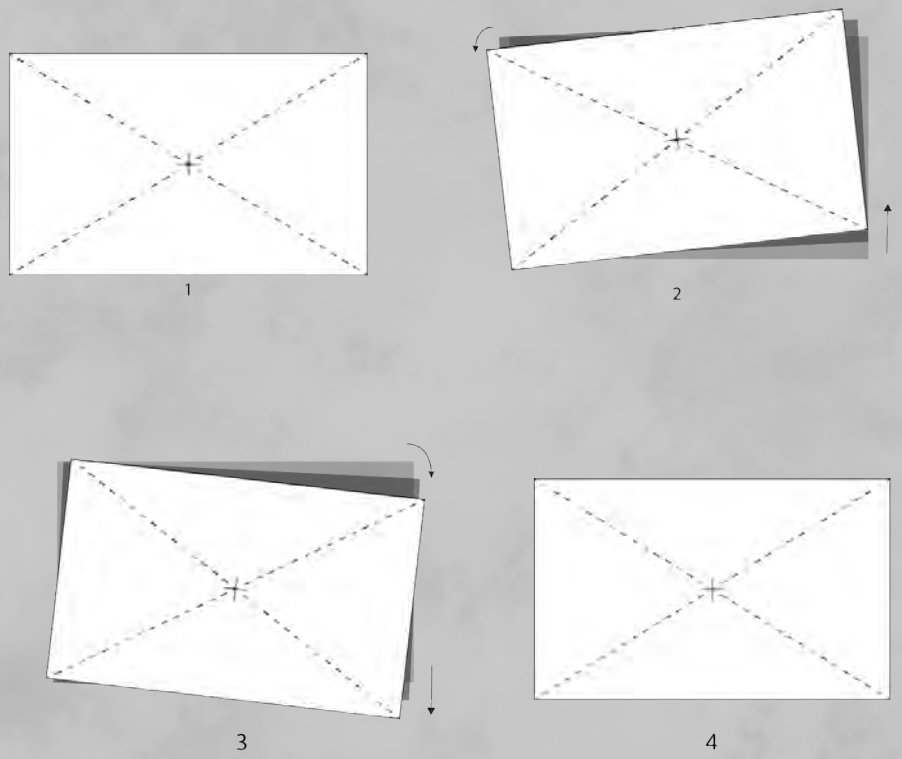


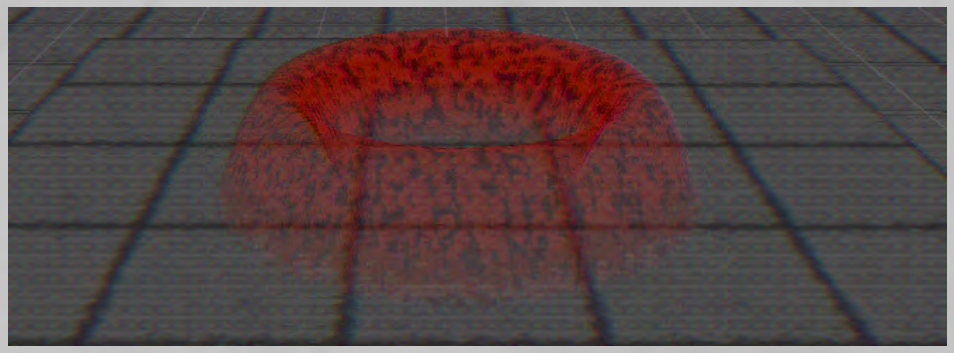
schéma du camera shake quand on touche le sol

Bounce Back :

Le bounce back du joueur n'est pas directement « feedbacké », mais il est ressenti par la montée dans les airs après l'impact.

ShockWave :

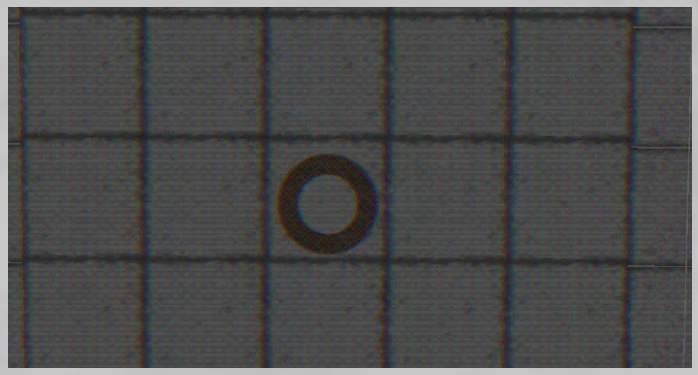
À l'arrivée au sol, une shockwave apparaît d'abord en petite taille, puis se déploie avec l'explosion. Elle renforce l'effet de puissance et sert aussi de repère visuel pour indiquer au joueur quand il touchera les ennemis.



VFX de la ShockWave

Onde :

Pour ajouter encore plus de puissance au pillonage, nous avons intégré un effet d'onde, qui simule un souffle d'explosion (comme avec les bombes atomiques). Elle se propage puis disparaît en fondu.



VFX de l'onde

Animation :

Enfin, l'animation joue un rôle essentiel dans le game feel de l'explosion, car elle initie l'attaque. Le bras du joueur revient puis frappe le sol, ce qui renforce la sensation de puissance.

Ground Pound	Max sphere range	0	0	100	500
	Damage	0	0	150	24000
	Consumption	0	0	2000	9000
	Usage Count	0	0	18	6
	Drop bonus	0	0	-10	-7
	Enemy In zone	0	0	10	20

Image de l'animation de l'explosion du pillonage

Systèmes

Objectifs :

Un objectif simple :
L'objectif principal de Kill Dem'On est volontairement simple : éliminer 1000 ennemis, puis affronter un boss final. Cette structure directe permet au joueur de comprendre immédiatement ce qu'il doit accomplir, sans être parasité par des sous-objectifs complexes. Une fois le boss vaincu, le joueur accède à l'écran du Leaderboard, où il peut entrer son nom et visualiser sa performance, en pleine conscience de ce qu'il a accompli.

Une efficacité forcé :
Ce système d'objectif pousse le joueur à être efficace. Pour atteindre les 1000 kills, il ne suffit pas de survivre : il faut optimiser chaque action, gérer intelligemment son énergie, et maintenir une pression constante sur les ennemis.
Plus particulièrement, l'enjeu devient de rester le plus longtemps possible au palier 4, le niveau maximal de puissance. Ce palier garantit une vitesse, une agressivité et une létalité accrues, mais demande au joueur de garder un équilibre énergétique positif pour ne pas redescendre.
Cela transforme la phase principale du jeu en une course à la performance, où l'endurance tactique et la maîtrise des mécaniques font toute la différence.

Boss Final :
Une fois les 1000 ennemis vaincus, le joueur affronte un boss final. Ce combat vient bouleverser les habitudes acquises jusque-là : il ne s'agit plus d'éliminer rapidement des groupes d'ennemis, mais de concentrer ses ressources sur un seul adversaire, résistant, dangereux, et au comportement unique.
L'objectif ici n'est plus l'optimisation de l'énergie dans la durée, mais l'utilisation maximale des outils offensifs pour infliger un maximum de dégâts. Ce changement de rythme marque une clôture forte de l'expérience, un pic de tension et un moment de concentration total avant le relâchement final.

Rejouabilité/Leaderboard :
L'apparition du Leaderboard à la fin de la partie permet de valoriser les efforts du joueur et d'inscrire sa performance dans une logique de compétition et de progression personnelle.
Il sait combien de temps il a tenu, à quel rythme il a éliminé ses ennemis, et peut comparer son efficacité à celle des autres joueurs.
Cette boucle encourage naturellement la rejouabilité : une nouvelle tentative est synonyme de nouvel objectif à battre, de score à améliorer, et de style de jeu à affiner.

Combo systeme :

Fonctionnement général :

Le système de combo est un multiplicateur de récompenses qui s'applique lorsque le joueur élimine plusieurs ennemis consécutivement dans un temps imparti. Il vient enrichir la boucle de jeu en récompensant les enchaînements rapides et constants, tout en créant une tension supplémentaire liée à la vitesse d'exécution.

Deux éléments sont directement influencés par le combo :
- L'énergie récupérée :
Lorsqu'un ennemi meurt, il lâche une orbe d'énergie.
Si le joueur n'est pas en combo, il récupère la valeur de base (ex : 10).
S'il est en combo (ex : x2), l'orbe sera multipliée par ce combo : 10 × 2 = 20.
Le joueur récupére donc une orbe de 20 d'énergie.
- Le score :
Chaque élimination accorde 1 point hors combo.
Lorsqu'un joueur est en combo, le score est multiplié (ex : 1 kill × combo x1.6 = 1.6 points).

Déroulement du combo :

1. Un premier kill déclenche le début du combo.
2. Un timer propre au palier actuel démarre (ex : 3.5s au Palier 1).
3. Si une nouvelle élimination a lieu avant la fin du timer, le combo augmente.
4. Si aucun kill n'est effectué dans le temps imparti, le combo est réinitialisé à x1.0.

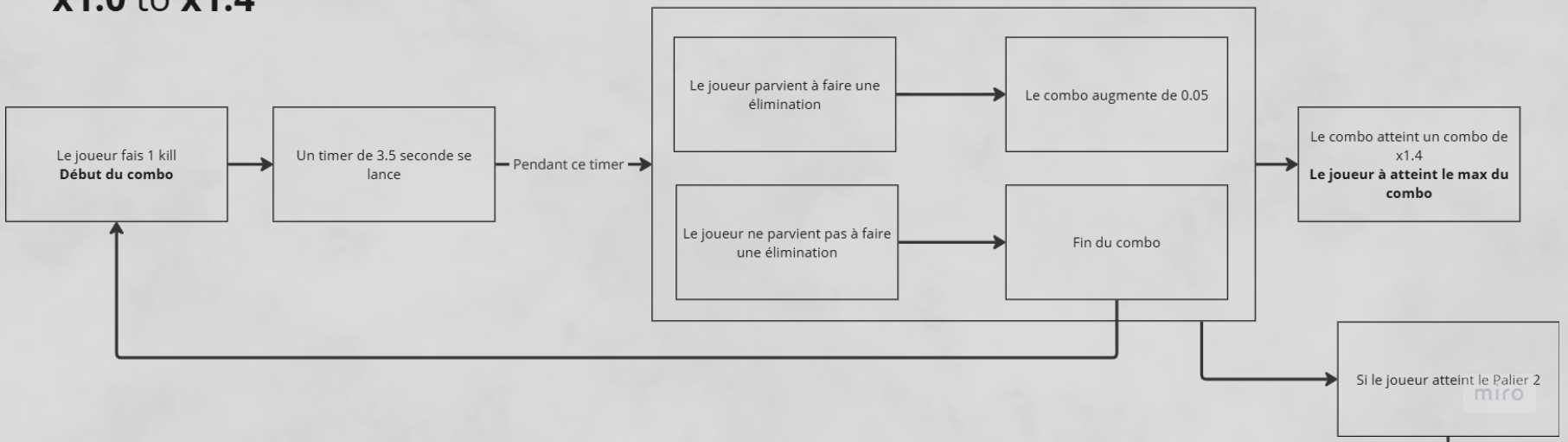
Le système récompense l'agressivité maîtrisée : tuer vite pour monter en combo et rester précis pour ne pas le perdre.

Progression par palier :

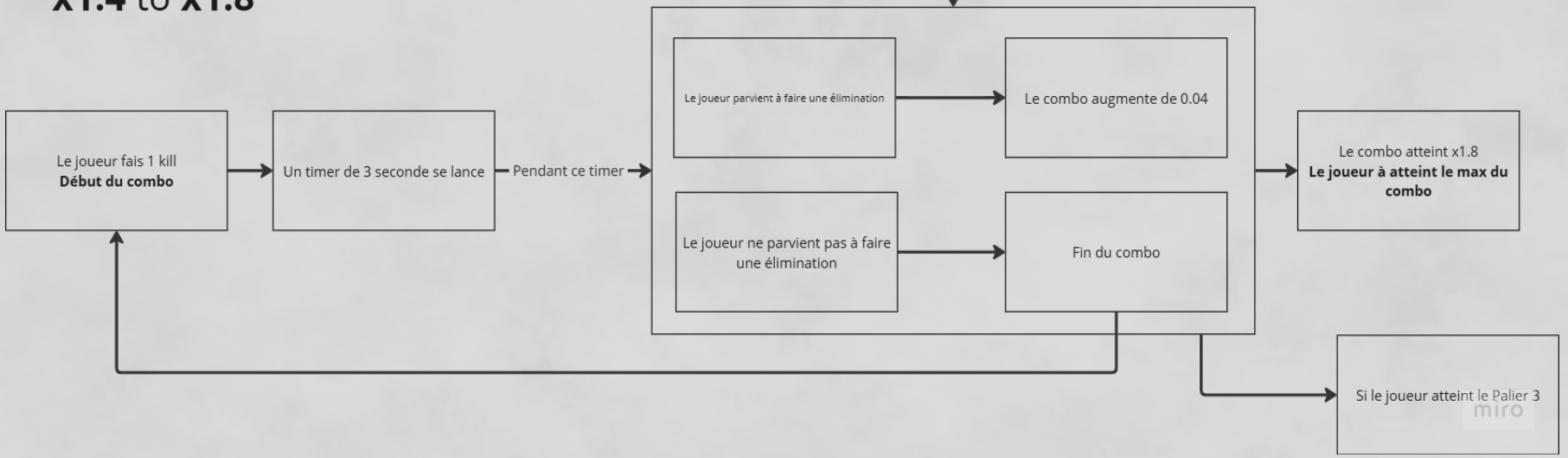
Le système de combo est volontairement verrouillé par paliers : un joueur ne peut pas dépasser certaines valeurs de multiplicateur tant qu'il n'a pas atteint le palier correspondant.

Par exemple, un joueur au Palier 1 ne pourra jamais dépasser un combo de x1.4, même s'il enchaîne 200 éliminations d'affilée. Pour continuer à faire progresser son combo, il devra impérativement atteindre le Palier 2, puis le Palier 3, etc.

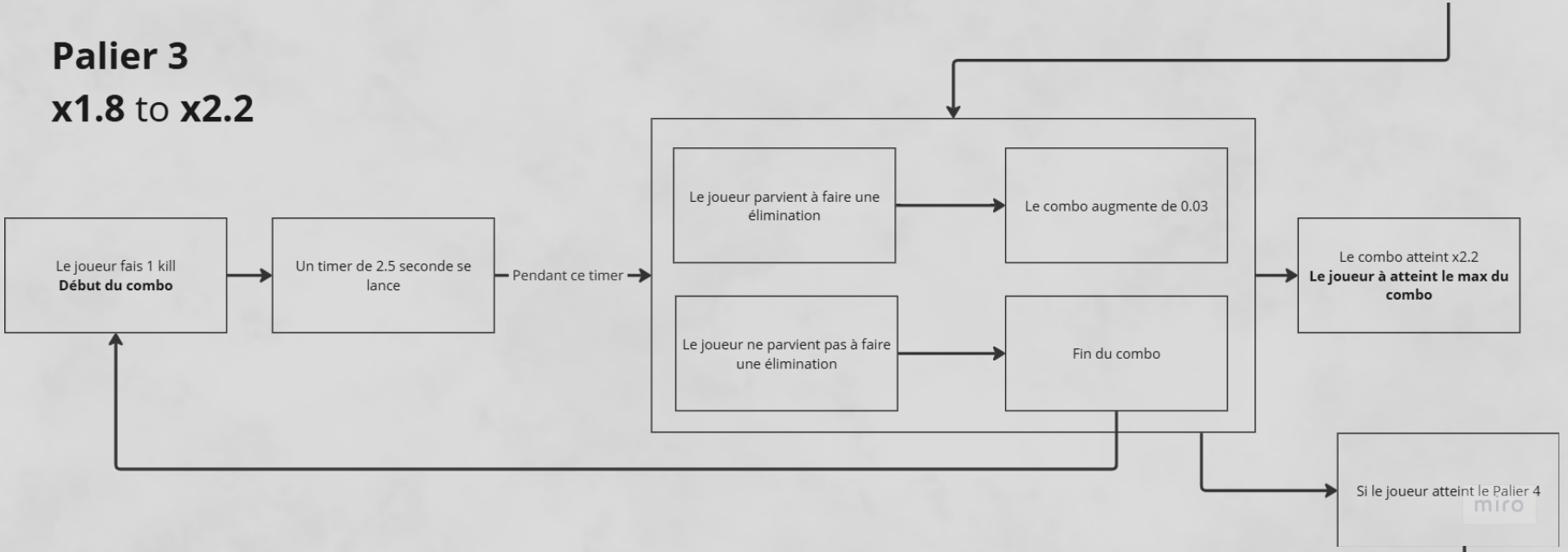
Palier 1
x1.0 to **x1.4**



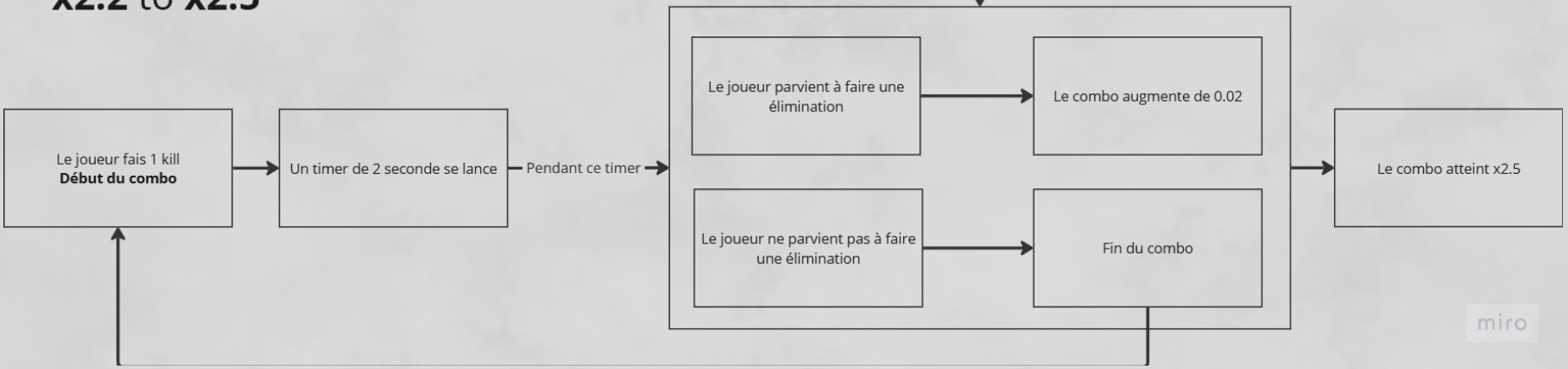
Palier 2
x1.4 to **x1.8**



Palier 3
x1.8 to **x2.2**



Palier 4
x2.2 to **x2.5**



Ce choix de design répond à plusieurs objectifs fondamentaux :
- Éviter une montée en puissance trop rapide

Sans cap, le joueur pourrait atteindre un multiplicateur élevé dès les premières minutes, rendant le système :
- trop généreux, trop tôt
- déséquilibré face à la difficulté
- limitant la montée en tension prévue par le gameplay
Le cap impose une progression contrôlée où chaque palier devient une étape vers une plus grande efficacité.

Structurer la progression du joueur :
En conditionnant l'évolution du combo à celle du palier, le système crée une dynamique claire et motivante :
- Le joueur comprend que ses performances (kills, enchaînements, timing) servent à atteindre des paliers
- Chaque palier débloque de nouvelles opportunités de gains (combo, score, énergie)
- Renforcer la difficulté et la tension à chaque niveau

Plus le joueur progresse :
- plus la fenêtre pour maintenir un combo se réduit
- plus le gain par kill diminue
- plus l'intensité d'exécution augmente

Cette évolution rend les paliers supérieurs plus techniques, exigeants, et gratifiants à maîtriser, tout en conservant une pression constante.
- Valoriser la maîtrise du jeu
Ce système empêche qu'un combo élevé ne soit atteint par chance.
Il récompense :
- la compréhension du système
- la capacité à jouer vite et efficacement
- la précision dans l'enchaînement des éliminations

Chaque kill devient une action intentionnelle et non un simple enchaînement opportuniste.
Créer des milestones motivants

Chaque palier est une étape gratifiante :
- Il permet au joueur de gagner plus
- Il lui donne accès à de nouveaux multiplicateurs
- Il renforce le sentiment de montée en puissance maîtrisée

Image du rendu en UI du combo système

Point faible :

Fonctionnement général :

Certains ennemis disposent d'un point faible : une zone vulnérable temporairement exposée, qui peut être exploitée pour accélérer leur élimination et maximiser les récompenses. Ce point faible n'est pas toujours visible : il n'apparaît qu'après avoir infligé suffisamment de dégâts à l'ennemi.
C'est un signal de vulnérabilité, indiquant que l'ennemi est affaibli et peut être achevé plus efficacement.

Effets du point faible :

Si le joueur touche le point faible avec une attaque quelconque (ex : tir) :
- L'ennemi subit des dégâts supplémentaires,
- Ce qui permet de le tuer plus rapidement,
- Et donc de maintenir le rythme du combo ou de réduire l'exposition au danger

Le point faible devient une opportunité d'efficacité, à repérer, viser et exploiter.

Mais si le joueur touche le point faible avec le coup de poing uniquement :
L'ennemi meurt comme prévu, mais l'orbe d'énergie relâchée est amplifiée :
- le joueur gagne plus d'énergie que s'il l'avait simplement frappé n'importe où
Cette mécanique incite à prendre un risque supplémentaire : se rapprocher d'un ennemi encore actif, viser une petite zone exposée, et l'atteindre avec une attaque de courte portée.

Pourquoi cette mécanique ? :

1. Valoriser la précision et la prise de risque
- Le joueur peut tuer facilement à distance
- Mais en s'approchant et en frappant précisément un point faible, il gagne plus d'énergie

Plus de risque = plus de reward

On renforce la maîtrise du joueur au lieu de la sécurité

2. Ajouter de la stratégie dans le combat

- Le point faible apparaît en réponse à l'état de l'ennemi. Le joueur doit observer, comprendre, puis agir.
- Cela enrichit l'interaction avec l'ennemi, c'est un système réactif qui ouvre des fenêtres d'opportunité à exploiter avec précision

3. Favoriser les cycles combat rapproché / combat à distance

- Les joueurs orientés tir peuvent affaiblir un ennemi à distance
- Puis décider de switcher en poing pour toucher le point faible et optimiser leur gain d'énergie.

Cette bascule rend le combat rythmé et polyvalent.

Condition de victoire/défaite :

Conditions de Victoire

Pour remporter une partie de Kill Dem'On, le joueur doit :
- Éliminer 1000 ennemis à travers une montée en puissance continue,
- Affronter et vaincre un boss final, conçu comme un pic de difficulté, exigeant la pleine maîtrise du système.

Nous avons opté pour un objectif clair, simple et brutal : 1000 éliminations, suivi d'un boss.
Ce choix sert plusieurs intentions :
- Créer un objectif lisible dès le départ, sans détour narratif ni objectifs secondaires.,
- Encourager la performance continue : il ne suffit pas de survivre, il faut dominer pendant toute la durée de la partie.
- Le boss final agit comme un test ultime de maîtrise du système : le joueur a engrangé de la puissance, maintenant il doit l'exploiter pleinement contre une menace unique.

Conditions de Défaite

La défaite survient si le joueur subit le moindre dégât alors que sa barre d'énergie est à zéro.

À 0 énergie :
- Le joueur ne meurt pas immédiatement, mais perd l'accès à certaines compétences et devient extrêmement vulnérable.
- C'est un état critique, où chaque mouvement devient risqué.
Un seul impact ennemi dans cet état entraîne la mort instantanée et met fin à la partie.

Pourquoi perdre à 0 énergie + un seul coup ?
Ce système de défaite repose sur un état de vulnérabilité extrême mais non immédiate, pour générer :
- Une tension dramatique forte dans les moments critiques : tu peux encore t'en sortir, mais chaque erreur est fatale.
- Une punition claire mais juste : tu perds parce que tu as mal géré ton énergie et parce que tu t'es fait toucher — pas parce qu'un système t'a surpris arbitrairement.
- Un jeu d'équilibre constant entre dépense et survie : l'énergie est ta seule ligne de vie, et c'est au joueur de la protéger.

Systeme de Reward :

Kill Dem'On ne propose aucun système de récompenses classiques. Pas d'argent, pas d'items, pas de progression permanente.
Chaque partie est un cycle autonome, sans héritage ni accumulation. Et pourtant, le jeu récompense constamment mais à sa manière.

Dans notre système, deux types de rewards ont été retenus et pleinement assumés :

Reward de Sustenance : l'énergie

L'énergie est la seule ressource active du jeu, à la fois :
- Gagnée en éliminant des ennemis,
- Dépensée pour se déplacer, attaquer, survivre,
- Volatile : elle n'est jamais stockée durablement, mais immédiatement consommée.

Elle incarne une reward instable et vitale, qui n'existe que pour être utilisée.
Elle n'accumule rien, mais entretient le game state.

Pourquoi ce choix ?
- Créer une tension constante entre survie et action.
- Forcer une lecture permanente du risque.
- Empêcher toute forme de confort ou de planification à long terme.
- Récompenser l'efficacité immédiate, pas l'économie.

Reward de Connaissance : la maîtrise du système
Le vrai gain du joueur ne réside pas dans ce qu'il possède, mais dans ce qu'il comprend :
- Comprendre comment gérer l'énergie.
- Reconnaître les patterns ennemis.
- Apprendre les meilleures priorités selon la situation.
- Savoir ce qui fonctionne ou non selon son état et son environnement.
- Découvrir des timings optimaux, des enchaînements, etc.

Ce reward n'est ni visible ni tangible, mais il est persistant : c'est le joueur qui évolue.

Pourquoi ce choix ?
- Mettre en avant la montée en compétence pure.
- Offrir une progression organique sans artifices.
- Rendre chaque partie formatrice, même en cas de défaite.
- Valoriser la rejouabilité par la maîtrise, pas par le farming.

Slippery slope et Come Back :

Dans Kill Dem'On, nous avons volontairement intégré un slippery slope énergétique :
Quand tu consommes mal ton énergie, tu descends de palier, tu perds en puissance et deviens plus vulnérable.

Mais pour éviter que cette pente ne devienne définitive ou injuste, nous avons mis en place des systèmes de comeback intégrés au cœur du gameplay, qui encouragent la prise de risque et la précision.

Récompense d'énergie proportionnelle au niveau de l'ennemi :
- Chaque ennemi a un niveau associé à la quantité d'énergie qu'il lâche à sa mort.
- Si le joueur est au palier 1, mais parvient à éliminer un ennemi de niveau 3 ou 4, il récupère l'énergie correspondante à ce niveau supérieur.
- Cela permet des remontées rapides si le joueur prend des risques, vise juste, et s'attaque à des cibles dangereuses.

Intentions design :
- Récompenser la prise d'initiative même en faiblesse.
- Maintenir une porte de sortie ouverte dans toutes les situations.
- Valoriser les décisions intelligentes plus que la simple survie.

Système de points faibles :
- Chaque ennemi (sauf boss) possède un point faible, visible uniquement après avoir subi un certain nombre de dégâts.
Si le joueur utilise un coup de poing sur ce point faible :,
- L'ennemi est tué instantanément,
- Il droppe plus d'énergie que la normale.

Ce système offre un raccourci tactique : même faible, le joueur peut rentabiliser au maximum ses actions s'il est précis.,

Intentions design :
- Offrir une opportunité technique pour les joueurs en difficulté.,
- Récompenser le risque calculé et le timing, pas le spam.,
- Relancer la boucle de gameplay avec un seul kill bien joué.

Courbe d’apprentissage :

Apprentissage :

L’apprentissage dans Kill Dem’On suit un modèle cyclique, centré sur l’énergie et la maîtrise des compétences, avec une approche Learn -> Practice -> Master très claire. Le joueur est confronté à un système épuré mais exigeant, qui demande :

Compréhension du système d’énergie :

- Chaque action a un coût -> apprendre à ne rien faire "gratuitement"
- Apprendre à "récupérer pour mieux dépenser"

Lecture des ennemis :

- Identifier les patterns, les évolutions par paliers
- Comprendre les timings de point faible
- Gérer plusieurs types d’ennemis en simultané

Maîtrise des compétences :

- Apprendre les timings de saut, sprint, pilonnage, coup de poing
- Optimiser les enchaînements dans des environnements hostiles
- Adapter ses choix à son énergie restante (penser efficacité)

Savoir quand risquer, et quand temporiser :

- Économie mentale constante : "Est-ce que ce move vaut sa dépense ?"

L’apprentissage est à la fois mécanique (input) et cognitif (gestion, anticipation). Il n’est pas linéaire : chaque erreur sert de leçon, chaque kill est une micro-victoire dans une boucle.

Difficulté :

La difficulté de Kill Dem’On est progressive, adaptative et punitive :

- Progressive : Les ennemis évoluent au fil des paliers, deviennent plus rapides, plus résistants, et plus nombreux.
- Adaptative : Le jeu répond indirectement à ton état (palier), en augmentant la pression si tu deviens plus fort.

Punitive mais juste :

- Aucune régénération passive.
- À 0 énergie, tu es en danger de mort immédiat : un seul coup et c’est fini.
- Si tu veux survivre, chaque dépense doit être un investissement rentable.

Progression :

La progression dans Kill Dem’On ne repose pas sur des déblocages de contenu, mais sur un sentiment de maîtrise, structuré en paliers énergétiques :

Micro-progression immédiate :

- Chaque ennemi tué -> énergie
- Energie -> compétence activée
- Compétence -> mobilité/pouvoir/avantage

Macro-progression par palier :

- Palier 1 à 4 = montée en puissance
- Chaque palier déverrouille plus de vitesse, plus d’outils, plus de liberté

Objectif final :

- 1000 ennemis tués = accès au boss
- Tuer le boss = victoire
- Optimiser sa run = place sur le leaderboard

Score Systeme :

Intention :
Le système de score est au cœur de la motivation et de l'engagement à long terme dans notre jeu. Il repose sur l'efficacité du joueur et sa maîtrise des différents paliers du gameplay.
L'objectif est de fidéliser le joueur et de le pousser à se surpasser. Nous avons constaté une réelle profondeur dans notre jeu, et le score agit comme un levier pour maintenir l'intérêt du joueur au fil des parties.
La première partie du jeu est très différente de la cinquième, et le système de score sert justement à encourager la progression et la maîtrise.

Ce système repose sur deux facteurs : la rapidité d'exécution de l'objectif et le nombre d'ennemis éliminés.

Lié au temps :
Le score est tout d'abord lié au temps. Plus l'objectif est accompli rapidement, plus le multiplicateur de score augmente.
Le temps agit donc comme un multiplicateur du score global, récompensant les joueurs les plus efficaces.

Lien Combo Système :
Le système de combo est également lié au score. En effet, le score obtenu à chaque élimination dépend du combo en cours : si le multiplicateur affiche ×3, alors le score de l'élimination sera multiplié par trois.
Cela pousse le joueur à enchaîner les éliminations, sans quoi son score sera forcément plus faible.

Graphique du multiplicateur de score en fonction du temps de réussite de l'objectif

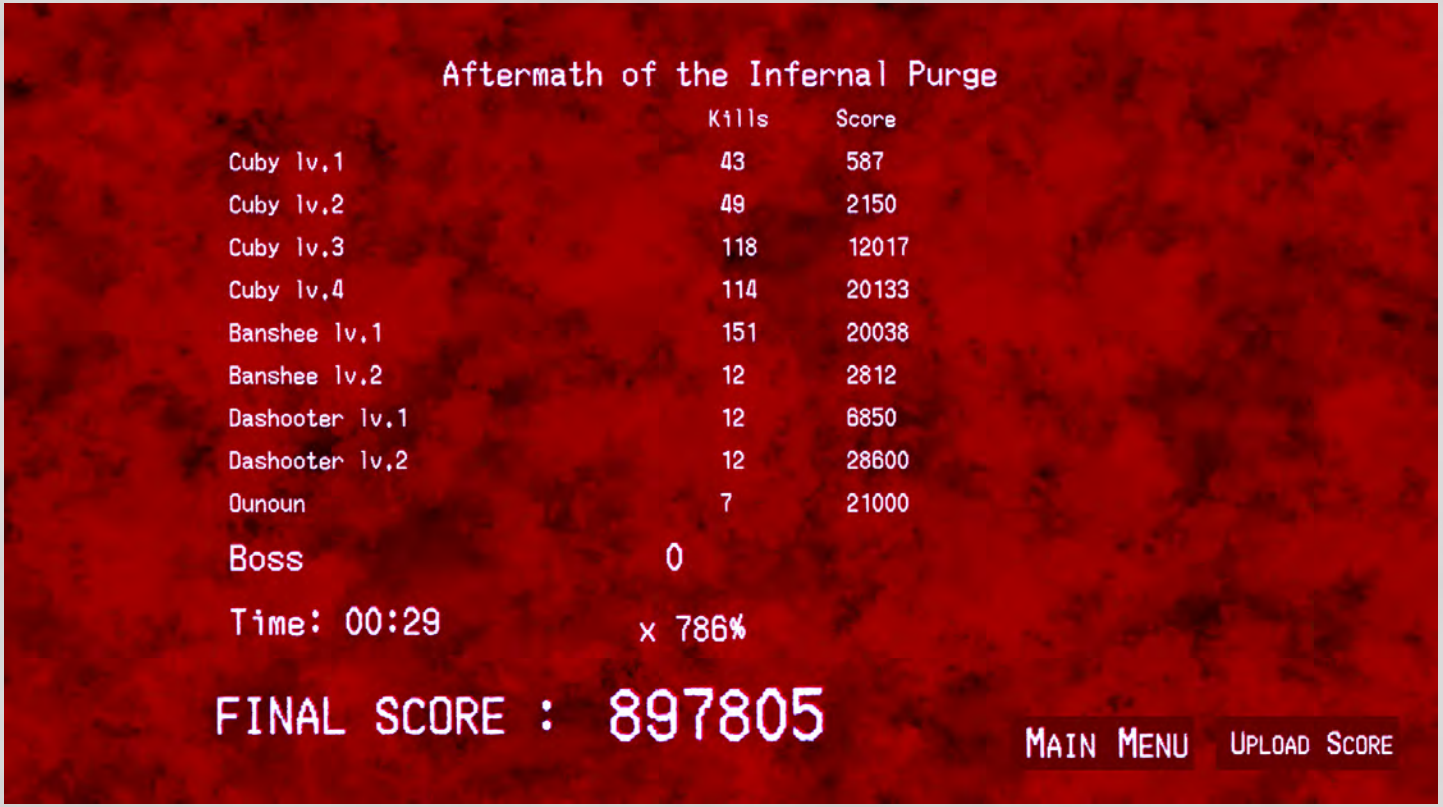


Image du score final d'une partie

Score d'ennemi :

Le score est principalement lié à l'élimination des ennemis. Chaque ennemi tué fait progresser l'objectif principal, mais surtout, chaque type d'ennemi rapporte un score différent.

Cela ajoute une dimension stratégique au système de score, mais aussi à chaque action du joueur. Pour obtenir le score optimal, le joueur doit éliminer les ennemis le plus rapidement possible, tout en choisissant soigneusement ses cibles afin de maximiser ses gains.

LeaderBoard :
Afin d'ajouter un aspect encore plus compétitif à notre jeu, nous avons intégré un leaderboard global. Ce dernier est connecté à un serveur et regroupe tous les joueurs.
Il permet aux plus persévérants de grimper progressivement dans le classement, jusqu'à peut-être apparaître en première page — voire atteindre la première place.

Ce système renforce fortement la rejouabilité de notre jeu. En effet, l'aspect compétitif correspond parfaitement à notre cible de joueurs confirmés, amateurs de fast-FPS exigeants.

Le leaderboard enregistre trois informations essentielles :
- Le temps de complétion
- Le score global
- Le gamertag du joueur

Image du leaderboard

Systeme de spawners :

Intentions de design :

Le système de spawners dans est conçu pour maintenir une pression constante et stratégique autour du joueur, sans jamais le plonger dans un chaos incontrôlable.

Les objectifs :

- Créer une sensation de submersion maîtrisée
- Forcer le déplacement du joueur pour éviter le camping
- Donner un rythme soutenu et progressif à l'action
- Favoriser l'apprentissage des zones et des types d'ennemis

Fonctionnement du système :

Le système de spawn repose sur une structure modulaire par zones.

- Une carte divisée en zones interconnectées

L'espace de jeu est découpé en zones circulaires de spawn (zones colorées sur le schéma), qui se superposent partiellement.
Lorsque le joueur entre dans l'une d'elles, elle s'active : les spawners qu'elle contient commencent alors à générer des ennemis.
Le cercle noir représente la zone globale du jeu, tandis que chaque cercle coloré symbolise une zone de spawn autonome mais interconnectée.

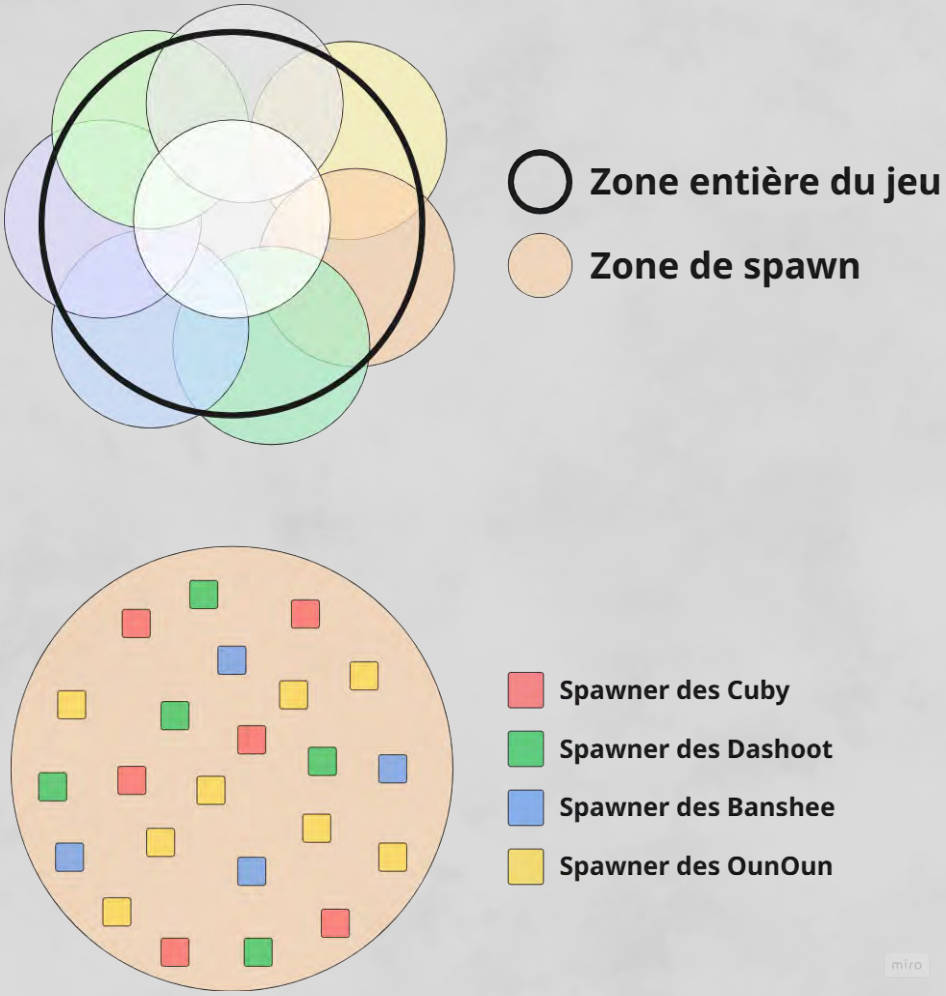


Schéma du fonction des emplacements du spawn

Spawners spécialisés par ennemi

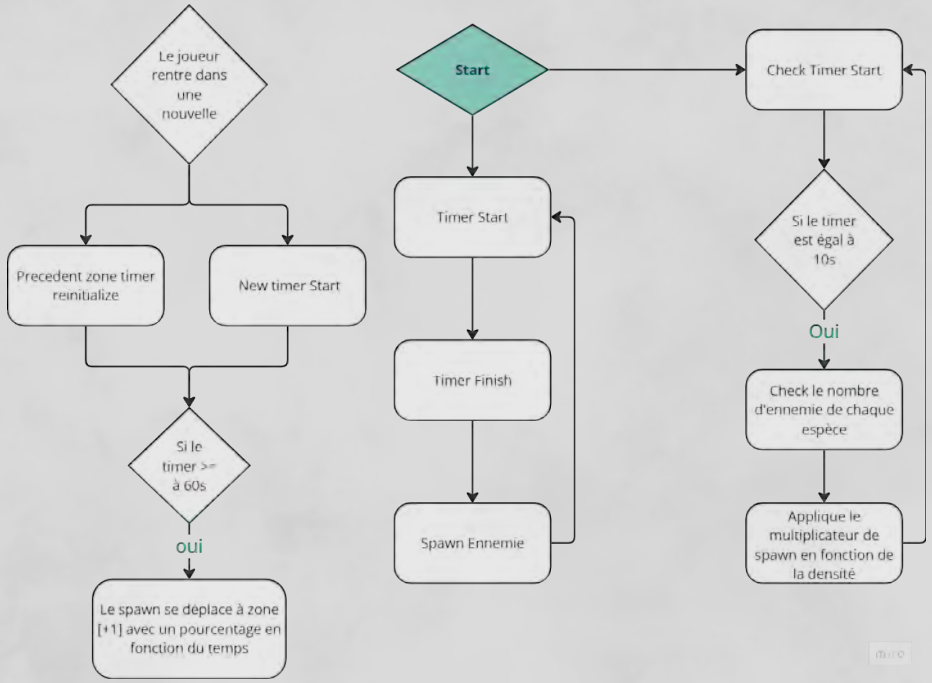
À l'intérieur de chaque zone, plusieurs spawners sont positionnés manuellement. Chaque type de spawner est associé à une espèce ennemie spécifique.

Cette organisation permet une distribution équilibrée des ennemis dans chaque zone, tout en laissant au joueur la possibilité d'anticiper les menaces selon la zone traversée.

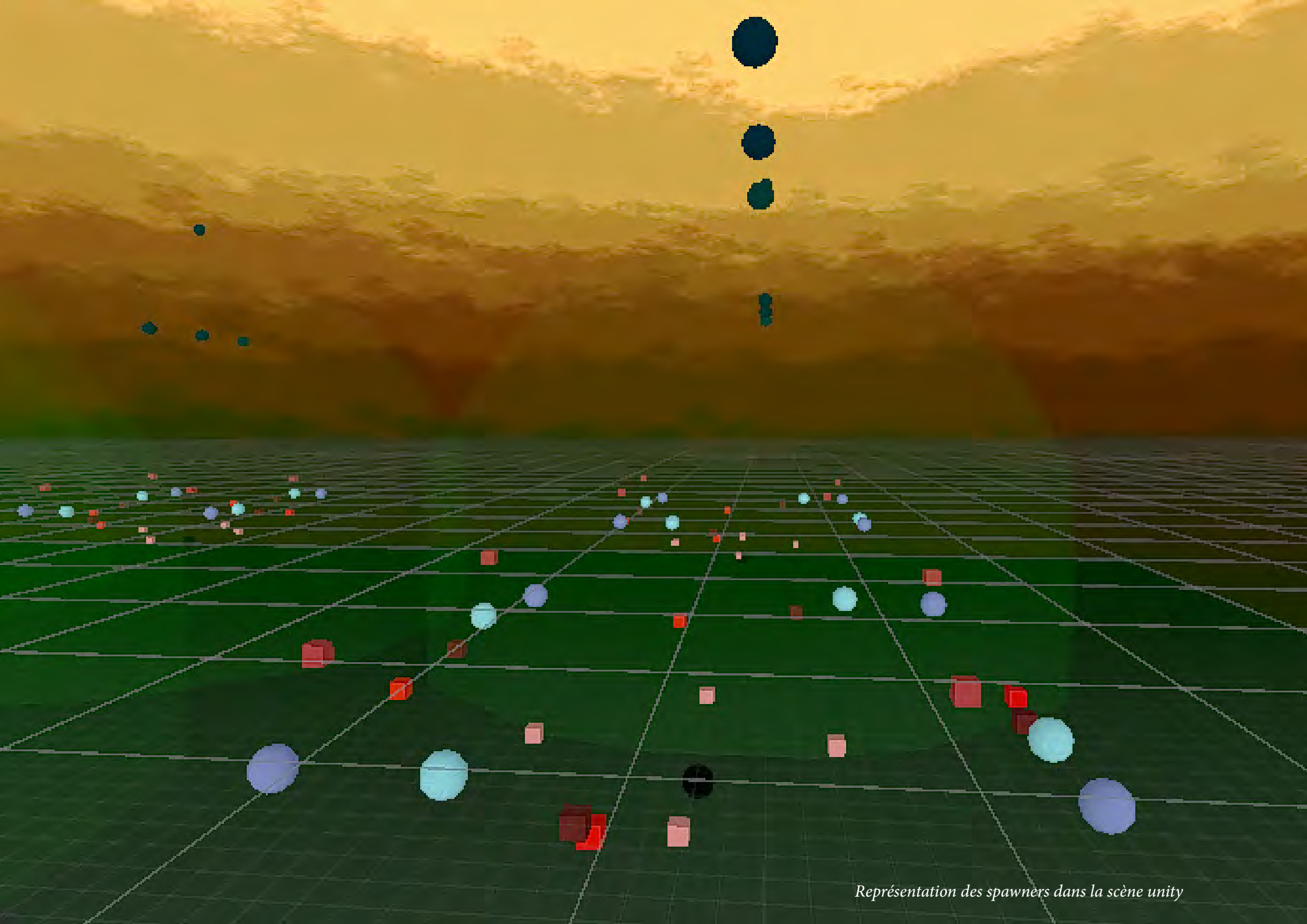
Le cercle rose sur le schéma du bas montre l'unité logique d'une zone activée, avec les différents spawners colorés prêts à générer leurs types d'ennemis respectifs.

Spawn dynamique

Quand une zone est active :
- Chaque spawner peut faire apparaître un ennemi aléatoirement, selon un timer et une quantité définie par palier.
- Le système choisit aléatoirement un spawner de chaque type pour maintenir une variété dans les apparitions.
- Ce système garantit que le joueur se sente entouré, sans pour autant être attaqué toujours depuis les mêmes points.



FlowChart du système de spawn



Représentation des spawners dans la scène unity

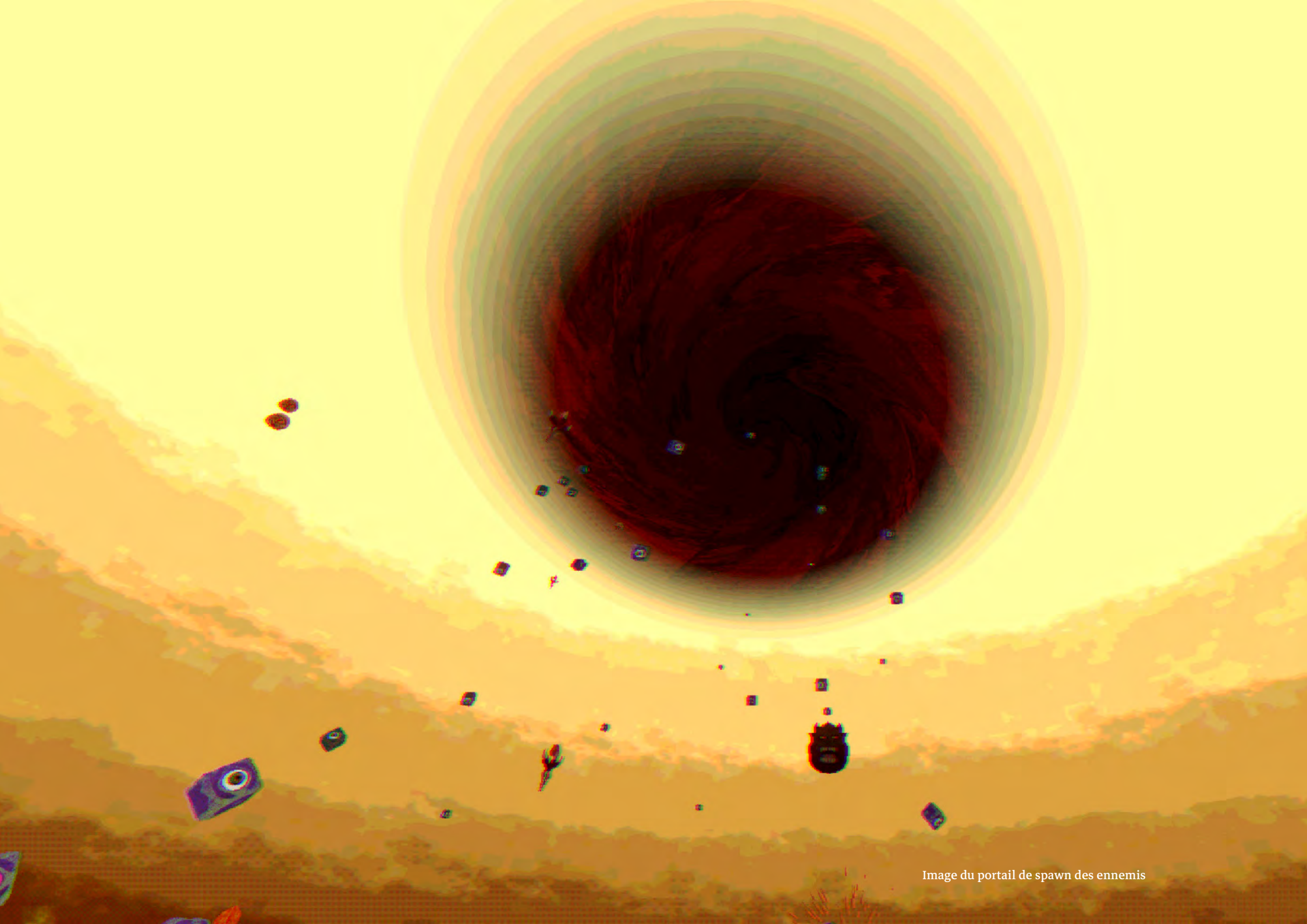


Image du portail de spawn des ennemis

Recherches et intégration d'une DDA

Un système de spawners inspiré de la DDA

Notre système de spawners n'est pas figé : il évolue selon la position du joueur, son niveau de performance, et le palier atteint. Il s'agit d'une forme simplifiée de DDA (Dynamic Difficulty Adjustment), intégrée directement au cœur de notre boucle de jeu.

En ajustant la fréquence, la quantité et la répartition des spawns en fonction du joueur, on parvient à maintenir une pression dynamique, qui suit le rythme du joueur sans jamais le briser brutalement. Cette approche renforce notre objectif global : maintenir la tension, encourager l'agressivité et récompenser la prise de risque.
Une recherche basée sur des jeux à vague dynamique

Dans notre démarche de conception, nous avons étudié plusieurs formes de DDA utilisées dans l'industrie, notamment :

- Le système de vagues contrôlées de Left 4 Dead.
- L'ajustement de difficulté adaptatif de Resident Evil 4.

Ces jeux ont en commun de s'appuyer sur des données statistiques (temps sans kill, dégâts subis, efficacité offensive...) pour adapter le rythme du jeu. Nous avons repris ce principe.

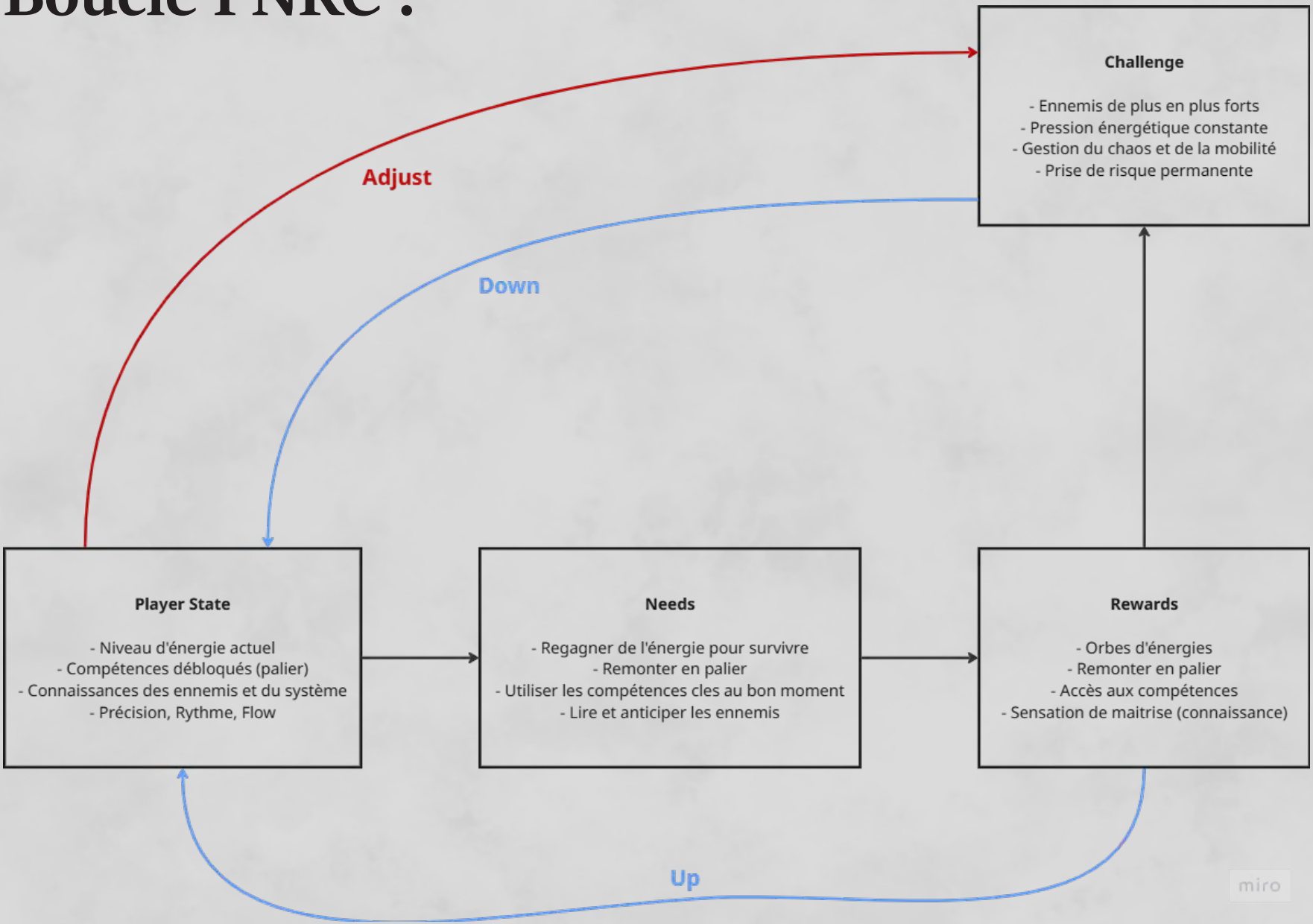
Une DDA intégrée au rythme

Contrairement à certaines approches de DDA qui ralentissent ou facilitent le jeu pour aider un joueur en difficulté, notre système ne vise ni à freiner, ni à corriger.
Il est conçu pour accompagner le tempo du joueur, en maintenant la courbe de tension ludique au plus près de ce que demande notre système de jeu.

Dans Kill Dem'On, le joueur est un prédateur, pas une victime :
- Le système ne cherche pas à équilibrer pour équilibrer, mais à prolonger l'état de danger contrôlé.
- Il agit en continu, sans effet visible ou déclencheur brutal.
- Il réinjecte du rythme, pas de difficulté arbitraire.

Lien de nos recherches sur la DDA :
https://www.notion.so/DDA-1a5fe1db0a5180a190fefd91214b8ae6?pvs=4

Boucle PNRC :


Challenge
- Ennemis de plus en plus forts
- Pression énergétique constante
- Gestion du chaos et de la mobilité
- Prise de risque permanente
Adjust
Down
Player State
- Niveau d'énergie actuel
- Compétences débloqués (palier)
- Connaissances des ennemis et du système
- Précision, Rythme, Flow
Needs
- Regagner de l'énergie pour survivre
- Remonter en palier
- Utiliser les compétences cles au bon moment
- Lire et anticiper les ennemis
Rewards
- Orbes d'énergies
- Remonter en palier
- Accès aux compétences
- Sensation de maitrise (connaissance)
Up
miro

Notre jeu repose sur une boucle minimaliste mais intense, construite autour du système énergétique et de l'apprentissage pur.

Player State :

Dans Kill Dem'On, le player state n'est pas basé sur des stats ou des objets persistants, mais sur :
- Le niveau d'énergie actuel, qui définit l'accès aux compétences.
- Le palier atteint, qui reflète la puissance et la mobilité.
- La maîtrise acquise du système : connaissance des ennemis, des timings, des priorités.

Il s'agit donc d'un état volatil mais profond, où tout est lié à l'instant et à la compétence réelle du joueur.

Challenge

Le challenge est dynamique, et évolue en fonction directe du player state :
- Lorsque le joueur monte en palier, il débloque de nouvelles compétences...
- ... mais en parallèle, les ennemis deviennent plus agressifs, plus nombreux et plus résistants.

Le système crée un effet miroir :
- Plus tu es fort, plus le jeu t'en demande.

Cela permet :
- Un ajustement automatique du challenge sans cassure dans le flow.
- Une tension constante, même au sommet de ta puissance.
- Une pression toujours adaptée : tu n'es jamais trop fort pour l'arène.

Needs :

Les besoins du joueur émergent de la boucle elle-même :
- Il a besoin d'énergie pour rester en vie et monter en puissance.
- Il a besoin de comprendre pour optimiser ses actions.
- Il a besoin de précision, de rapidité, de timing — pas d'équipement.

Ces besoins sont ressentis et auto-alimentés. On n'a pas besoin de forcer la progression : la survie crée le besoin.

Le reward dans ce système Le reward n'est pas un objet, mais une capacité regagnée, un palier récupéré, ou une maîtrise mieux incarnée. Cela crée :
- Une motivation intrinsèque, où le joueur revient non pas pour gagner plus, mais pour jouer mieux.
- Une satisfaction liée à la performance et non à la collection.
- Une boucle qui récompense l'efficacité, l'intelligence d'action et l'adaptation.

Pourquoi ce choix ? Nous avons choisi une boucle de motivation centrée sur le player state et l'adaptation du challenge, et non sur la profusion de contenus ou la récompense extérieure.
- Pas de loot, tout se joue sur ce que tu sais.
- Pas d'accumulation,tu recommences, mais tu recommences différemment.
- Pas d'objets, mais chaque session te rend plus fort, toi, pas ton personnage.

Ce design permet de créer une expérience hautement rejouable, intense, et méritée : Tu joues pour éprouver, pour maîtriser, pas pour remplir un inventaire.

MDA :

Aesthetics :

Challenge :

Le fun a challenger le joueur :
- **Confrontation** : Le combat perpétuel dans lequel se plonge le joueur dès le début de la partie verra sa difficulté croître au fil de son évolution. À mesure qu'il progresse, le jeu deviendra de plus en plus exigeant, demandant une maîtrise accrue de ses compétences. Cela permettra de maintenir un niveau de difficulté équilibré en fonction des améliorations du joueur.

- **Soft Gestion** : L'énergie, essentielle à la survie du joueur, à l'utilisation de ses compétences et à sa progression, est une ressource qu'il récolte en éliminant des monstres. Chaque action effectuée consomme de l'énergie, ce qui impose une gestion attentive de cette ressource. Cependant, le système reste généreux et tolérant envers les erreurs, permettant au joueur de s'adapter sans être excessivement pénalisé. Trouver l'équilibre entre le coût des actions et la récolte d'énergie est au cœur de notre jeu.

- **Précision** : Les capacités de mouvement du joueur et des démons étant primordiales, un défi de précision se pose. L'élimination des ennemis repose principalement sur le tir, qui semble être le moyen le plus simple. Cependant, les déplacements, auparavant un atout pour le joueur, augmentent désormais la difficulté en exigeant une visée plus précise. De plus, chaque tir consommant de l'énergie, manquer sa cible devient d'autant plus pénalisant, renforçant l'importance d'une gestion rigoureuse des ressources.

- **Anticipation** : Au fil du jeu, le joueur sera de plus en plus rapidement submergé par les démons. Il devra alors anticiper ses prochaines actions et séquences de jeu à l'avance, là où il pouvait auparavant se contenter d'improviser. S'il ne le fait pas, il risque de perdre totalement le contrôle de la situation et, à terme, la partie.

Sensation :

Le fun de la sensation :
- **Mouvement :** Nos mouvements dynamiques rendent le fait de se déplacer dans le jeu amusant et procurent une sensation de liberté qui s'intensifie au fil des paliers, le tout étant fortement accentué par des feedbacks.
- FeedBack : Les retours visuels et sonores dont bénéficiera le joueur tout au long de son gameplay seront nombreux et riches, afin d'amplifier le plaisir procuré par les actions effectuées dans le jeu. Cela passe par différentes techniques, comme la sensation de gestion de la caméra et l'ajustement du champ de vision (FOV).

- **Stress/Pression :** Au cours du jeu, le joueur sera soumis à du stress et de la pression, car il sera constamment assailli par des démons venant de toutes parts. Cette menace permanente créera une tension liée à la nécessité de tous les éliminer, mais aussi à sa propre survie dans un environnement hostile. Ces situations, pouvant mener à un risque de défaite, généreront par moments un stress intense.

- **Puissance :** Au cours du jeu, le joueur est amené à évoluer. Cette montée en puissance doit être ressentie de manière drastique afin de l'inciter à profiter de son niveau actuel tout en le poussant à progresser encore davantage. Cependant, cette augmentation de puissance s'accompagne de nouveaux dangers, avec l'apparition d'ennemis plus puissants et robustes. Il s'installe alors un véritable duel de force entre le joueur et le jeu, une lutte constante pour déterminer qui sera le plus fort. Ces éléments doivent être perçus par le joueur à tout moment.

Fantasy :

Le fun de la fantasy :
- Incarnation : Étant à la première personne, notre jeu suggère une incarnation directe du joueur dans l'espace de jeu au travers de son avatar. Ce faisant, les sensations liées au personnage, telles que l'évolution, seront amplifiées par cette immersion. Le joueur aura ainsi la sensation que c'est LUI qui évolue et devient plus puissant, et non simplement son personnage.

- Immersion : Cette immersion amplifie également la satisfaction du joueur face au carnage qu'il perpétue dans le jeu. Si tuer des démons est déjà amusant, le fait de ressentir cette action comme étant réellement accomplie par lui-même, grâce à une immersion réussie, rend l'expérience d'autant plus plaisante.

Recette du fun :

Notre esthétique principale est le Challenge (MDA), car il est essentiel à notre jeu et constitue le moteur du fun. Cependant, ce seul aspect n'est pas suffisant pour rendre le jeu réellement amusant. C'est pourquoi notre deuxième esthétique principale, la Sensation (MDA), vient ajouter une dimension satisfaisante aux actions du joueur. Ces deux esthétiques combinées créent une dynamique de challenge récompensée, offrant ainsi un gameplay plaisant. Notre dernière esthétique, la Fantasy (MDA), est secondaire. Bien que le cœur de notre jeu repose sur le carnage et le dynamisme, cette esthétique nous permet d'amplifier les sensations et le plaisir qu'elles génèrent.

La recette du fun de notre jeu, Kill Dem'On, est donc :
Challenge, Sensation, Fantasy

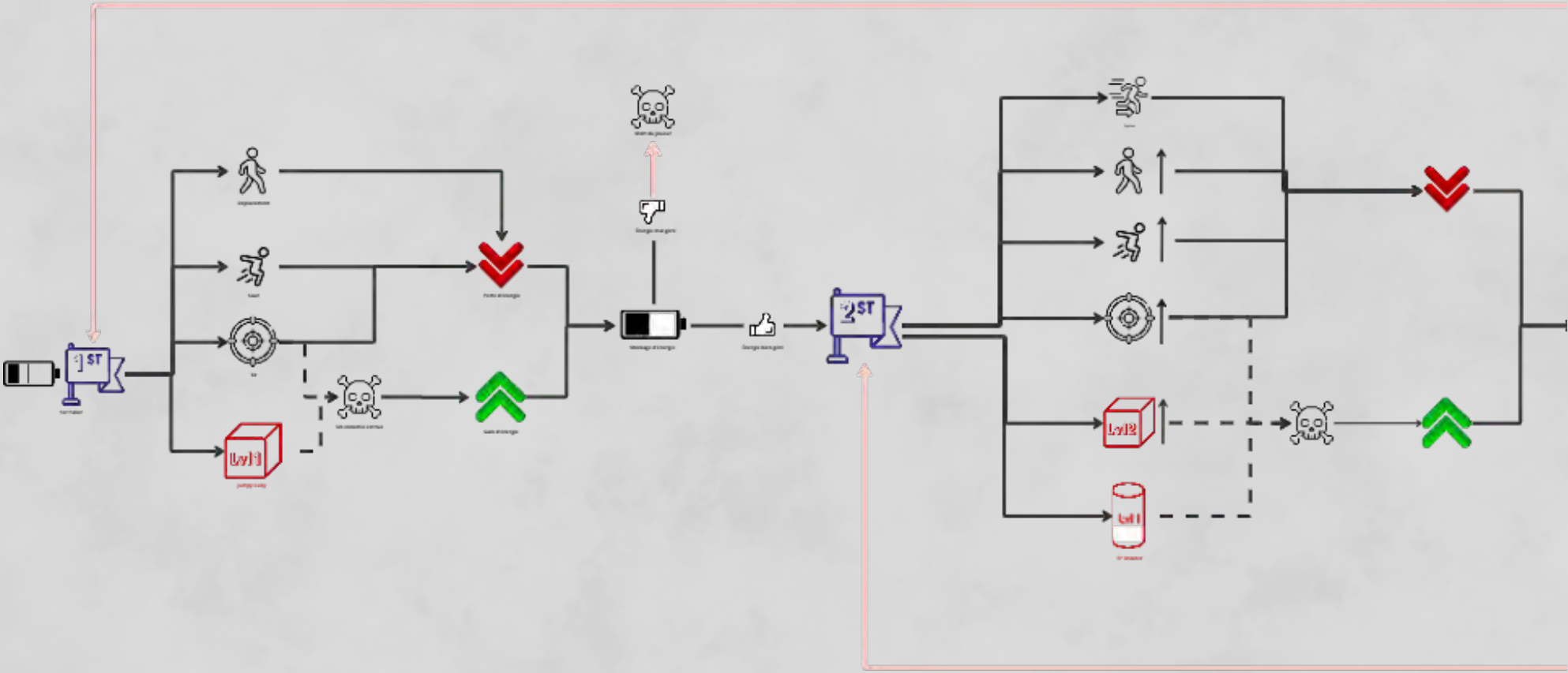


Schéma du fonctionnement globale du jeu

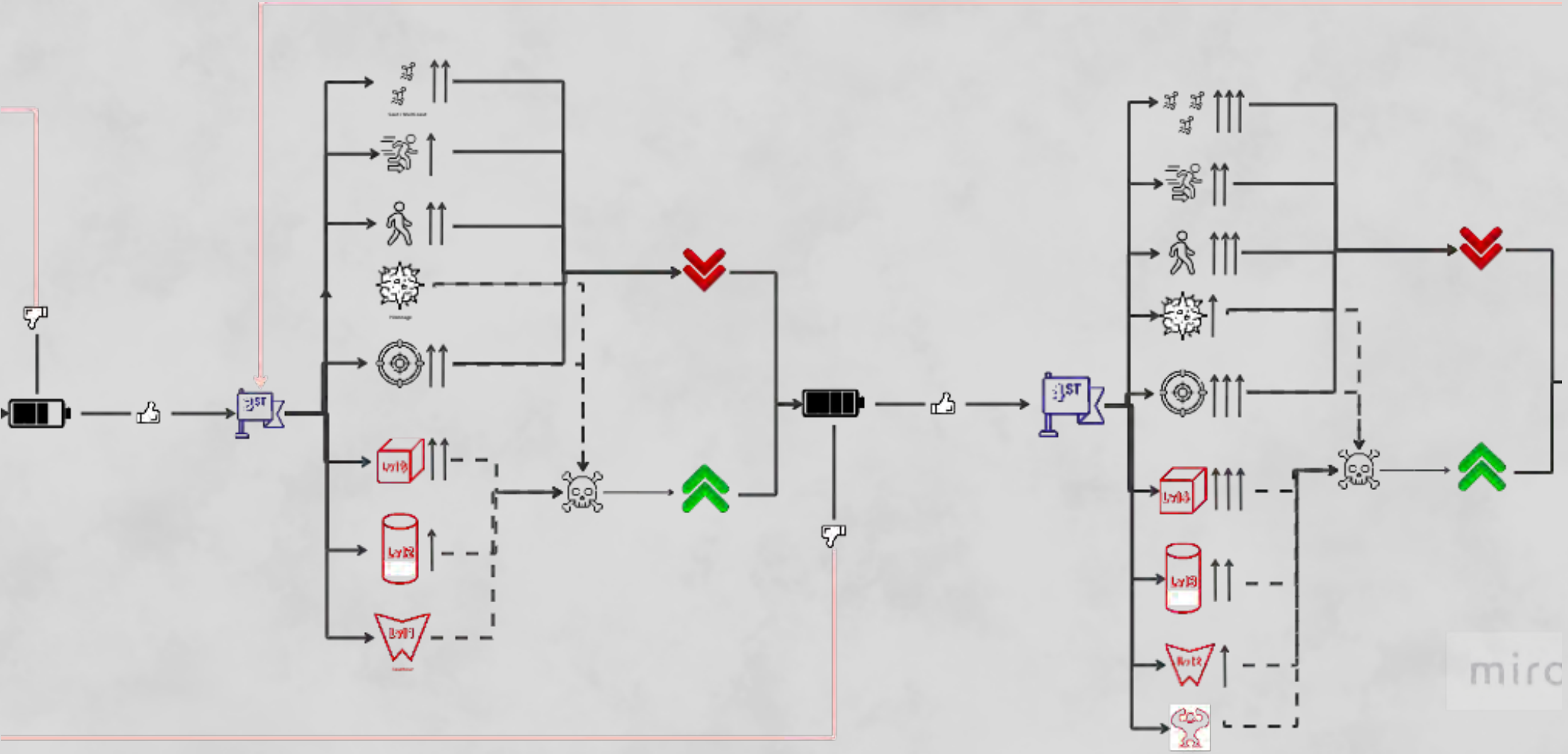
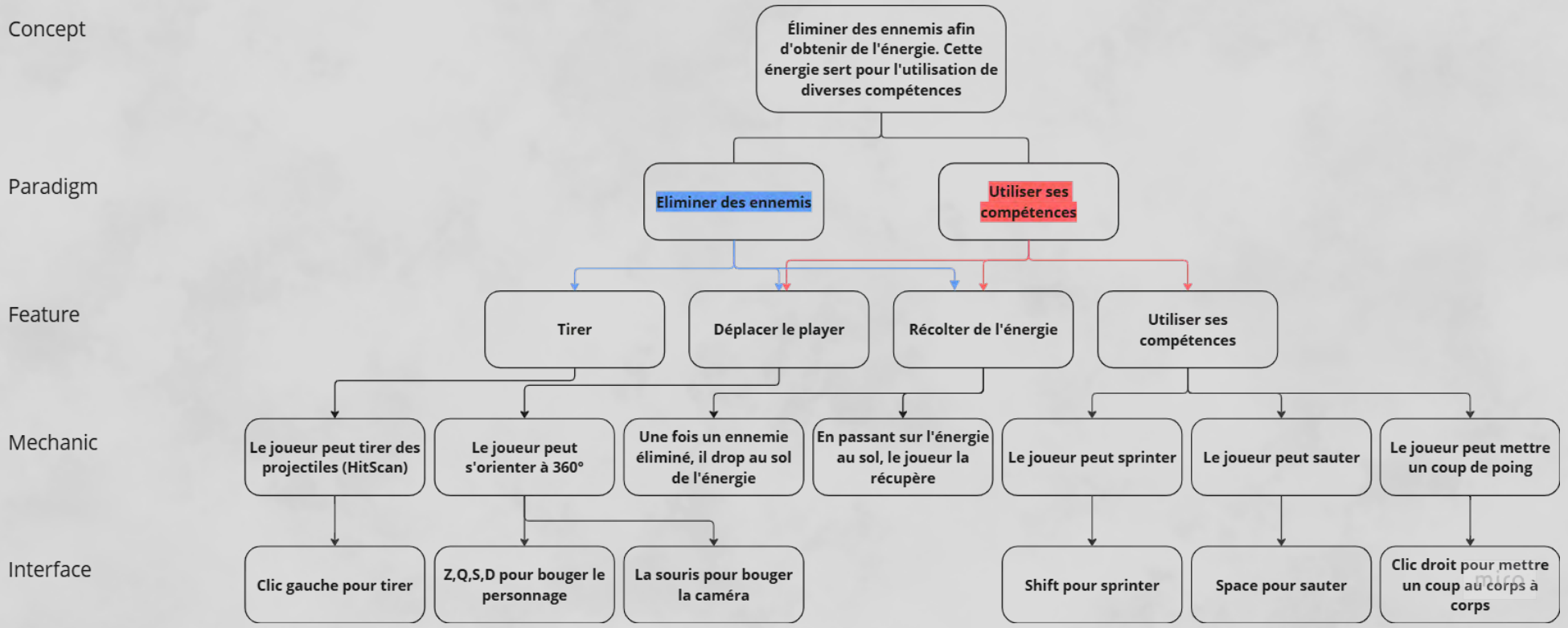
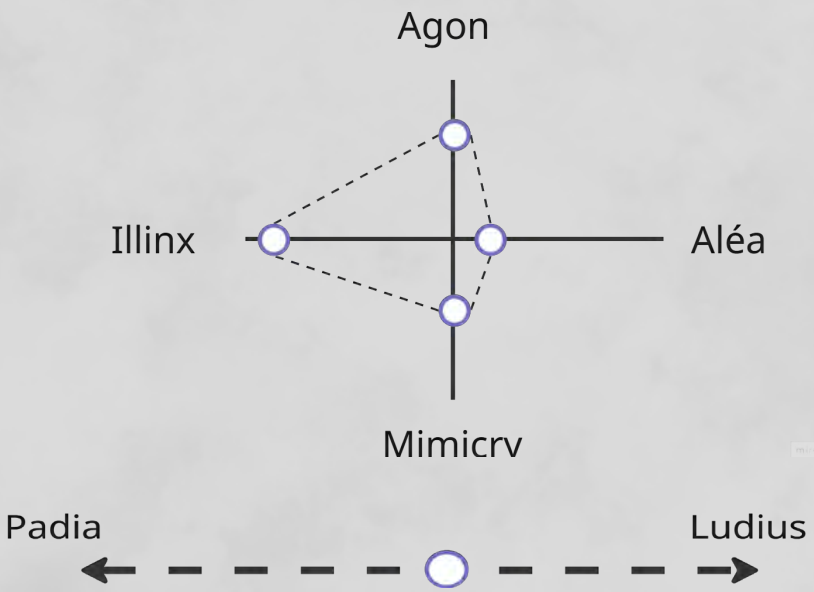


Schéma de Ventrice :





Un vertige (Illinx) accentué par la sensation de puissance, et donc la Mimicry

Kill Dem'On provoque une forme de vertige contrôlé, non pas par la perte de repères sensoriels, mais par une expérience de puissance débridée.
L'accélération du FOV, les dashs violents, les pilonnages aériens et les feedbacks visuels renforcent cette sensation de transe dynamique.
Le joueur incarne une entité destructrice, une force surnaturelle. Ce pouvoir incarné, renforcé par l'immersion à la première personne et la montée en paliers, active une forme de Mimicry : on ne joue pas un humain, mais une puissance. Le plaisir vient alors d'habiter ce rôle, et de ressentir viscéralement sa montée en contrôle.

Un Agon lié au leaderboard, au score et au combat contre les ennemis

La structure de Kill Dem'On repose aussi sur une logique hautement compétitive.
Le joueur est constamment mis à l'épreuve :
- par les ennemis évolutifs,
- par la pression énergétique constante,
- par le système de combo et la gestion optimisée des ressources.

À cela s'ajoute le leaderboard final, qui compare les performances sur la base d'un score public.
On est donc dans une logique d'Agon pur, où la performance individuelle devient une échelle de valeur, et où le skill est le seul critère de progression.

Ludius vs Padia : une tension cachée entre rigueur et liberté

En surface, Kill Dem'On semble relever d'un Ludius strict :
des règles claires,
- une difficulté croissante,
- un objectif chiffré (1000 ennemis, un boss, un score).

Mais en réalité, le cœur de l'expérience est Padia.
Le joueur improvise, réagit, invente son flow.
Il n'y a pas une méthode, mais une infinité de micro-décisions possibles :
- Quelle compétence privilégier ?
- Fuir ou attaquer ?
- Risquer un point faible ou sécuriser une zone ?

Chaque action est réfléchie dans l'instant, selon les sensations, le rythme, l'analyse personnelle.
C'est une liberté tactique déguisée sous une apparente rigueur.

Boucle OCR:

Objectif	Challenge	Reward
Tuer un ennemi	- Lire le pattern d'un ennemi et anticiper ses déplacements. - Se positionner intelligemment dans l'arène pour éviter d'être touché. - Maintenir une précision constante en tir tout en gérant sa mobilité.	Obtention d'une orbe d'énergie

Objectif	Challenge	Reward
Toucher un point faible	- Identifier visuellement et temporellement l'activation du point faible. - Rapprocher le joueur sans mourir : lecture de l'environnement + anticipation ennemie. - Aligner spatialement et rythmiquement le coup de poing avec la hitbox exposée. - Gérer la prise de risque : c'est un pari entre l'économie d'énergie et la menace directe de mort. - Être suffisamment mobile et précis pour transformer cette fenêtre courte en opportunité stratégique.	Kill instantané + regain bonus

Objectif	Challenge	Reward
Gagner un palier	- Enchaîner les éliminations sans laisser retomber le rythme. - Rester en mouvement pour esquiver tout en maximisant ses dégâts par secondes. - Conserver un équilibre entre dépense et récupération d'énergie à chaque action. - Adapter sa stratégie à l'état actuel du jeu (ennemis plus dangereux aux paliers supérieurs). - Prendre des risques calculés (point faible, corps-à-corps, timing) pour aller plus vite tout en évitant la chute d'énergie.	Accès à de nouvelles compétences / plus de puissance

Objectif	Challenge	Reward
Tuer les 1000 ennemis	- Maîtriser parfaitement toutes les mécaniques fondamentales du jeu. - Gérer la pression constante sans relâcher la vigilance. - Maintenir un flow optimal, sans déséquilibre d'énergie, tout au long de la progression. - S'adapter à la montée en puissance des ennemis au fil des paliers. - Revenir d'une situation critique (palier bas) sans perdre le fil de sa performance.	Débloqué l'accès au boss final

Objectif	Challenge	Reward
Finir 1er leaderboard	- Optimiser chaque instant de la partie. - Prendre toutes les bonnes décisions tactiques sous pression, sans erreur de jugement. - Maximiser le multiplicateur de combo sans interruption. - Ne jamais perdre un palier, ou savoir le récupérer dans les secondes critiques. - Être rapide, précis, et efficace en permanence, avec zéro gaspillage de ressource.	Reconnaissance, estime personnelle

Objectif	Challenge	Reward
Tuer le boss	- Mobiliser l'ensemble des compétences maîtrisées pendant la partie. - Réagir sans délai, tout en gérant les timings précis de chaque compétence (pilonnage, dash, coup de poing). - S'adapter à un environnement plus fermé, plus risqué, sans source d'énergie facile. - Garder son calme et sa rigueur sur une séquence où une seule erreur peut être fatale.	Fin du jeu / Score total

Ennemis

Nos ennemis :

Intentions de design

Dans Kill Dem'On, les ennemis ne sont pas uniquement des obstacles : ils sont le carburant du gameplay. Chaque ennemi est une ressource potentielle, un déclencheur de skill, et une variable de tension.

Ils sont conçus pour :
- Pousser le joueur à agir vite et précisément
- L'inciter à utiliser ses compétences intelligemment
- Le récompenser en énergie et en rythme lorsqu'il réussit
- Construire un environnement de jeu où chaque type d'ennemi incite le joueur à adapter sa manière de jouer

Un écosystème basé sur le danger et la récompense

Chaque ennemi a une fonction systémique claire :
- Certains submergent (pression constante)
- D'autres chassent (menace ciblée)
- D'autres encore rendent le joueur mobile (projectiles, AOE, course)
- Et certains offrent simplement une source d'énergie, sans représenter de menace

L'objectif n'est pas d'empiler les difficultés, mais de composer une arène dynamique où les ennemis interagissent entre eux et avec les capacités du joueur. Points faibles & montée en puissance
Tous les ennemis du jeu disposent d'un point faible à la fois visuel et mécanique.

Ce point faible apparaît uniquement après qu'un certain nombre de dégâts a été infligé à l'ennemi. Il sert plusieurs objectifs :
- Permettre de terminer l'ennemi plus rapidement s'il est visé avec précision
- Offrir un bonus d'énergie si le joueur frappe ce point avec un coup de poing
- Créer une lecture visuelle intuitive de l'état de l'ennemi (le point faible est illuminé et bien identifiable)

Progression ennemie à chaque palier

Les ennemis sont introduits palier par palier, mais ne disparaissent jamais. À chaque palier, tous les ennemis voient leur durabilité et leur dangerosité évoluer :
- Leur vie augmente
- Leurs dégâts sont amplifiés
- Certains voient aussi leur fréquence d'attaque ou vitesse légèrement ajustées

Cette montée progressive permet de :
- Maintenir leur pertinence dans les phases avancées
- Varier les compositions de menace dans les arènes
- Offrir une courbe de difficulté vivante, toujours cohérente avec la progression du joueur
- Renforcer une pression constante, sans jamais briser le rythme

Cela garantit que même les ennemis introduits tôt dans la partie restent menaçants et forcent le joueur à adapter son engagement.

Une conception liée aux compétences du joueur

Chaque ennemi est designé en lien direct avec une ou plusieurs compétences du joueur :
- Certains obligent à sauter ou à sprinter
- D'autres demandent des tirs précis ou des flics rapides
- Certains ne peuvent être vaincus qu'en engageant physiquement ou en gérant leur environnement

L'idée est de faire de chaque rencontre une opportunité de skill expression.

Feedseed :

Intention de design

La FeedSeed est une source de régénération libre dans un monde où l'énergie est au cœur de chaque action. Elle n'impose aucun danger direct, mais apporte une tension stratégique :
- Faut-il l'utiliser maintenant ?
- Ou la conserver pour plus tard ?
- Sera-t-elle encore là si je reviens ?

Elle stabilise le rythme du joueur, sans jamais le sortir du danger.

Fonctionnement :

- Entité statique placée à des endroits choisis sur la carte
- Le joueur peut tirer dessus pour en extraire de l'énergie
- Une fois utilisée, elle entre en recharge pendant un délai fixe
- Lorsqu'elle est prête, elle redevient disponible automatiquement

Impact gameplay

- Permet de remonter une jauge rapidement, sans éliminer d'ennemi
- Offre une sécurité mentale dans certaines zones (“backup spot”)
- Crée des zones d'intérêt secondaires que le joueur mémorise
- Peut devenir un objectif défensif ou offensif selon son positionnement

Variation de challenge

Le challenge n'est ni dans l'action, ni dans l'ennemi, mais dans :
- Le temps de recharge
- L'espace de placement
- La gestion du risque autour (la récupérer ou mourir en essayant)

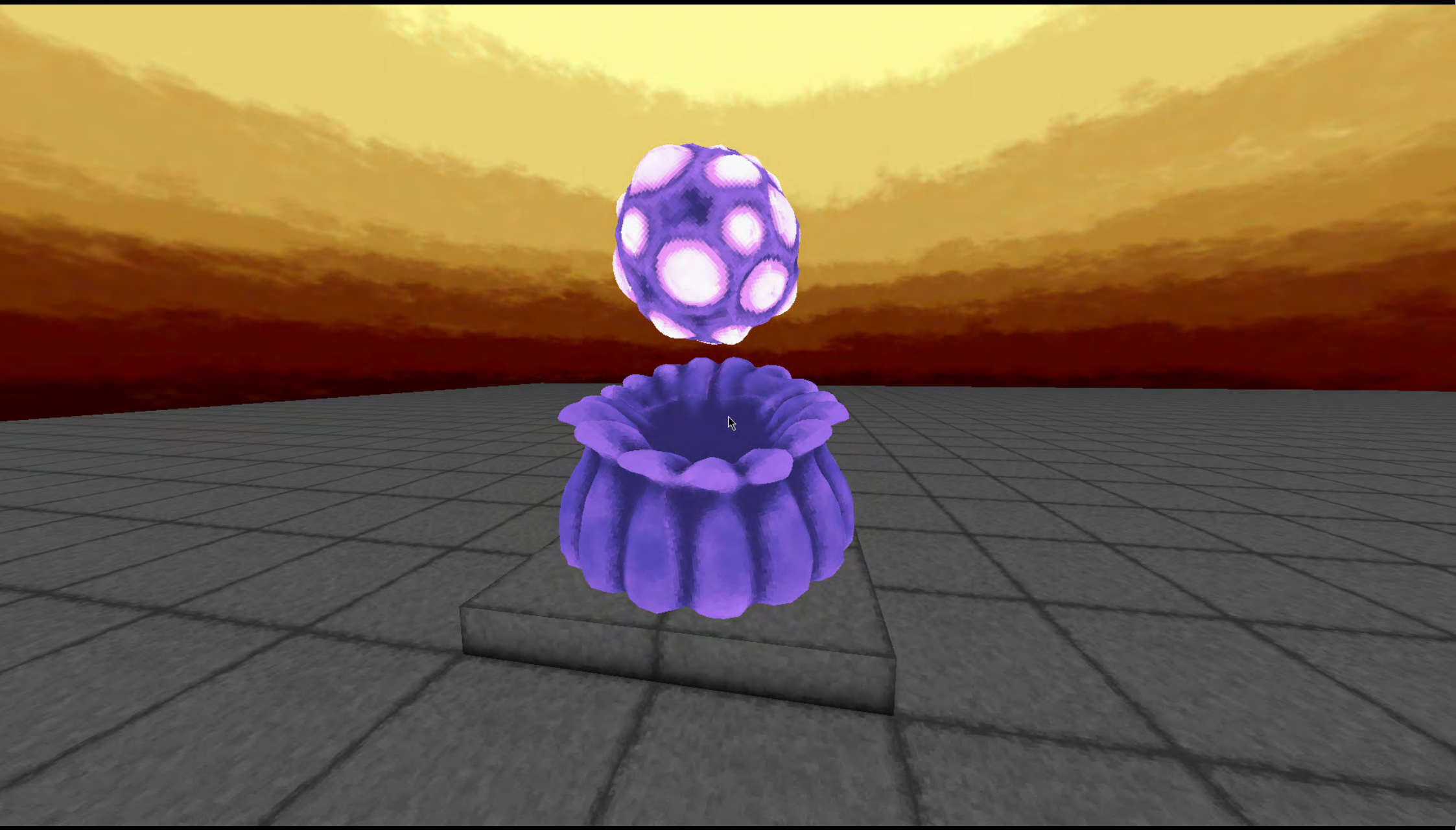


Image Feed

Cuby :

Intention de design :

Le Jumpy Cuby est l'ennemi de base du jeu. Il n'est pas conçu pour représenter une menace complexe individuellement, mais pour :

- Mettre la pression par la quantité
- Créer des vagues de submersion visuelle et spatiale
- Introduire une lecture de rythme simple, mais imprévisible
- Il incarne le danger diffus : peu dangereux seul, mais écrasant en meute.

Fonctionnement :

- Se déplace par sauts réguliers vers la position du joueur
- Chaque saut a une légère variance angulaire, rendant sa trajectoire non parfaitement prévisible
- S'il touche le joueur, il inflige des dégâts de contact

Détail important :
Chaque Jumpy Cuby possède un point faible visuel situé au niveau des yeux, qui n'apparaît qu'après avoir subi une certaine quantité de dégâts.

- En le frappant ensuite au corps à corps, le joueur récupère plus d'énergie.

Impact gameplay

- Crée une pression d'environnement : force le joueur à se repositionner
- Demande d'anticiper les rebonds, pas juste de réagir
- Devient une opportunité d'énergie bonus si bien géré

Variation de challenge

- Leur vitesse de saut et temps d'attente entre deux bonds peuvent varier selon le palier
- Leur quantité influe directement sur la sensation de vague ou d'étouffement
- Le placement spatial (proximité, regroupement) les rend plus ou moins menaçants

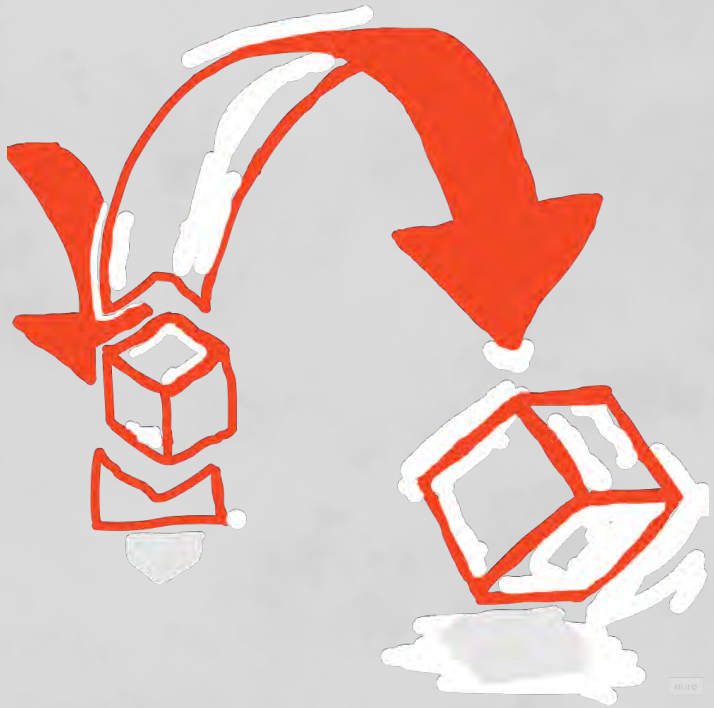


Schéma fonctionnement Cuby

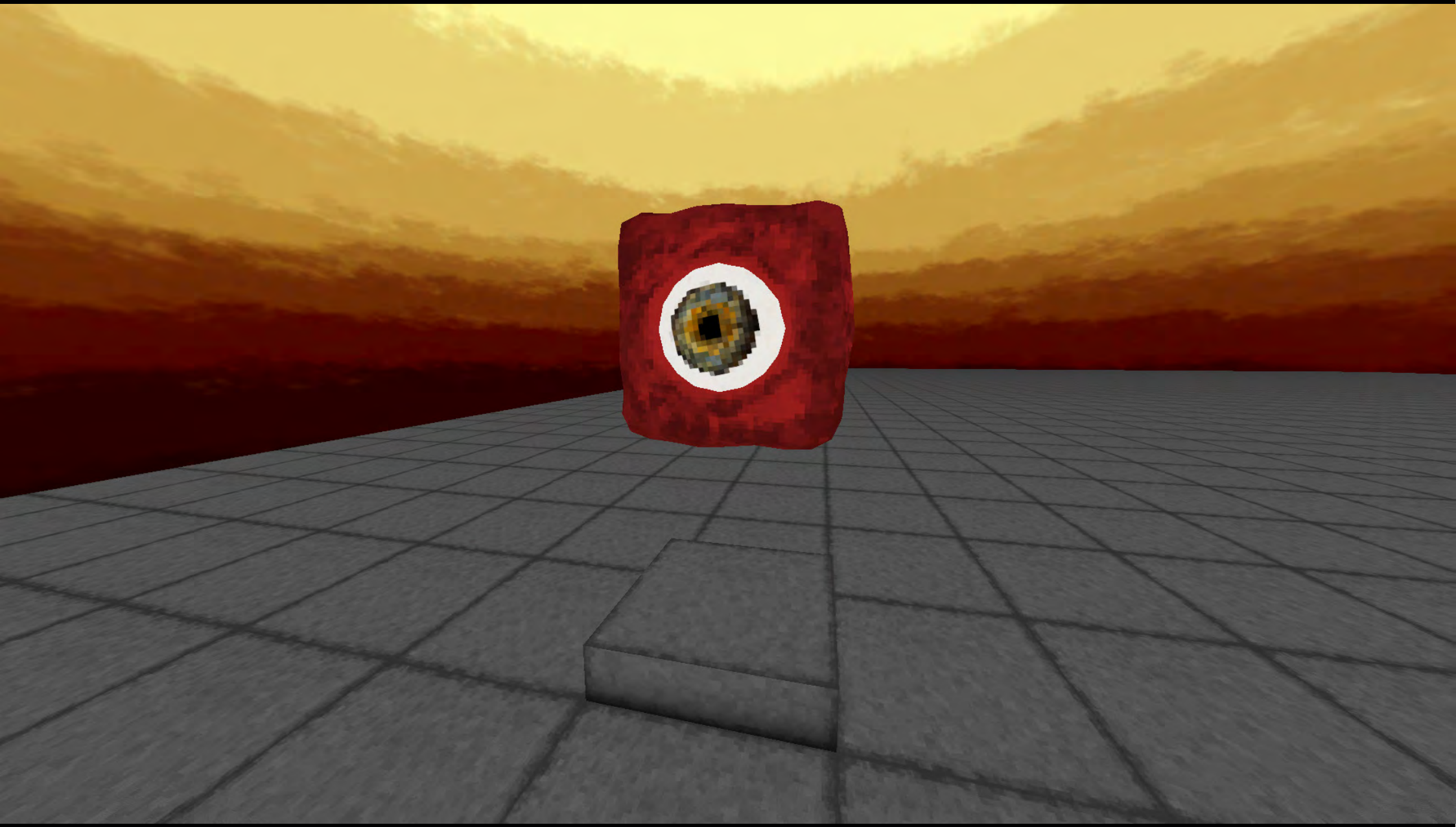


Image Cuby

Dashoot :

Intention de design :

LeDashoot est un ennemi pensé pour perturber la stabilité du joueur :
- Il force à viser rapidement, avec précision
- Il pousse au déplacement, empêchant toute position statique
- Il utilise la téléportation et le tir à distance pour harceler sans relâche

Il est conçu pour briser la stabilité du joueur, en l'obligeant à se repositionner constamment sous la menace de tirs à distance.

Fonctionnement :

Toutes les X secondes au sol, il vérifie la distance entre lui et le joueur :
- Pour premièrement, se téléporter :
    - Si le joueur est à portée, il cherche dans une grille autour de lui un point valide, lui permettant de rester à une distance raisonnable du joueur :
        - Trop proche : s'éloigner
        - Trop loin : se rapprocher
    - Bonne distance : se déplace vers la gauche/droite
    - Si le joueur est hors de portée, il cherche autour du joueur un point où il pourra se téléporter pour être à portée du joueur sans être trop proche ni trop loin.
- Si le joueur est à portée, il traque le joueur et il charge un laser pendant une durée X. Tire le laser s'il est chargé, puis vérifie l'objet touché par le laser. Si l'objet correspond au joueur, lui inflige des dégâts.
- Il ne peut tirer qu'un laser par intervalle de téléportation.

Variables :

- Temps entre chaque TP
- Temps de charge du tir
- Taille de la zone de téléportation
- Cadence des attaques

Point faible :

Crital en haut

Impact gameplay :

- Pousse à bouger constamment
- Introduit une menace à distance dès les premiers paliers
- Devient un ennemi prioritaire dans les combats de groupe
- Sa présence oblige à gérer l'espace et la ligne de vue

Variation de challenge

La difficulté varie selon :
- Le rythme des téléportations
- La précision du laser
- La vitesse d'enchaînement entre TP et tir
- Peut créer des zones de non-droit si plusieurs sont actifs en même temps

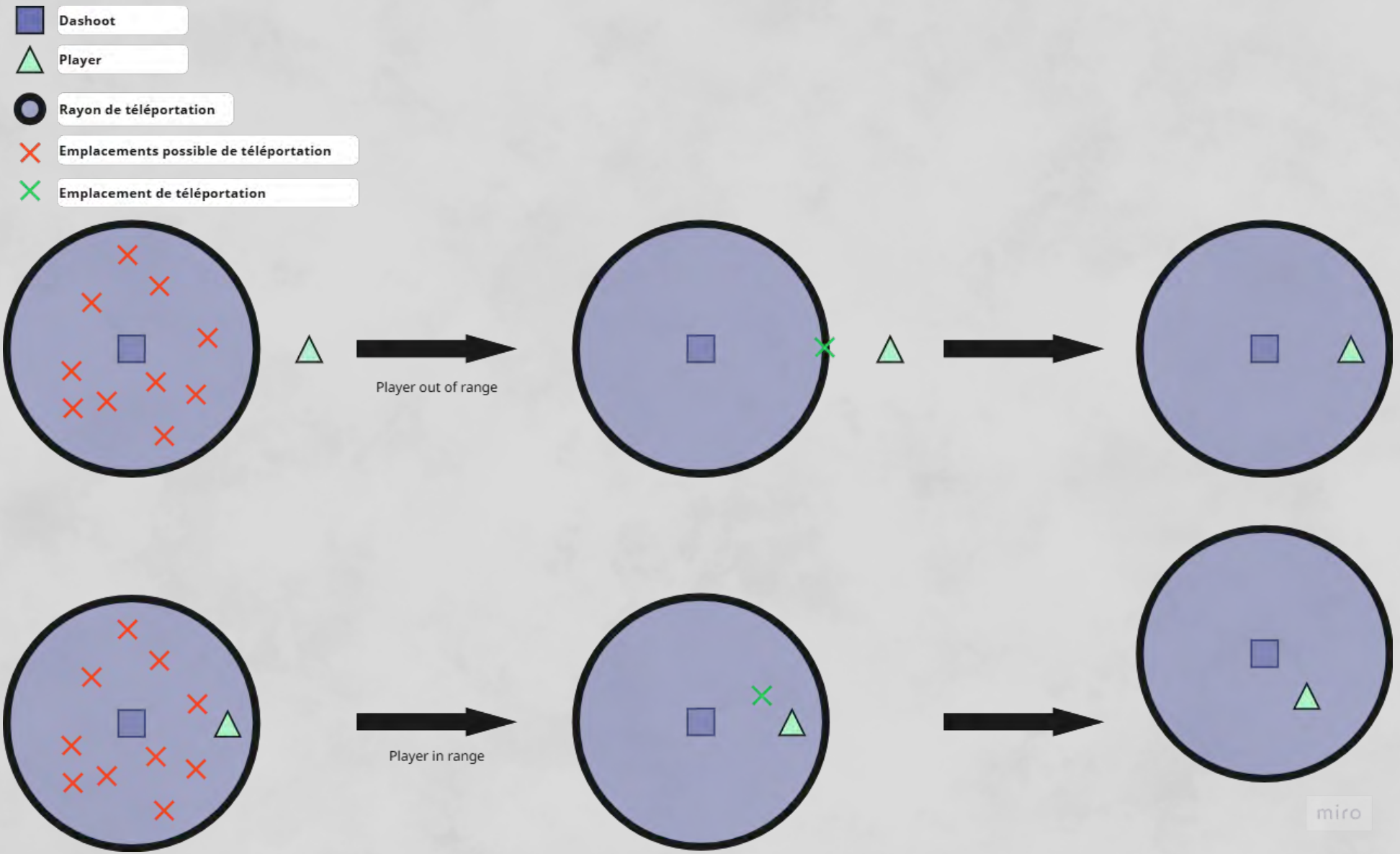
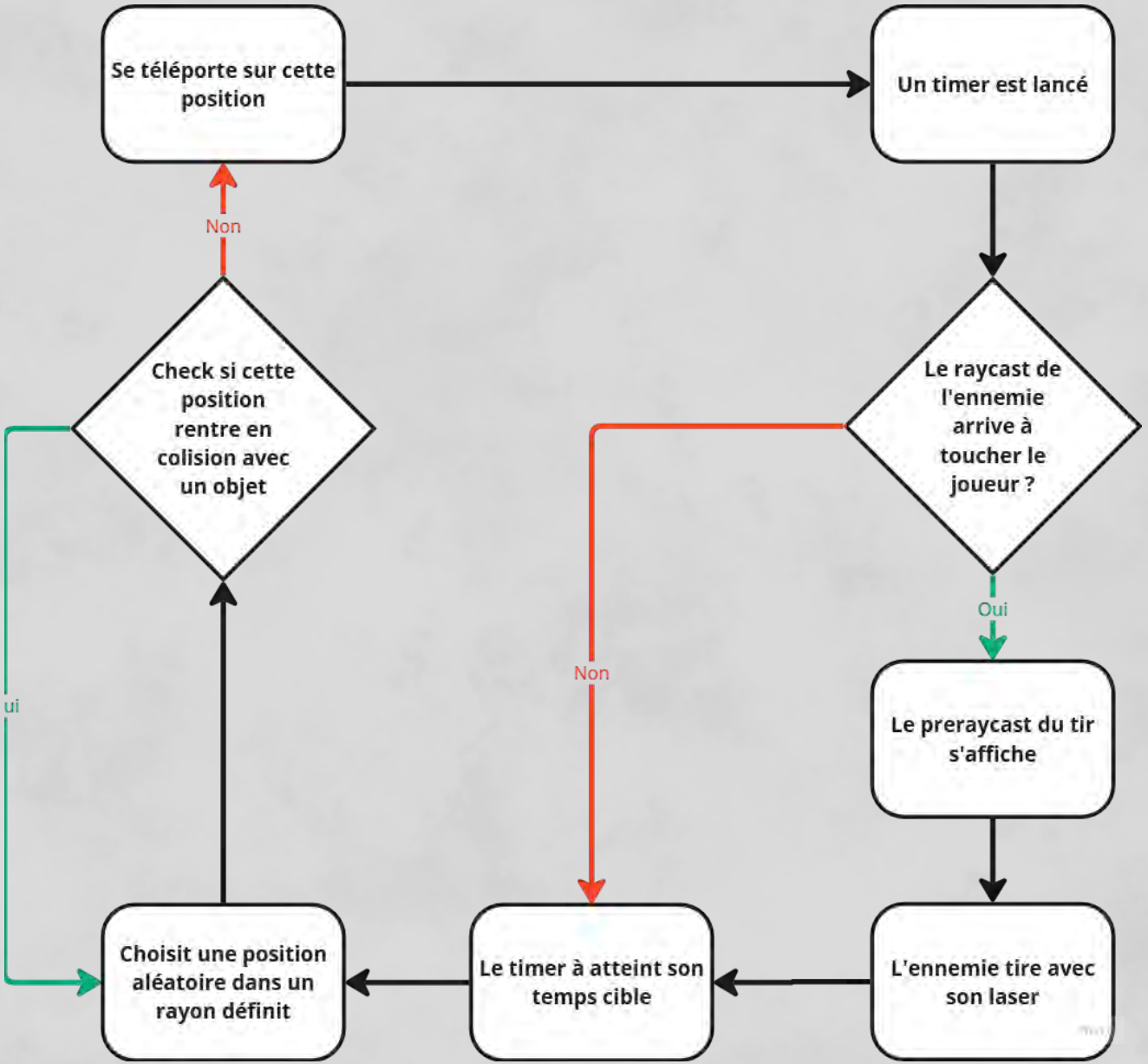


Schéma fonctionnement Dashoot



Flow Chart fonctionnement Dashoot

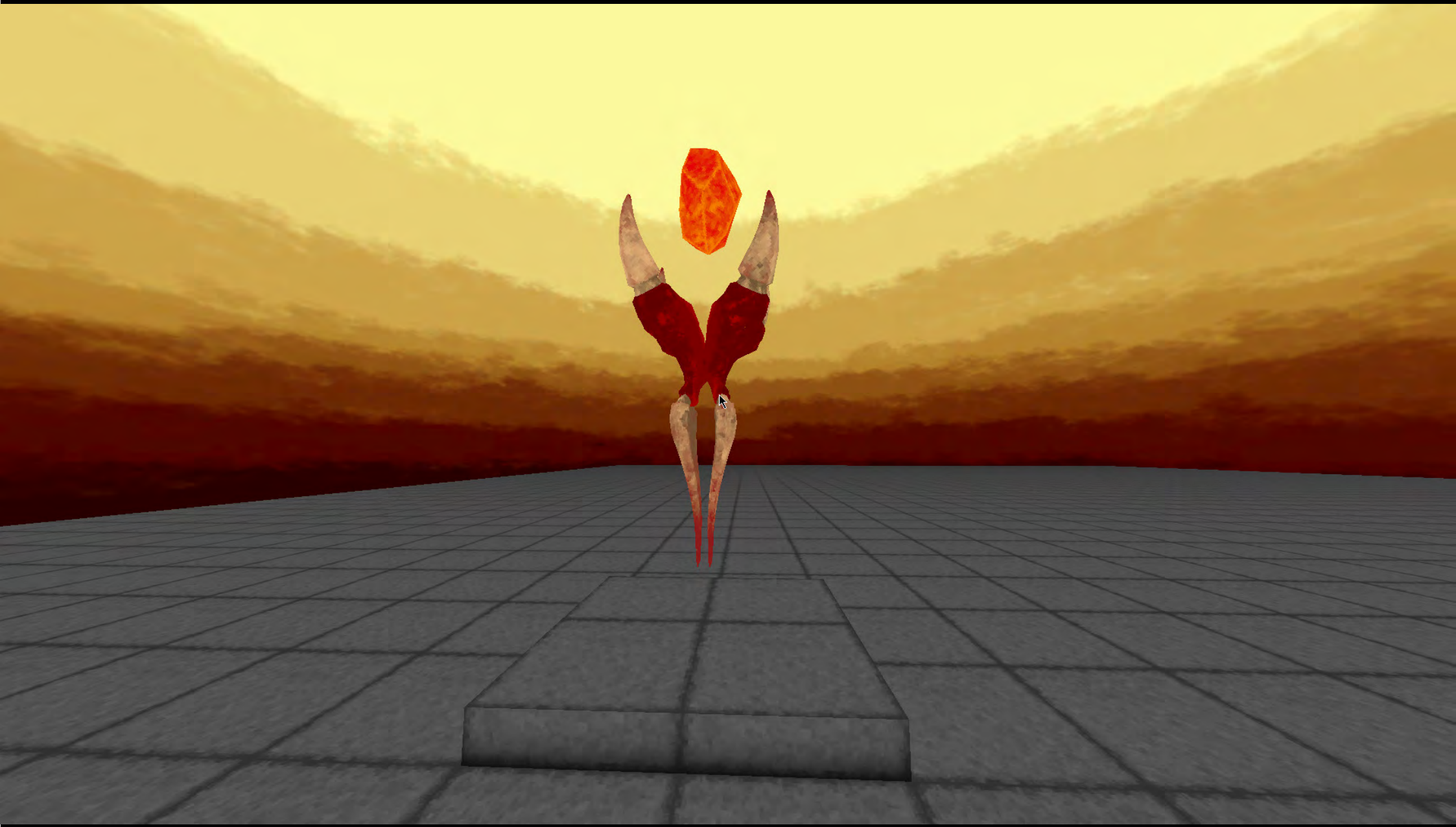


Image Dashoot

Banshee :

Intention de design :

Un ennemi volant forçant le joueur à lever la tête, donne de l'intérêt aux mécaniques aériennes du joueur et remplit le ciel.

Son rôle est double :
Une petite entité volante facile à tuer mais se déplace en horde. Il attaque le joueur en lui fonçant dessus pour s'enfuir à grande vitesse derrière.

Fonctionnement :

Il vérifie la distance entre lui et le joueur :
Si le joueur est à portée, il attaque le joueur et lui inflige des dégâts. Après cela, il va s'enfuir à grande vitesse pour s'éloigner du joueur.
Si le joueur est hors de portée, il fonce tout droit en direction du joueur jusqu'à être à portée du joueur. S'il y a un obstacle, il essaie de l'éviter.

Point faible :

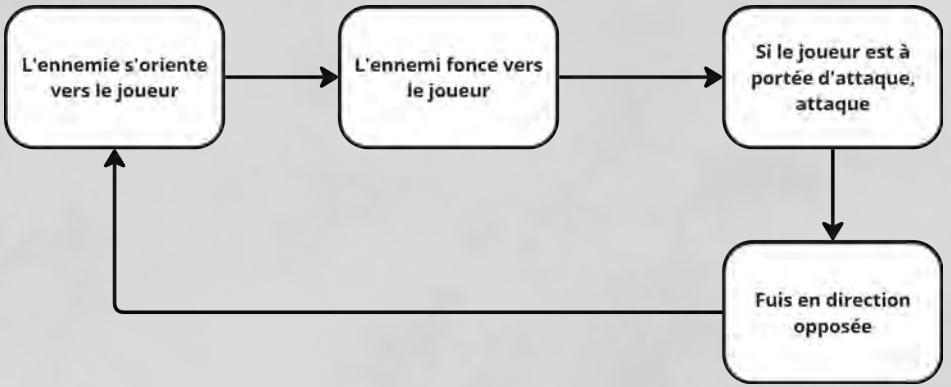
Oeil Banshee

Impact gameplay :

- Empêche le joueur de dominer les hauteurs sans conséquences
- Introduit une menace à effet de zone, différente du tir direct
- Encourage à gérer le timing des sauts et attaques
- Peut piéger le joueur au sol s'il ne bouge pas assez vite

Variation de challenge :

- Rapidité de déplacement dans les airs
- Difficuté de tir lié à la taille des banshees
- Nombre d'ennemi



FlowChart fonctionnement Banshee

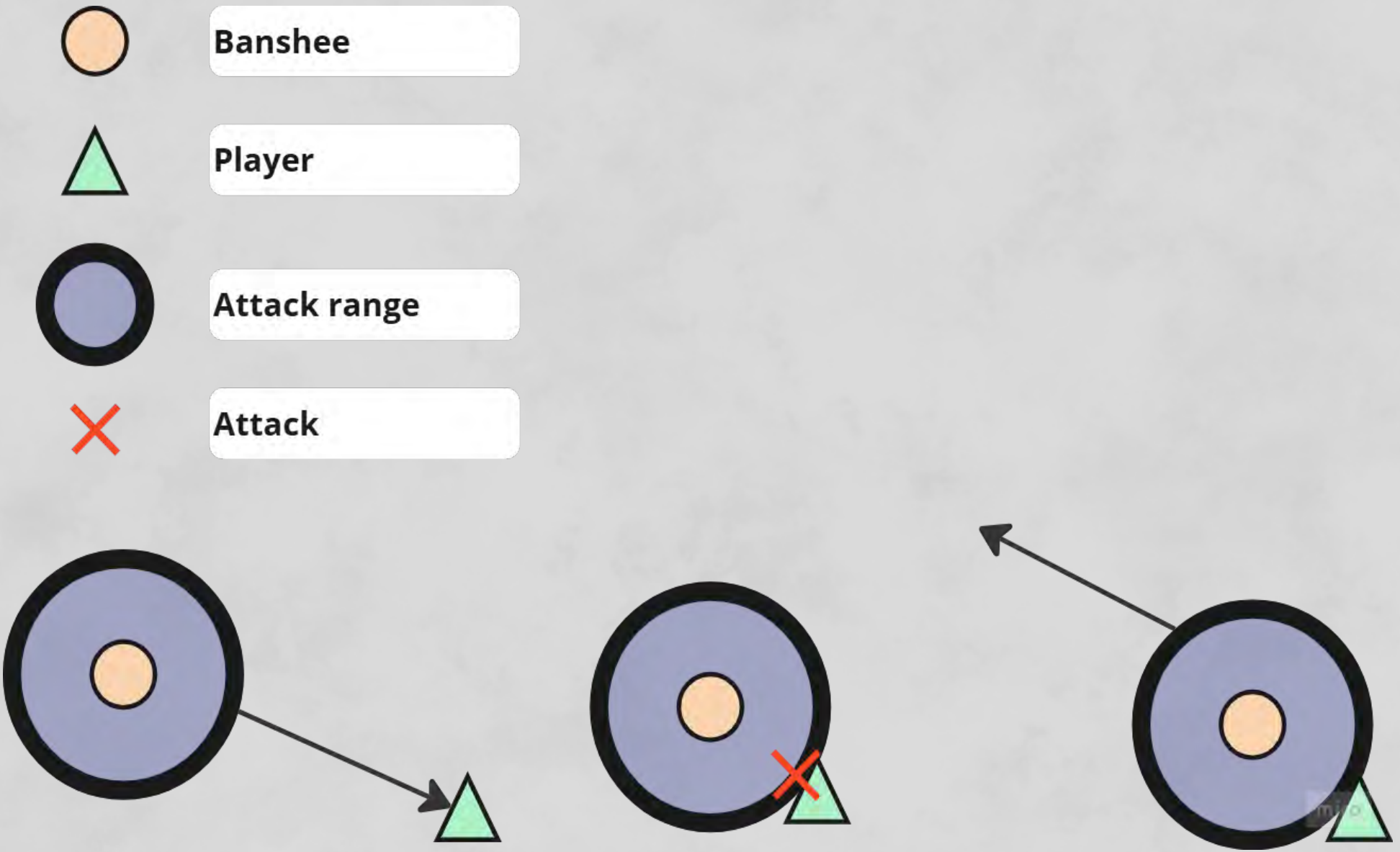


Schéma fonctionnement Banshee

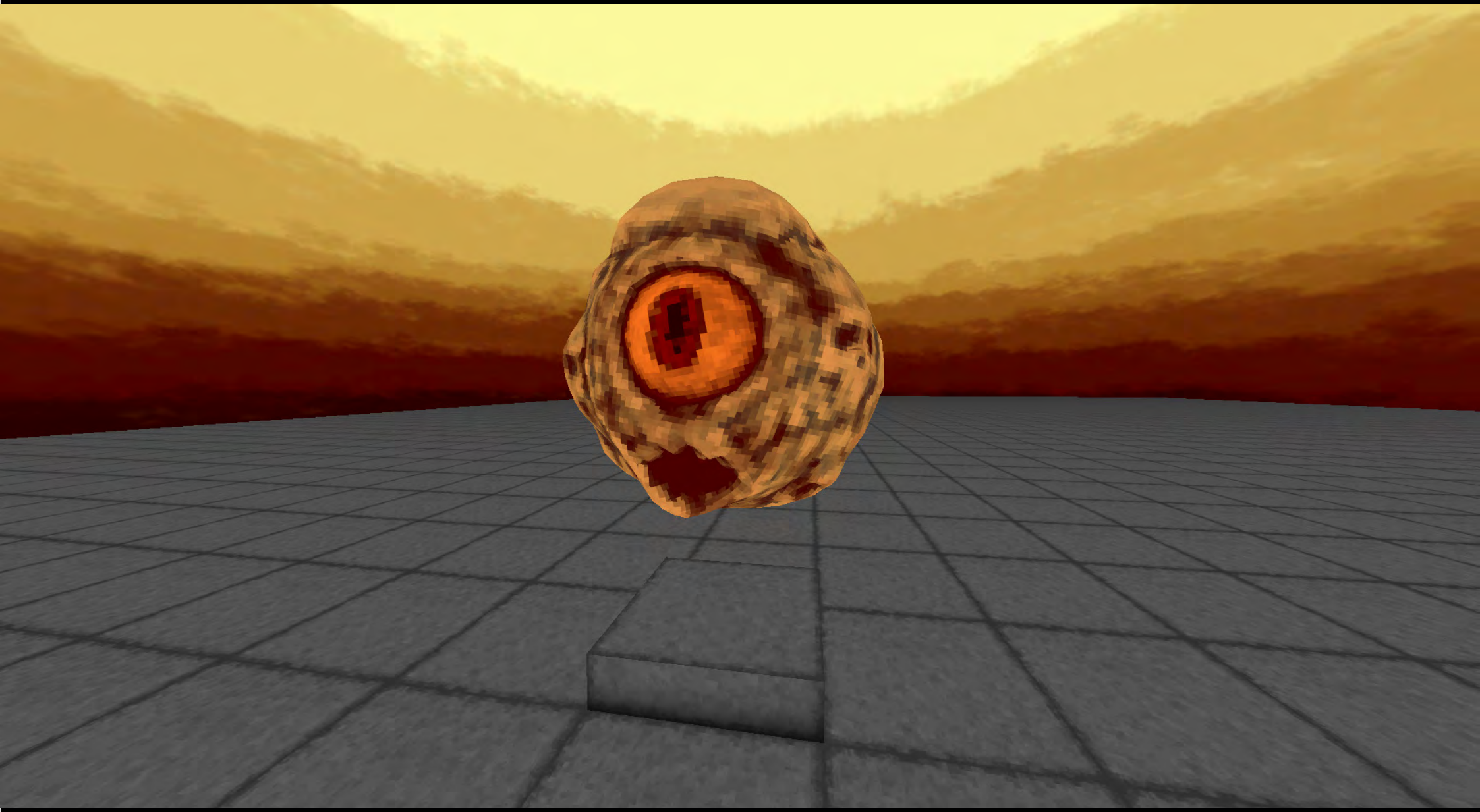


Image Banshee

OunOun :

Intention de design :

Le OunOun est un ennemi massif et lent, conçu pour représenter une menace prioritaire.

Il est pensé pour :
- Saturer l'espace visuel et tactique du joueur
- Imposer une prise de décision rapide : fuir, contourner ou affronter
- Offrir un risque maximal contre une récompense énergétique élevée

C'est un boss miniature : pas par son statut, mais par son poids stratégique.

Fonctionnement

- Se déplace lentement, mais traque constamment le joueur
- Tire à haute fréquence des projectiles vers le joueur dès qu'il est en vue
- Si le joueur n'est plus visible, il avance vers la dernière zone d'interaction connue
- Possède beaucoup de points de vie

Point faible :

Dent du haut

Impact gameplay

- Demande au joueur de faire un choix stratégique immédiat (l'affronter frontalement ? l'éviter ? créer une diversion ?)
- Offre un gain élevé mais nécessite une prise de risque importante
- Favorise des usages comme :
- Le pilonnage pour le surprendre par le haut
- Le contournement via level design

Variation de challenge

- Sa présence unique dans l'arène est suffisante pour modifier le rythme du combat
- Devient de plus en plus dangereux à chaque palier :
- Plus de vie
- Tirs plus fréquents
- Dégâts accrus

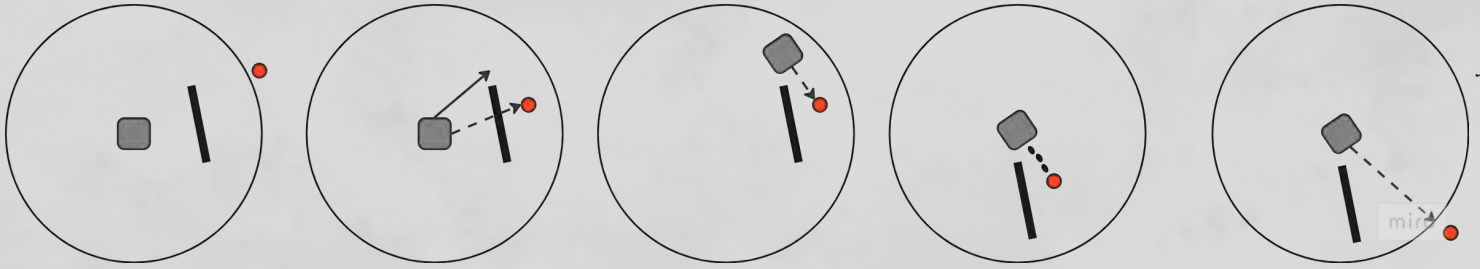
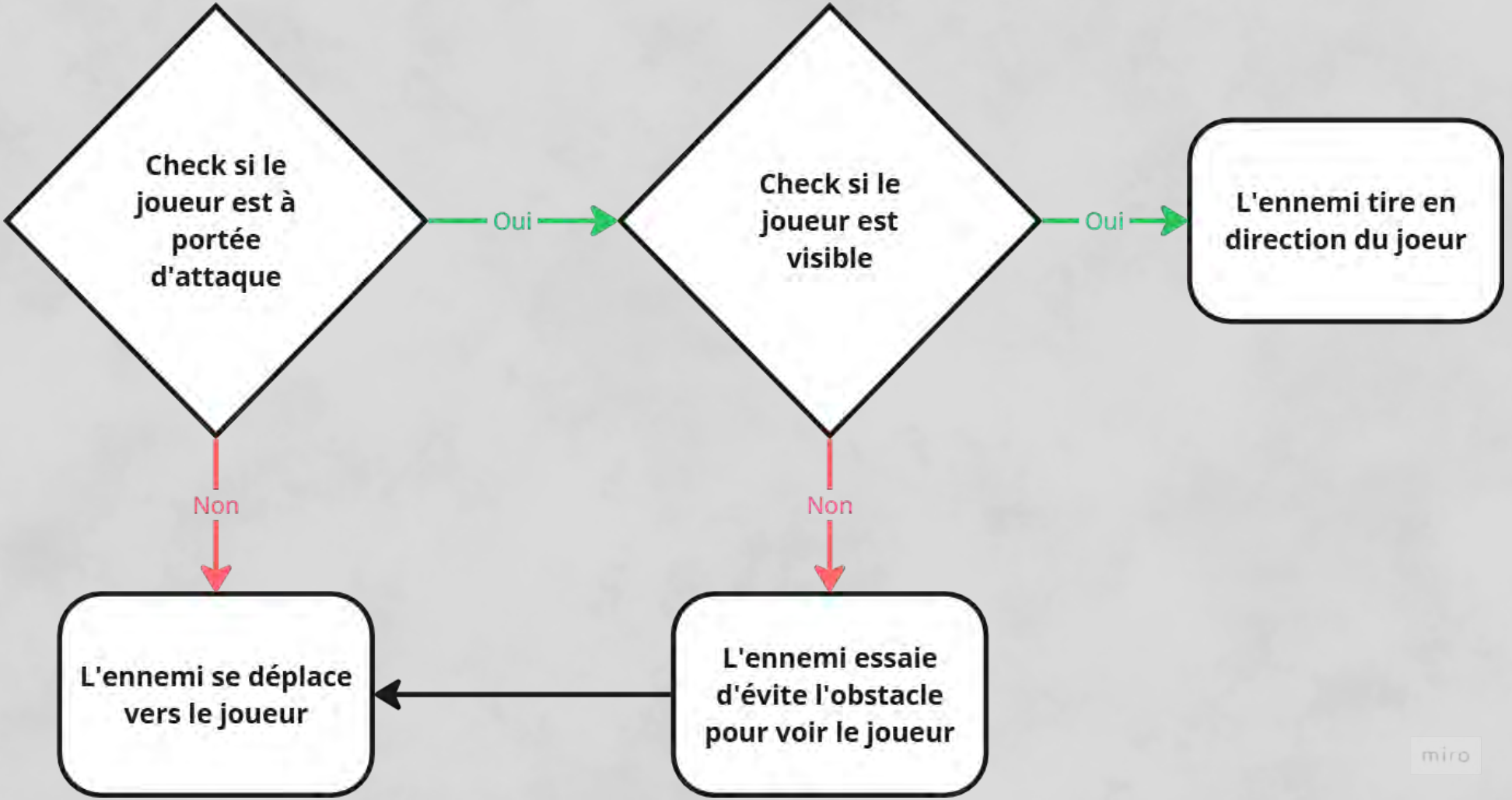


Schéma fonctionnement OunOun



Flow Chart fonctionnement OunOun

Image OunOun

Boss :

Intention :

Ce boss a été conçu dans le but d'incarner un ennemi oppressant, capable de pousser le joueur à exploiter pleinement les capacités de son personnage. Le déroulement du combat contre ce boss se distingue également du reste du jeu : une fois la phase de boss engagée, seuls des monstres de haut niveau apparaîtront afin de permettre au joueur de régénérer son énergie. Cette énergie pourra ensuite être convertie en dégâts et entièrement déversée sur le boss, créant ainsi une boucle de gameplay.

Les mécaniques du boss invitent également le joueur à alterner constamment entre proximité et éloignement pour maximiser ses dégâts. Les sections suivantes détailleront comment ces intentions ont été concrètement mises en œuvre.

Mécaniques :

Les mécaniques du boss se divisent principalement en deux catégories :

La première est la mécanique des lasers. Lors de sa première attaque, le boss tire dix lasers simultanément. L'un d'entre eux, choisi au hasard, commencera après un court délai à accélérer et à traquer le joueur. Au bout de quelques secondes, il atteindra sa vitesse maximale prédéfinie. Pour mieux esquiver cette traque, le joueur devra s'approcher à une distance raisonnable du boss : en effet, plus il est éloigné, plus l'angle de poursuite rend l'esquive difficile, rendant la fuite beaucoup moins efficace.

La seconde est une mécanique d'AOE. Celle-ci se déclenche lorsque le joueur est trop proche du boss. Le boss commence alors à charger une attaque de zone, avec un effet visuel de mise en garde. Une fois la charge terminée, il libère une zone de dégâts continus très puissants, obligeant le joueur à s'éloigner rapidement pour éviter une élimination brutale.

Ainsi, le joueur est constamment invité à alterner entre l'approche et la fuite, en observant attentivement les trajectoires des lasers et en cherchant des ouvertures pour infliger des dégâts importants via des attaques de mêlée ou des attaques plongeantes. Durant cette phase, des ennemis supplémentaires, Dashooter lv2 et Ounoun, apparaîtront également pour forcer le joueur à rester en mouvement, rendant la gestion de l'espace encore plus stratégique.

Image Boss

Image Boss qui se prépare à tirer

Image Boss qui tire ses lazer

Image Boss qui se prépare à faire son AOE

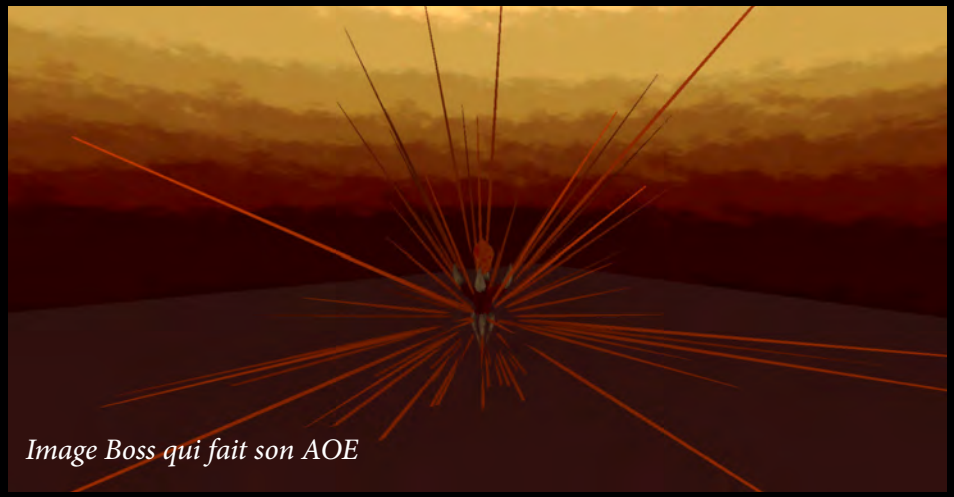


Image Boss qui fait son AOE

Balancing

Réflexion sur l'équilibrage des niveaux d'énergie et du rythme de jeu :

Le système de progression énergétique du joueur est structuré en plusieurs paliers, chacun influençant directement ses capacités de déplacement, d'attaque et d'interaction avec l'environnement. Cette structuration repose sur une intention de design claire : offrir une montée en puissance accessible au début, puis accroître progressivement les contraintes stratégiques pour les joueurs avancés.

Objectifs de design :

Niveaux 1 et 2 : ils sont conçus pour favoriser l'accessibilité. La majorité des joueurs doivent pouvoir y évoluer facilement, avec une sensation de montée en puissance rapide. Cela permet une prise en main fluide, sans punition précoce.,

Niveaux 3 et 4 : ces paliers introduisent une gestion plus serrée de l'énergie. Le joueur doit maximiser son efficacité de combat tout en contrôlant ses ressources. Cela crée une tension constante entre action et survie, qui devient le cœur du gameplay à haut niveau.

Mécanique de rattrapage :

Afin d'éviter une frustration excessive en cas de chute de niveau, le système introduit un mécanisme de récupération :

Un joueur temporairement affaibli peut viser les points faibles d'ennemis de niveau supérieur pour obtenir des orbes d'énergie plus puissants, lui permettant de remonter plus rapidement.

Ce mécanisme introduit une dimension de risque/récompense, et maintient une dynamique où le joueur garde une chance réelle de revenir dans la partie, même après une erreur.

Défi d'équilibrage :

Le principal enjeu d'équilibrage réside dans la recherche d'un point d'inflexion satisfaisant :
- Si la progression est trop linéaire, la tension disparaît.,
- Si la chute de niveau est trop punitive, le joueur abandonne.,
- Si la récupération est trop facile, la notion de gestion perd son intérêt.,

Nous cherchons donc un juste milieu où :
- La montée reste gratifiante mais pas triviale,
-La pression énergétique rend chaque choix significatif,
Et le système de comeback reste rare mais maîtrisable par les joueurs attentifs.

Au fil des essais, nous avons progressivement identifié plusieurs points d'ancrage dans la courbe de difficulté : des moments où l'écart entre deux niveaux d'énergie produisait soit une montée trop abrupte, soit un plateau sans impact. Ces observations ont permis d'isoler des variables-clés à ajuster, telles que :
- les seuils d'énergie pour changer de niveau,
- les coûts énergétiques des compétences,
- les fréquences et puissances des orbes lâchés par les ennemis selon leur type et leur niveau.

Chaque itération du système a fait l'objet de tests ciblés, en session de jeu interne ou en simulation, afin de mesurer les effets des ajustements sur la fluidité, la tension et la capacité de récupération du joueur. Ce processus itératif nous a permis de converger vers une structure équilibrée, à la fois lisible pour le joueur et fiable pour le designer.

			Lv.1	Lv.2	Lv.3	Lv.4
Player mobility	Movement	Speed	10	15	20	24
		Consumption per sec	1.5	40	400	1000
		Movable Time at max energy (sec)	2000	375	125	100
	Sprint	Sprint Speed	0	45	60	80
		Consumption per sec	0	250	800	4500
		Sprint Damage per sec	0	65	300	2000
		Movable Time at max energy (sec)	0	60	63	22
		Drop bonus	0	0	0	-10
		Movable Time at 10% energy (sec)	0	6	6	2
	Jump	Jump Height	8	18	25	35
		Max Jump count	1	1	2	3
		Vortex Range	0	30	30	60
		Consumption per jump	20	100	200	450

Par ailleurs, nous avons mis en place pour chaque type d'ennemi une table de correspondance avec les capacités du joueur, afin de définir des profils de combat cohérents. Chaque tableau spécifie notamment, par exemple :
- le nombre moyen de tirs requis pour éliminer l'ennemi,
- le temps estimé pour effectuer ce kill en condition standard,
- la quantité d'énergie que l'ennemi rapporte à sa mort.

Ces données ont été calibrées autour d'un indicateur empirique simple :
- un temps de kill supérieur à 0,3 seconde tend à rendre l'ennemi visiblement plus résistant, tandis qu'un kill en-dessous de ce seuil est généralement perçu comme un ennemi «one-shot» (léger, éliminable instantanément).

Ce référentiel nous a permis de contrôler de manière fine le ressenti du joueur, en jouant sur des micro-ajustements de points de vie, cadence des tirs ennemis, ou fréquence de tir du joueur. En parallèle, nous avons comparé ces valeurs avec le gain énergétique associé, afin de maintenir un équilibre entre risque (temps, dégâts subis) et récompense (récupération ou montée de niveau).

Ce travail de granularité garantit que chaque type d'ennemi occupe une fonction précise dans la boucle de jeu, et que leur présence contribue à la tension globale sans provoquer de déséquilibre brutal.

Punch	Attack Range	2	2	2	2
	Attack Distance	999	999	999	999
	Damage	10	50	160	12000
	Cooldown	0.7	0.7	0.7	0.7
	Dash Duration	0.1	0.1	0.1	0.1
	Dps	14	71	229	17143
	Hits to Expose Weakpoint	-2	0	1	1
	Consumption per punch	10	200	500	1000
	Punch count (Max Energy)	300	75	100	100
	Drop bonus	35	45	45	45
	Weak Point Drop bonus	65	65	80	80
Ground Pound	Max sphere range	0	0	100	500
	Damage	0	0	150	24000
	Consumption	0	0	2000	9000
	Usage Count	0	0	18	6
	Drop bonus	0	0	-10	-7
	Enemy In zone	0	0	10	20

	Jumpy Cuby			
	Diff 1	Diff 2	Diff 3	Diff 4
HP	40	80	200	500
Energy Sphere	EnergyPoint lv1	EnergyPoint lv2	EnergyPoint lv3	EnergyPoint lv4
Sphere Drop Quantity	8	13	15	15
Sphere Drop Quantity per hit WeakPoint	10	8	10	15
Hits to Kill	2	2	3	3
Time to Kill	0.40	0.30	0.24	0.13
Hits to Kill with Weakpoint	6	9	12	2
Energy sphere Drop with weakpoint	68	85	135	45
Final Gain Energy without weakpoint	21	178	460	*-200*
Enemy to kill for next level With shoot	143	85	109	*-500*
Final Gain Energy with weakpoint	121	1225	2820	1400
Enemy to kill for next level With shoot weakPoint	25	13	18	72
Final Gain Energy With Punch	136	1048	2160	2800
Enemy to kill for next level With Punch	23	15	24	36
Final Gain Energy With Sprint		-100	-113	-925
Final Gain Energy With GroundPound			-2600	-2600
Damage	30	200	300	800
Downgrade Requirement with current level (Enemy Hits)	100	75	167	125
Jump Force (Height)	~~3~~	~~5~~	~~15~~	~~15~~
Tracking Force (Move Distance)	~~12~~	~~14~~	~~15~~	~~8~~

	Banshee			
	Diff 1		Diff 2	
HP	~~40~~	40	150	150
Energy Sphere	~~EnergyPoint lv2~~	EnergyPoint lv2	EnergyPoint lv3	EnergyPoint lv4
Sphere Drop Quantity	~~4~~	4	5	5
Sphere Drop Quantity per hit WeakPoint	~~3~~	2	2	2
Hits to Kill	~~2~~	1	2	1
Time to Kill	~~0.40~~	0.15	0.18	0.04
Hits to Kill with Weakpoint	~~6~~	5	9	1
Energy sphere Drop with weakpoint	~~22~~	14	23	7
Final Gain Energy without weakpoint	~~13~~	49	260	*-200*
Enemy to kill for next level With shoot	~~231~~	307	193	*-500*
Final Gain Energy with weakpoint	~~29~~	149	*-76*	*-4*
Enemy to kill for next level With shoot weakPoint	~~104~~	101	*-657*	*-25000*
Final Gain Energy With Punch	~~128~~	904	1880	1380
Enemy to kill for next level With Punch	~~24~~	17	27	73
Final Gain Energy With Sprint		-90	-260	-477.5
Final Gain Energy With GroundPound			-3400	-10120
Damage	100	100	300	300
Downgrade Requirement with current level (Enemy Hits)	30	150	167	334
Jump Force (Height)	~~3~~	~~5~~	~~15~~	~~15~~
Tracking Force (Move Distance)	~~12~~	~~14~~	~~15~~	~~8~~

	Ounoun			
	Diff 4			
HP	~~40~~	~~100~~	~~150~~	3000
Energy Sphere	~~EnergyPoint lv2~~	~~EnergyPoint lv2~~	~~EnergyPoint lv3~~	EnergyPoint lv4
Sphere Drop Quantity	~~4~~	~~12~~	~~4~~	80
Sphere Drop Quantity per hit WeakPoint	~~3~~	~~6~~	~~3~~	30
Hits to Kill	~~2~~	~~3~~	~~2~~	15
Time to Kill	~~0.40~~	~~0.37~~	~~0.18~~	0.75
Hits to Kill with Weakpoint	~~6~~	~~12~~	~~9~~	8
Energy sphere Drop with weakpoint	~~22~~	~~84~~	~~31~~	320
Final Gain Energy without weakpoint	~~13~~	~~147~~	~~232~~	*-900*
Enemy to kill for next level With shoot	~~231~~	~~103~~	~~216~~	*-111*
Final Gain Energy with weakpoint	~~29~~	~~1164~~	~~148~~	7360
Enemy to kill for next level With shoot weakPoint	~~104~~	~~13~~	~~338~~	14
Final Gain Energy With Punch	~~128~~	~~1032~~	~~1852~~	3480
Enemy to kill for next level With Punch	~~24~~	~~15~~	~~27~~	29
Final Gain Energy With Sprint		~~-192.6153846~~	~~-288~~	-4790
Final Gain Energy With GroundPound			~~-3680~~	31880
Damage	~~240~~	~~240~~	~~2000~~	500
Downgrade Requirement with current level (Enemy Hits)	~~13~~	~~63~~	~~25~~	200
Jump Force (Height)	~~3~~	~~5~~	~~15~~	~~15~~
Tracking Force (Move Distance)	~~12~~	~~14~~	~~15~~	~~8~~

	Dashooter			
	Diff 1		Diff 2	
HP	~~40~~	100	~~150~~	2000
Energy Sphere	~~EnergyPoint lv2~~	EnergyPoint lv2	~~EnergyPoint lv3~~	EnergyPoint lv4
Sphere Drop Quantity	~~4~~	20	~~20~~	35
Sphere Drop Quantity per hit WeakPoint	~~3~~	20	~~12~~	35
Hits to Kill	~~2~~	3	~~2~~	10
Time to Kill	~~0.40~~	0.37	~~0.18~~	0.50
Hits to Kill with Weakpoint	~~6~~	12	~~9~~	5
Energy sphere Drop with weakpoint	~~22~~	260	~~128~~	210
Final Gain Energy without weakpoint	~~13~~	275	~~680~~	*-1160*
Enemy to kill for next level With shoot	~~231~~	55	~~74~~	*-86*
Final Gain Energy with weakpoint	~~29~~	3980	~~1328~~	4880
Enemy to kill for next level With shoot weakPoint	~~104~~	4	~~38~~	21
Final Gain Energy With Punch	~~128~~	1160	~~1100~~	2220
Enemy to kill for next level With Punch	~~24~~	13	~~46~~	46
Final Gain Energy With Sprint		-65	~~160~~	-3800
Final Gain Energy With GroundPound			~~800~~	6680
Damage	~~240~~	240	~~2000~~	2000
Downgrade Requirement with current level (Enemy Hits)	~~13~~	63	~~25~~	50
Jump Force (Height)	~~3~~	~~5~~	~~15~~	~~15~~
Tracking Force (Move Distance)	~~12~~	~~14~~	~~15~~	~~8~~

Only with enemy Cuby						Input Value / Play		Shoot	Punch	Punch weakpoint	Ground Pound	Input Value / Play test
Player Current Level	Net Energy Change	Total ECPS	Estimated Average ECPS	ETEL PS	CEE	HitRat e	Average EGPS	Current EGPS	Current EGPS	Current EGPS	Current EGPS	Required KPS
Lv 1	7.5	27.8	27.5	0.3	10	0.10%	35.3	13	34.4	58.4	/	0.5
Lv 2	377.4	597.0	595.0	2	10	0.10%	974.4	312	1113.6	1497.6	/	1.5
Lv 3	2008.0	3158.0	3155.0	3	10	0.10%	5166.0	4200	6048	9576	840	6
Lv 4	-2803.0	15058.0	15050.0	8	10	0.10%	12255.0	7600	13680	21660	6080	19

Ce travail s'est appuyé sur plusieurs versions successives de tableaux d'équilibrage, régulièrement mis à jour à chaque nouvelle itération. Ces documents nous ont permis de :
- visualiser les écarts entre les types d'ennemis,,
- croiser les données de dégât, de temps et de récompense,,
- repérer les incohérences ou les zones mortes dans la courbe de difficulté,,
- et surtout, d'identifier des valeurs "pivot" : des points précis où une simple variation modifiait sensiblement le ressenti joueur.,

Ce processus d'analyse croisée entre ressenti empirique et valeurs chiffrées a été essentiel pour guider les ajustements les plus fins. L'ensemble du système repose ainsi non seulement sur une logique systémique, mais aussi sur un ensemble d'ancres perceptives validées par le test.

Player Shoot

Player Current Level	Required Energy for next level	Required Cuby Dif 1 Kills	Required Cuby Dif 2 Kills	Required Cuby Dif 3 Kills	Required Cuby Dif 4 Kills
Lv 1	3000	154	11	6	4
Lv 2	15000	-385	109	50	40
Lv 3	50000	-113	-447	834	313
Lv 4	100000	-84	-94	-109	-87

Player Shoot WeakPoint

Player Level	Required Energy for next level	Required Cuby Dif 1 Kills	Required Cuby Dif 2 Kills	Required Cuby Dif 3 Kills	Required Cuby Dif 4 Kills
Lv 1	3000	316	12	6	4
Lv 2	15000	-152	455	97	40
Lv 3	50000	-83	-116	-148	313
Lv 4	100000	-84	-94	-109	-87

Player Punch

Player Level	Required Energy for next level	Required Cuby Dif 1 Kills	Required Cuby Dif 2 Kills	Required Cuby Dif 3 Kills	Required Cuby Dif 4 Kills
Lv 1	3000	68	5	3	2
Lv 2	15000	-112	31	18	42
Lv 3	50000	-63	1786	179	1250
Lv 4	100000	-53	-94	-313	228

Player Punch Weak Point

Player Level	Required Energy for next level	Required Cuby Dif 1 Kills	Required Cuby Dif 2 Kills	Required Cuby Dif 3 Kills	Required Cuby Dif 4 Kills
Lv 1	3000	29	3	2	2
Lv 2	15000	-160	19	11	13
Lv 3	50000	-70	86	40	35
Lv 4	100000	-55	-196	152	55

	Energy per use	Energy per sec	Weight	Final ECPS
Move	/	1.5	1	1.5
Shoot	2.5	10	1	10
Punch	10	13	0.8	10.00
Jump	20	20	0.3	6
Total ECPS				27.5

Player Lv 2	Energy per use	Energy per sec	Weight	Final ECPS
Move	/	40	1	40.0
Shoot	15	90	1	90.0
Punch	200	250	0.8	200.0
Jump	100	100	0.4	40.0
Sprint	/	250	0.9	225.0
Total ECPS				595.0

Player Lv 3	Energy per use	Energy per sec	Weight	Final ECPS
Move	/	400	1	400.0
Shoot	80	800	1	800.0
Punch	500	625	0.6	375.0
Jump	200	200	0.7	140.0
Sprint	300	800	0.8	640.0
Ground Pound	2000	2000	0.4	800.0
Total ECPS				3155.0

Player Lv 4	Energy per use	Energy per sec	Weight	Final ECPS
Move	/	1000	1	1000.0
Shoot	200	4000	1	4000.0
Punch	1000	1250	0.3	375.0
Jump	450	450	0.5	225.0
Sprint	/	4500	0.9	4050.0
Ground Pound	9000	9000	0.6	5400.0
Total ECPS				15050.0

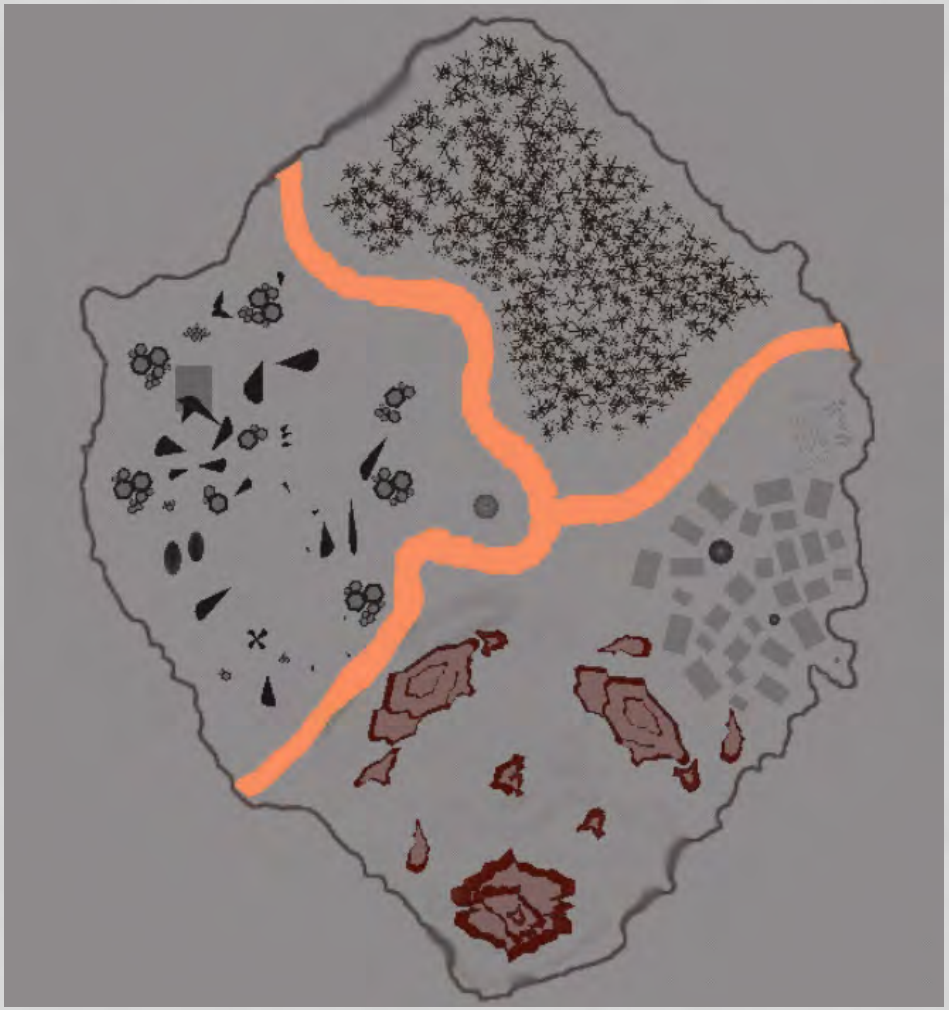
Level Design

Introduction :

Pour notre level design, notre objectif principal était d'accompagner le joueur dans sa montée en puissance. À chaque palier franchi, le level design devait s'ouvrir davantage à lui, l'incitant à en tirer parti pour éliminer ses ennemis le plus rapidement possible, tout en proposant des défis variés. Pour y parvenir, nous sommes passés par plusieurs phases de création et d'expérimentation.

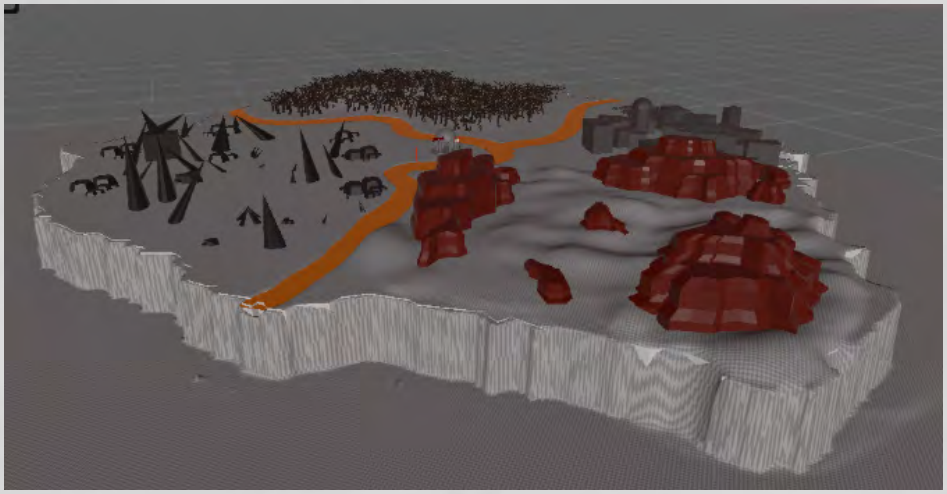
Itération :

<u>Ile Version V1 :</u>



Le LD ile v1 vue de haut.

Pour notre level design, notre première idée était de créer une grande carte composée de plusieurs biomes, chacun proposant au joueur des environnements variés et des défis différents.



Le LD ile v1 vue de loin

Ile Version V2 :

Le premier essai ne nous ayant pas vraiment convaincus, nous avons repensé notre level design sous la forme d'un archipel, en conservant uniquement les biomes qui nous semblaient les plus fun et intéressants à explorer.

Ce level design, bien qu'intéressant, posait des problèmes au niveau des échelles et de l'évolution du joueur à travers les paliers. Comme le joueur gagne en puissance au fil de la partie, ce level design, pourtant gigantesque au début, devenait trop restreint dans les derniers niveaux. Le concept de l'archipel a donc été abandonné, et nous avons décidé d'explorer de nouvelles pistes.

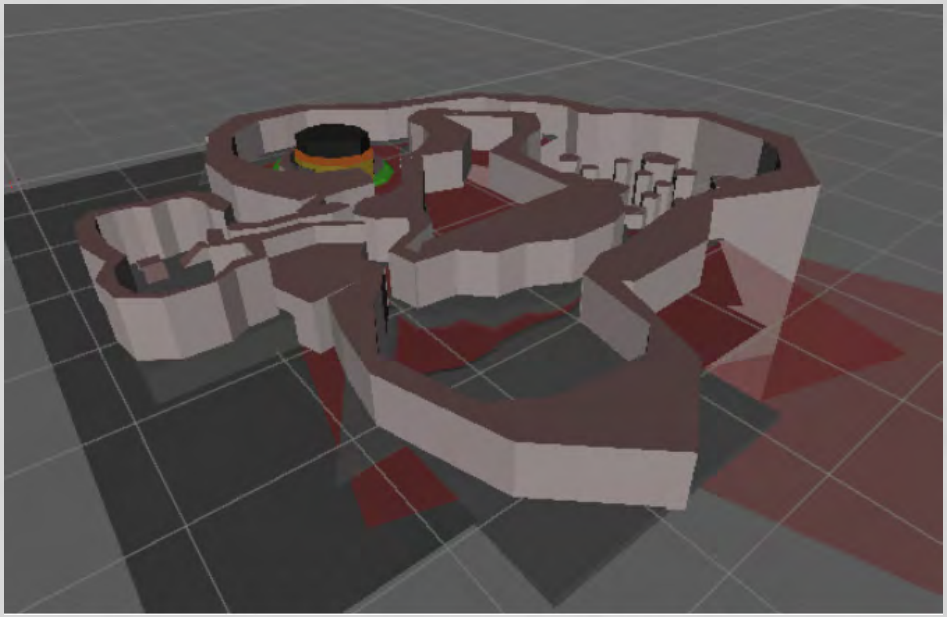
Le LD ile v2 vue de loin

Le LD ile v1 vue de haut

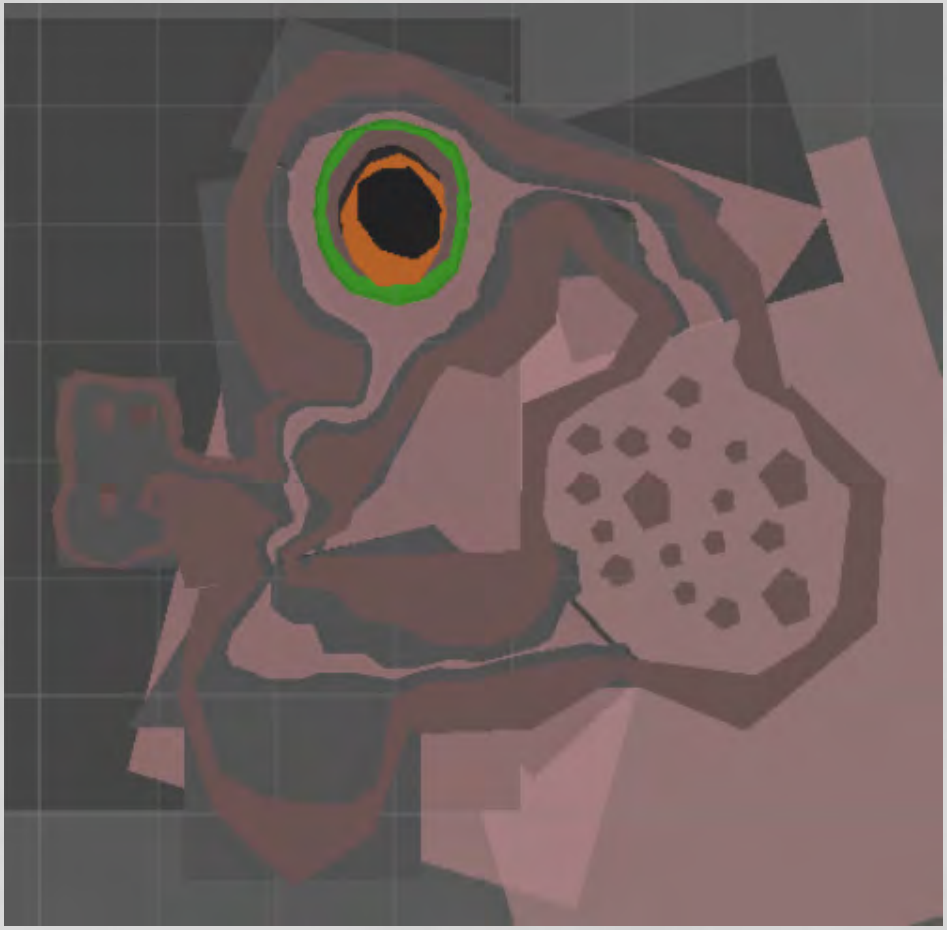
Canyon Version :

Un nouveau level design inspiré du Grand Canyon a été expérimenté. Il proposait de vastes zones ouvertes où le joueur pouvait courir et sauter de manière plus libre. Ce choix permettait d'alterner entre de grands espaces et des couloirs plus étroits, modifiant ainsi le rythme et le gameplay. De plus, une fois le palier 4 atteint, le joueur obtient un accès aux hauteurs du canyon, renforçant le sentiment de progression et d'ascension.

Ce level design a finalement été rejeté car il s'est révélé trop restrictif. Nous avons également constaté une tendance des joueurs à rester dans une seule zone sans se déplacer, ce qui allait à l'encontre de la dynamique de gameplay que nous voulions encourager.



Le LD Canyon vue de loin

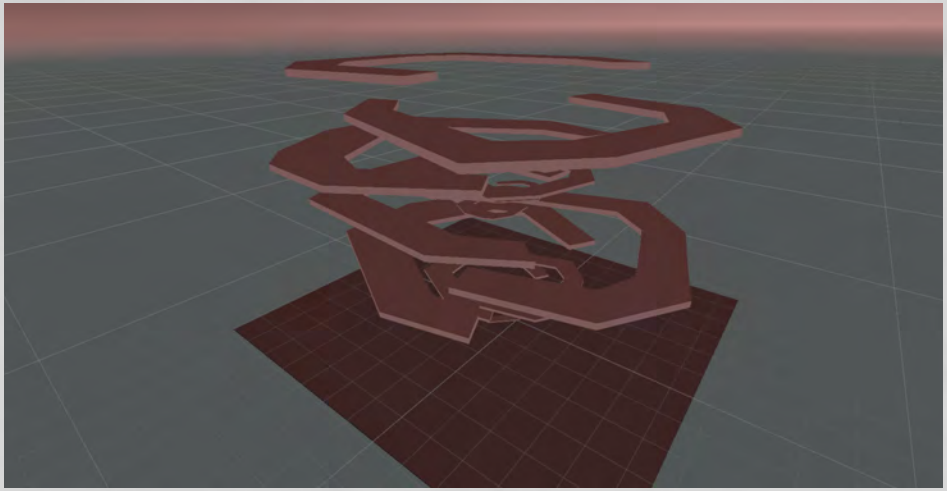


Le LD Canyon vue de loin

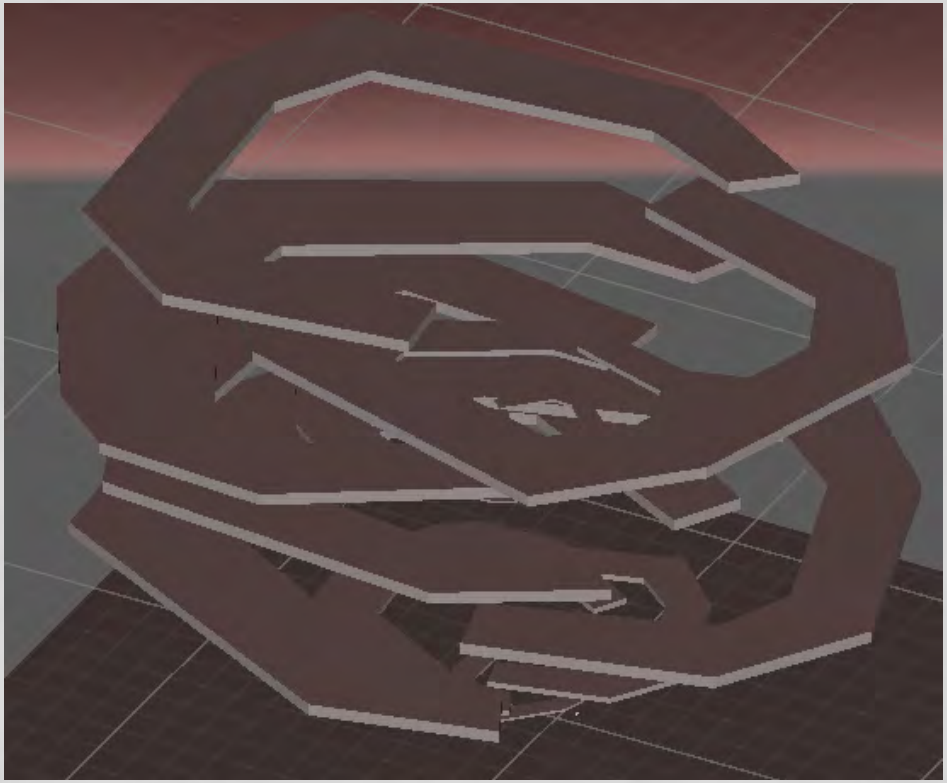
Plateforme aérienne :

À la suite de cette observation, nous avons tenté une nouvelle approche, beaucoup plus verticale, avec un level design basé sur des plateformes aériennes.

Ce level design a été abandonné, car l'idée d'un level design plus plat commençait à s'imposer comme une solution plus adaptée. Le LD vertical posait plusieurs problèmes : nos ennemis n'étaient pas programmés pour évoluer correctement sur des plateformes aériennes, ce qui entraînait de nombreux bugs. De plus, les joueurs avaient tendance à ignorer les plateformes ou à les utiliser très peu. Nous avons donc décidé de tester un level design plus plat.



Plateforme Aérienne LD

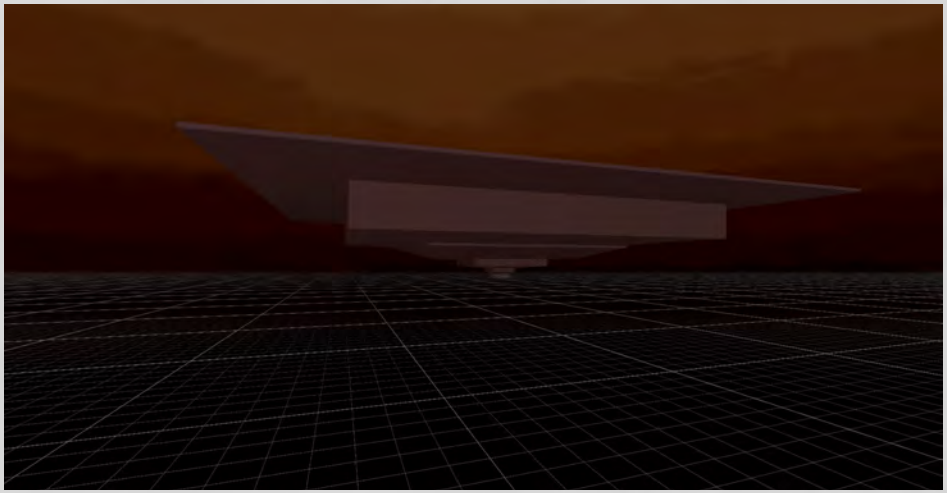


Plateforme Aérienne LD

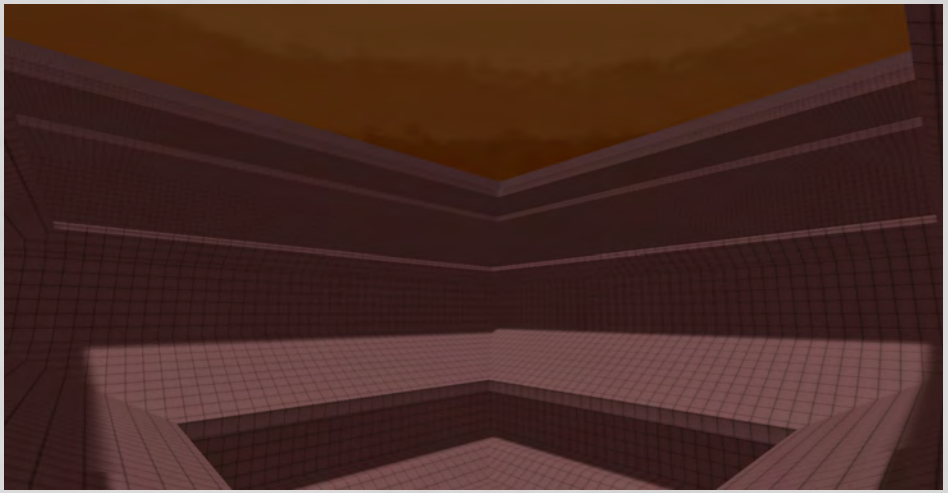
Entonnoir :

Nous avons choisi d'expérimenter un level design en pyramide inversée. Cette approche visait à éviter les problèmes liés aux ennemis qui bug-gent en hauteur : dans le pire des cas, ils ne feraient que s'enfoncer vers le centre, là où le joueur apparaît. Chaque étage nécessitait un palier pour y accéder et proposait un espace de plus en plus vaste pour se mouvoir. Ce système permettait une expérience de jeu rapide et dynamique dans les premiers niveaux, puis plus large et axée sur la compétence du joueur dans les étages inférieurs.

Cependant, c'est en jouant et surtout en faisant tester le jeu que nous nous sommes rendu compte que les joueurs n'utilisaient pas le level design comme prévu. La plupart avaient tendance à rester à un étage offrant suffisamment d'espace pour se déplacer, sans réellement exploiter la verticalité du niveau. Ils se contentaient d'utiliser leur mobilité de base sur un sol plat.



Le LD entonnoir vue de loin



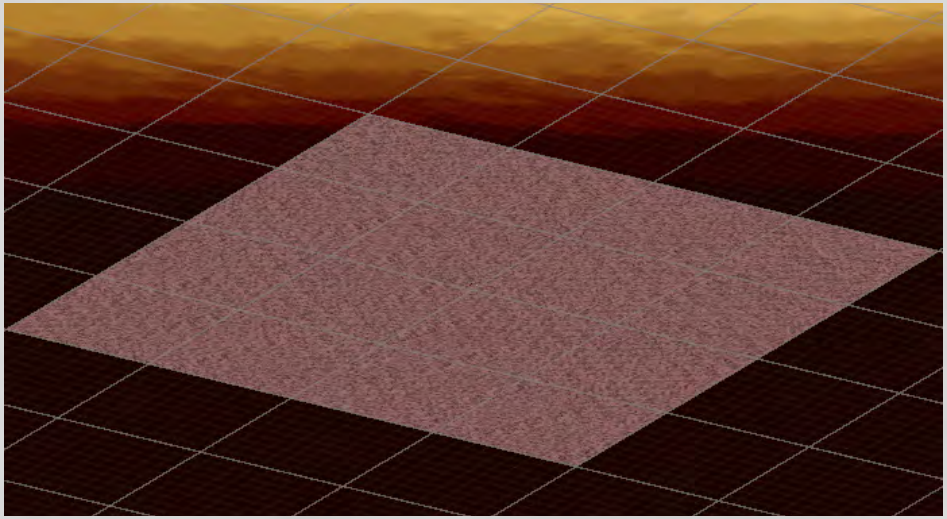
Le LD entonnoir vue d'intérieur

LD Plat :

C’est suite à ces retours de playtests que l’idée d’un level design entièrement plat et sans verticalité a commencé à émerger. Nous avons alors testé cette nouvelle configuration et observé les comportements des joueurs dans cet environnement.

Le level design plat, bien que très simple en apparence, a permis d’apporter une véritable profondeur aux mécaniques de jeu. Là où les anciens niveaux imposaient au joueur d’escalader pour prendre de la hauteur, ce nouveau design l’encourage à exploiter pleinement ses compétences, comme les sauts ou les attaques au corps à corps, pour se déplacer avec agilité.

Contre toute attente, ce level design a renforcé la sensation de mobilité, plutôt que de la brider. De plus, la forte visibilité offerte par cet environnement accentue la sensation de submersion par les ennemis : les joueurs peuvent les voir arriver de loin, ce qui crée une tension constante.
Ce type de level design nous a également permis de faire apparaître les ennemis à distance, forçant ainsi le joueur à se déplacer et à rester actif. Le feeling du dash et des attaques au corps à corps s’en est trouvé considérablement renforcé, rendant les affrontements plus dynamiques et satisfaisants.

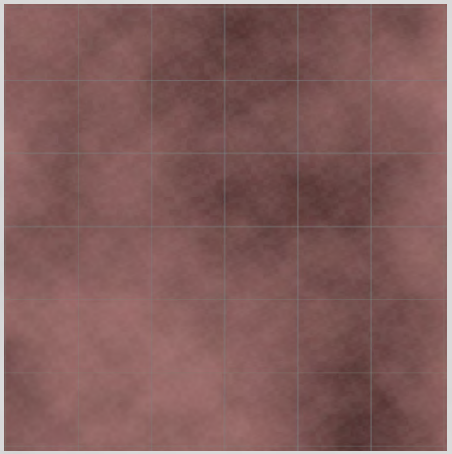


Le LD plat

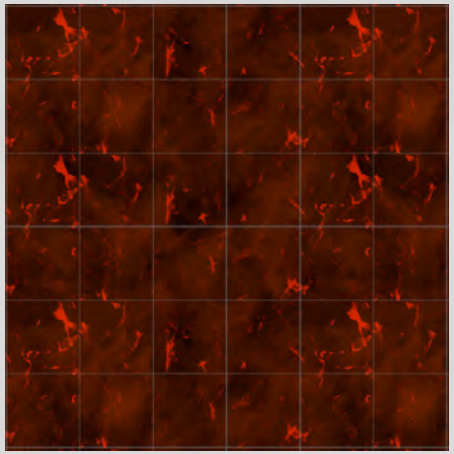
Cette approche plus épurée nous a permis, en termes de production, de nous concentrer sur les retours visuels et sonores (feedbacks) ainsi que sur la finalisation de notre direction artistique, jusqu'à obtenir le rendu souhaité.

L'un des inconvénients majeurs du level design plat était la répétition visible de la texture du sol, d'autant plus marquée compte tenu des capacités du personnage à prendre de la hauteur, offrant ainsi une vue plongeante sur l'environnement. Pour résoudre ce problème, nous avons dû créer une texture de sol optimisée, intégrant un système de LOD (Level of Detail) et un randomiseur sur la sur-texture de craquelures (cracks). Cela permettait de casser la répétition visuelle tout en préservant les performances.
Ici, on peut observer le LOD (Level of Detail) en action. Il a pour effet de faire progressivement disparaître les détails du sol lorsque le joueur prend de la hauteur. Cela permet, d'une part, de réduire la visibilité de la répétition des textures, et d'autre part, de rendre la montée en altitude plus lisible et agréable visuellement. En effet, un excès de détails vus de très haut peut créer un effet de «bouillie de pixels», rendant la lecture de l'espace confuse.

Cependant, le LOD ne fait pas totalement disparaître les textures : celles-ci restent légèrement visibles même à distance. Ce choix permet de maintenir une cohérence visuelle et d'accentuer la sensation de profondeur, tout en préservant la lisibilité de l'environnement. Par ailleurs, le LOD n'affecte pas les éléments de gameplay importants comme les ennemis ou les geysers, qui restent toujours visibles. Ainsi, même à haute altitude, le joueur peut repérer clairement les points d'intérêt et planifier ses déplacements.



Texture Sol V1

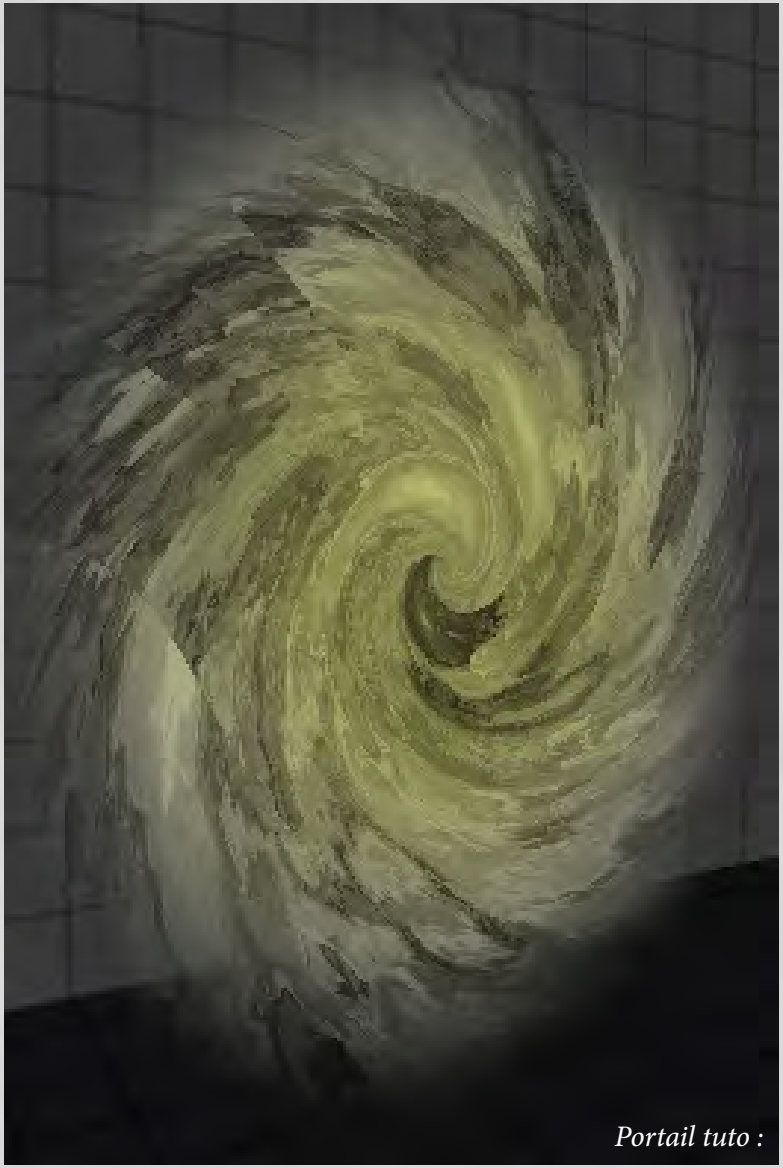


Texture Sol V2

Tutoriel :

Intention :
Le level design du tutoriel a une vocation principalement explicative plutôt qu'esthétique. Son objectif était d'enseigner au joueur les bases du gameplay de notre jeu. Pour cela, le joueur progresse à travers trois salles distinctes, dans lesquelles il doit accomplir certaines actions spécifiques afin de faire apparaître un portail. Ce portail lui permet d'accéder à la salle suivante, jusqu'à atteindre, en fin de parcours, la scène principale.

Portail :
Les portails du tutoriel se distinguent de ceux de la scène principale par leur couleur. En effet, dans le tutoriel, les portails servent à transporter le joueur vers la salle suivante, tandis que dans la scène principale, les portails sont utilisés pour invoquer des ennemis.
La première salle du tutoriel a pour objectif d'enseigner au joueur les bases du déplacement et du saut. Elle propose un environnement simple et progressif, conçu pour que le joueur se familiarise avec les contrôles de base dans un espace sans danger.

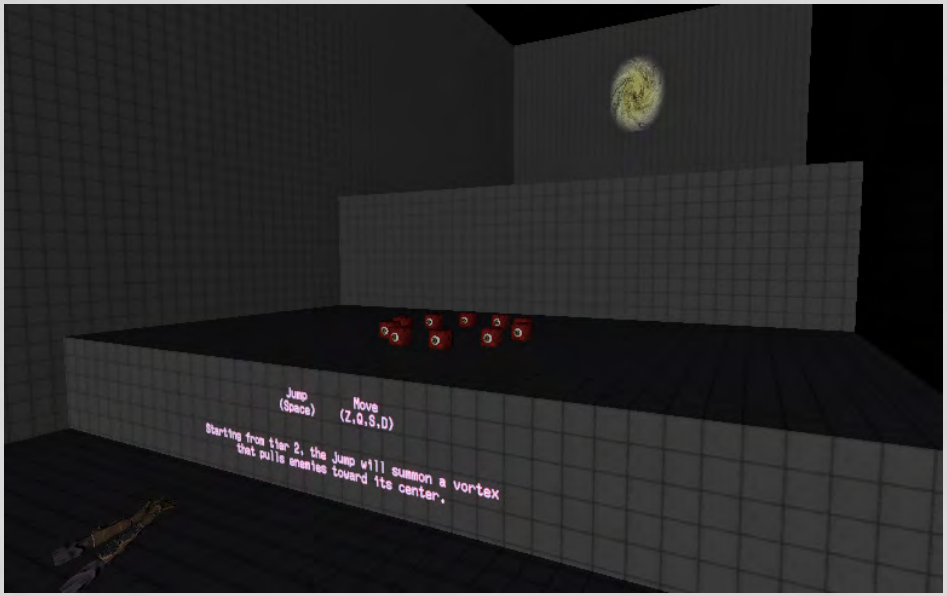


Portail tuto :

Salle apprentissage des mouvements :

Dans la première salle, deux panneaux indicateurs accompagnent une mise en situation pour guider le joueur. Cette section introduit le déplacement et le saut, tout en présentant une particularité du gameplay : le saut génère un vortex qui attire les ennemis vers son centre. Pour illustrer cela, des ennemis inoffensifs, immobiles et sans IA, sont placés dans l'environnement afin de servir d'exemple sans représenter de danger.

La deuxième salle a pour objectif d'apprendre au joueur à se défendre. Elle introduit les mécaniques de tir à distance ainsi que l'attaque au corps à corps. Là Wencore, le joueur est placé dans un environnement contrôlé lui permettant d'expérimenter ces actions en toute sécurité, avant d'être confronté à de véritables menaces dans la scène principale.

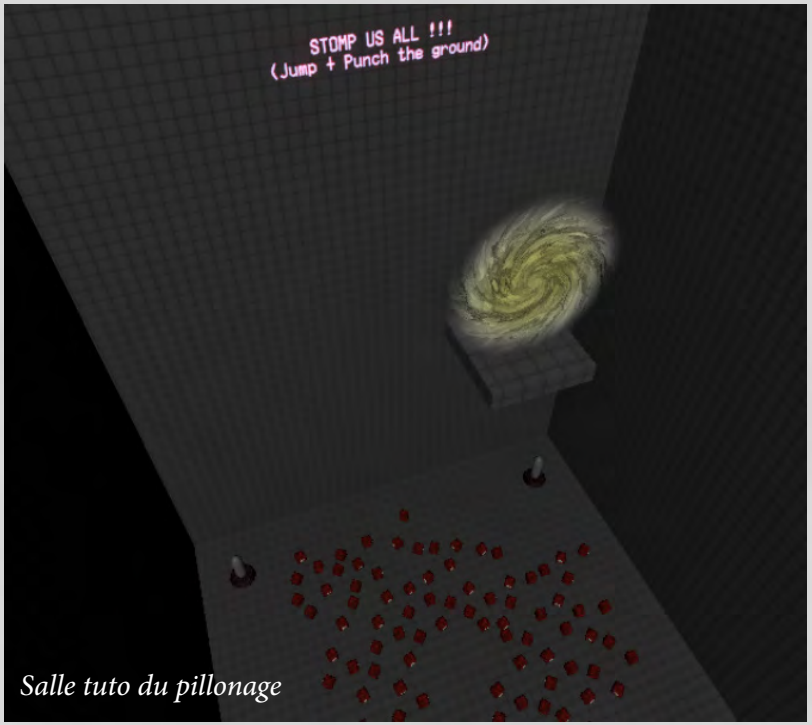


Salle tuto des mouvements

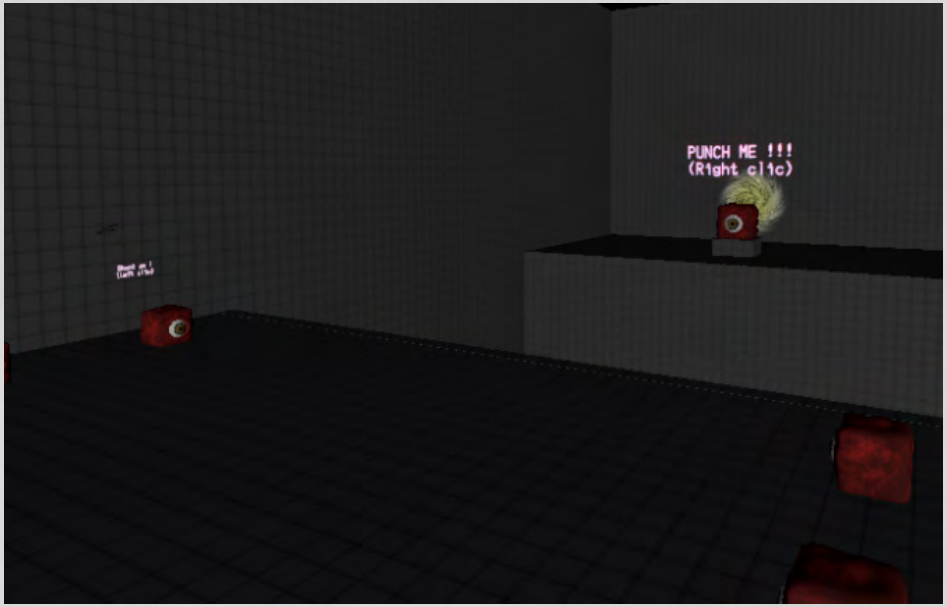
Salle apprentissage du tir et du punch :

Dans la salle 2, le saut est volontairement désactivé afin de forcer le joueur à utiliser l'attaque au corps à corps (le punch) pour progresser. Cette contrainte encourage l'expérimentation de cette mécanique dans un contexte sécurisé. En cas de chute, le joueur est simplement replacé au début de la salle sans perdre sa progression, ce qui permet d'apprendre par l'erreur sans générer de frustration.

La salle 3 a pour objectif d'enseigner deux éléments clés : le fonctionnement des geysers et l'utilisation du pilonnage, une compétence spéciale du personnage. Cette dernière salle prépare le joueur aux interactions complexes qu'il rencontrera dans la scène principale, en liant mobilité verticale et capacité offensive.



Salle tuto du pillonage



Salle tuto du tir et du punch

Salle d'apprentissage du Pillonage :

Le stomp (ou pilonnage) étant l'une des mécaniques les plus importantes du jeu, la troisième salle est conçue pour forcer le joueur à l'utiliser et à expérimenter avec. À travers des situations concrètes, le joueur comprend que les geysers lui permettent de gagner rapidement de la hauteur, ce qui est essentiel pour déclencher un stomp efficace. Cette mise en situation renforce l'association entre mobilité verticale et attaque, tout en préparant le joueur aux défis de la scène principale.

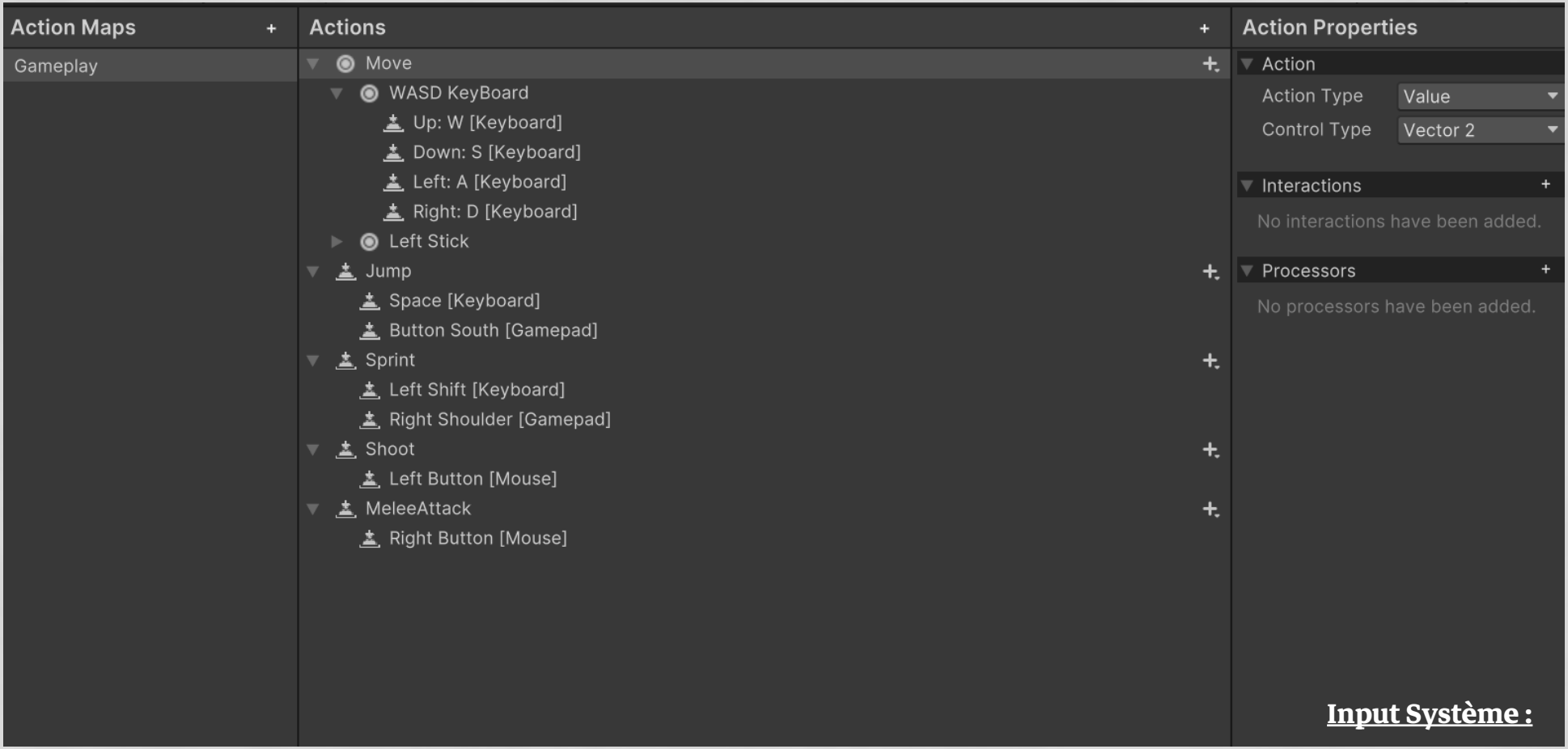
Une fois cette salle complétée, le portail final transporte le joueur vers la scène principale.

Programmation

Player Controller Systeme :

Le système d'entrée du joueur utilise le nouveau Input Manager officiel d'Unity. En créant et en configurant à l'avance une action map, les touches sont ensuite assignées de manière fixe. Ici, les touches sont basées sur les touches physiques, ce qui signifie que, quel que soit le type de clavier ou la langue, la disposition reste la même.

Un script spécifique nommé S_InputManager est chargé de gérer les entrées afin que les autres scripts puissent les utiliser facilement. Cela permet également de faciliter la création future d'une fonctionnalité de rebind des touches personnalisées, offrant aux joueurs la possibilité de configurer les touches selon leurs préférences.

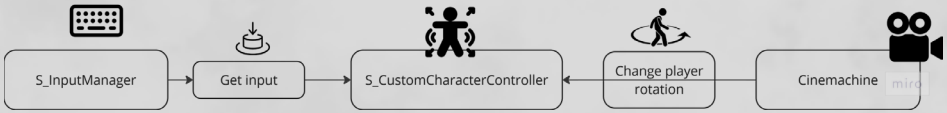


Input Système :

Player (Personnage Ultime) :

Le système de contrôle du joueur utilise le Character Controller officiel d'Unity pour exécuter des fonctions telles que le déplacement et la détection des pentes. Ensuite, en combinant cela avec des scripts personnalisés, nous avons obtenu un contrôleur de personnage parfaitement adapté aux besoins de notre jeu.

Ce contrôleur est lié à un autre script, S_InputManager, pour recevoir les commandes d'entrée du joueur. Il est également associé à la caméra afin d'appliquer divers retours visuels. L'orientation du joueur est directement gérée par la rotation de la caméra, ce qui garantit des changements de direction rapides en vue à la première personne. Le déplacement, quant à lui, s'adapte simplement à la direction actuelle.



Voici le diagramme de flux représentant la logique du déplacement du personnage :

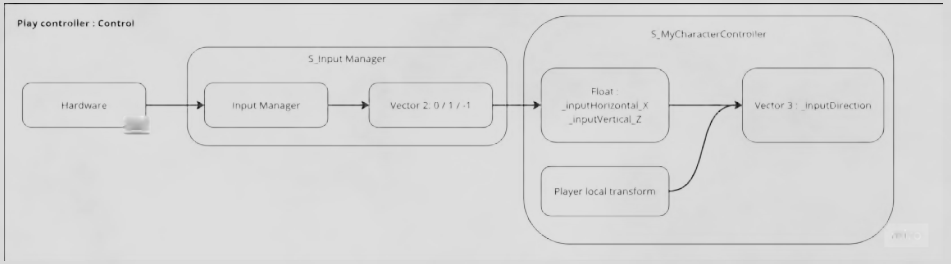


Schéma de la vitesse du Controller

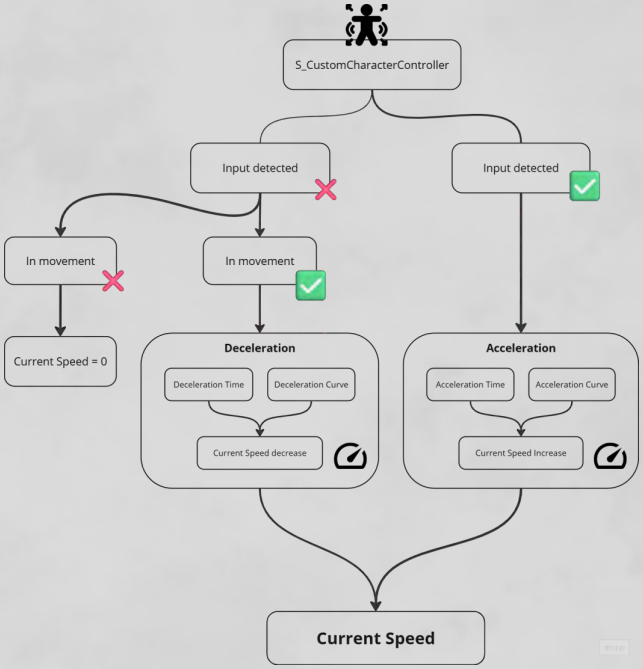
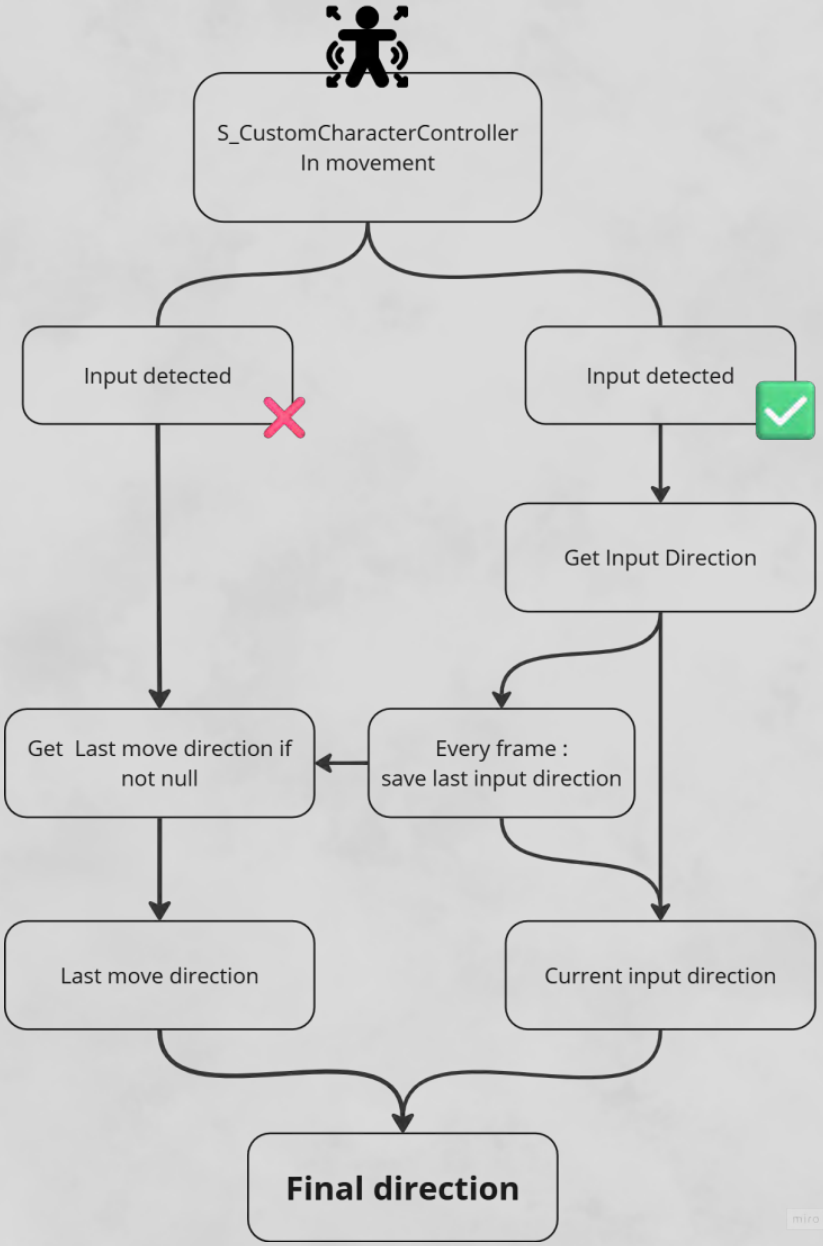
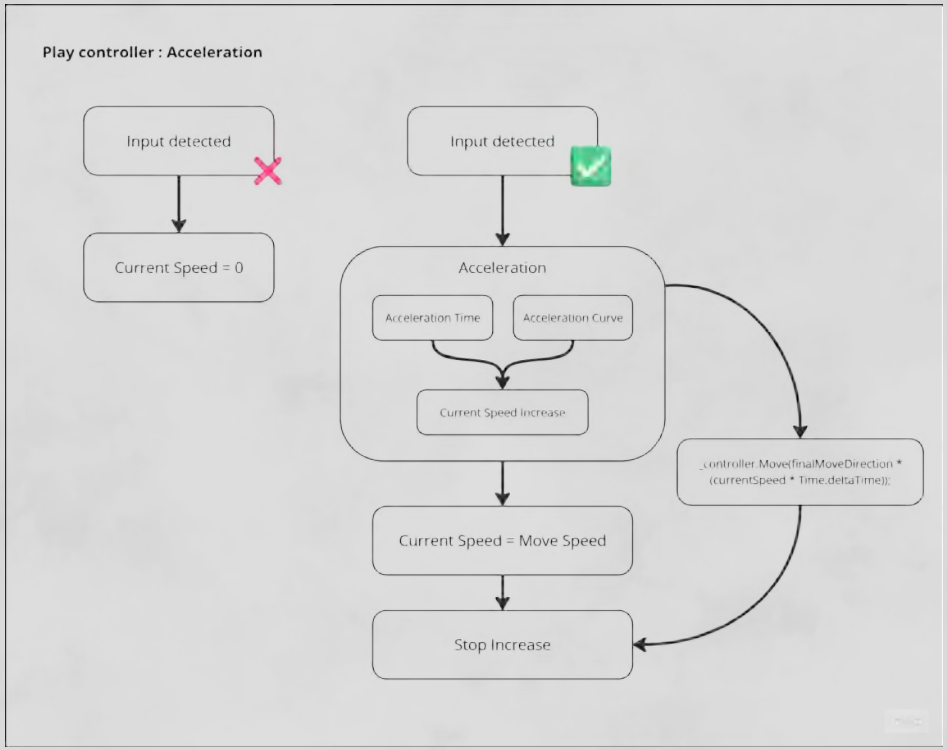
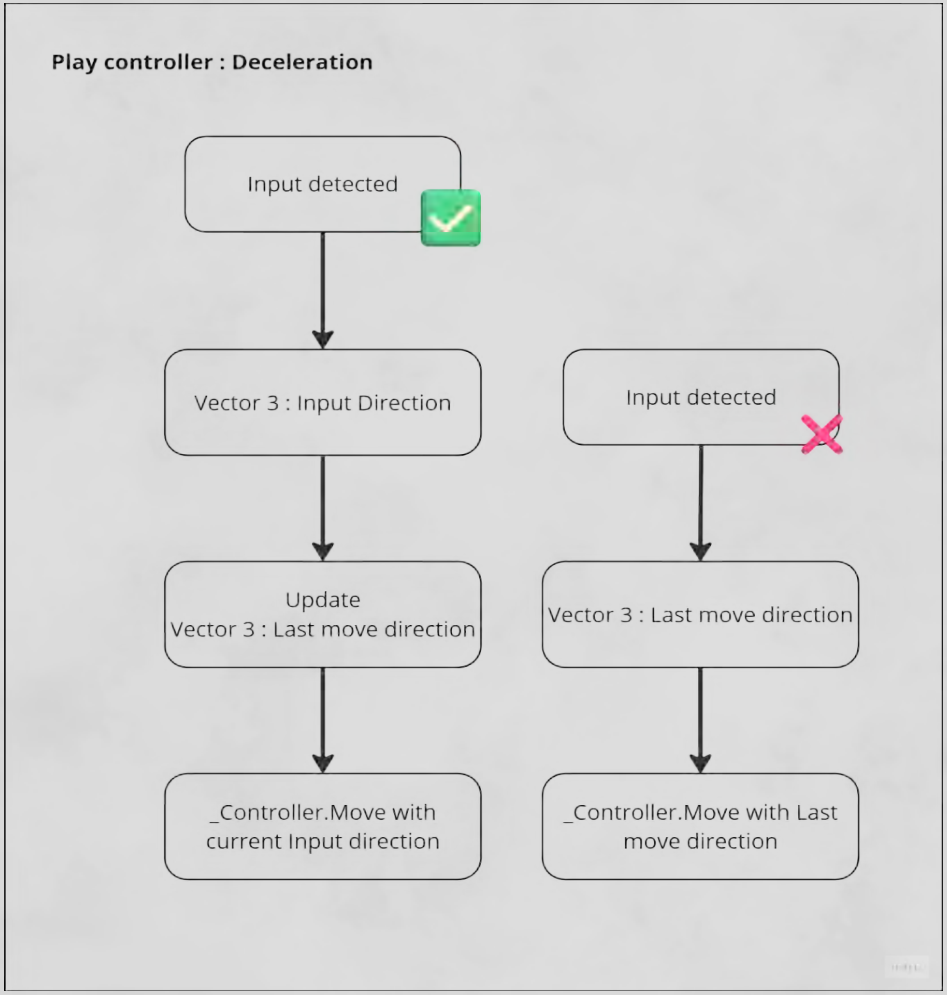


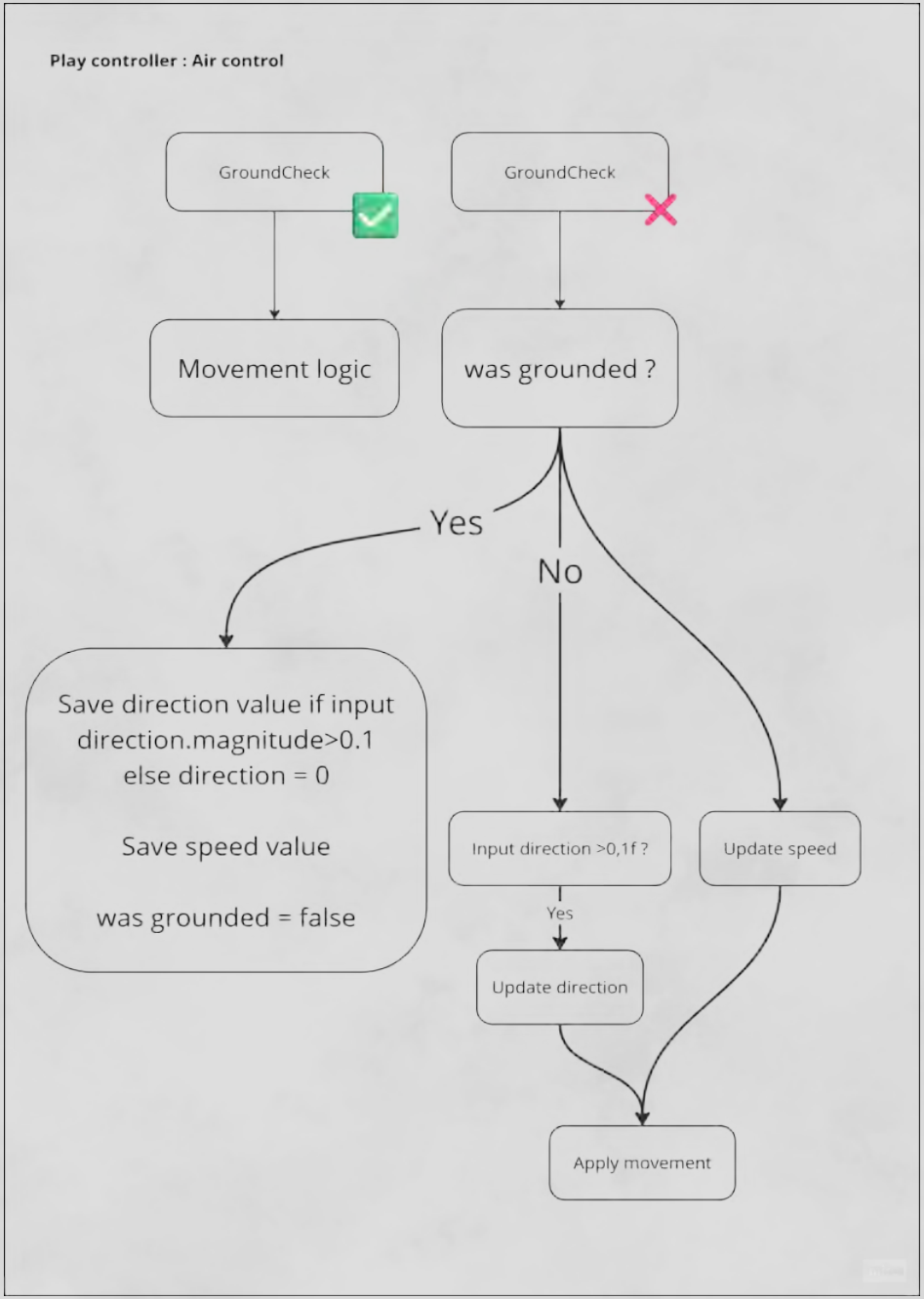
Schéma de la direction du Controller

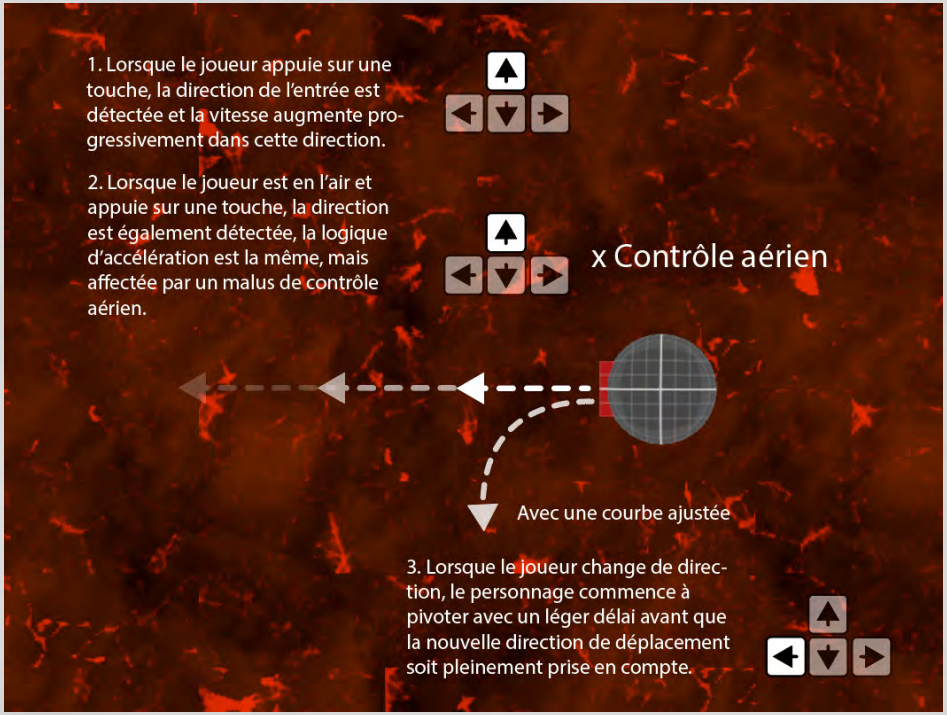


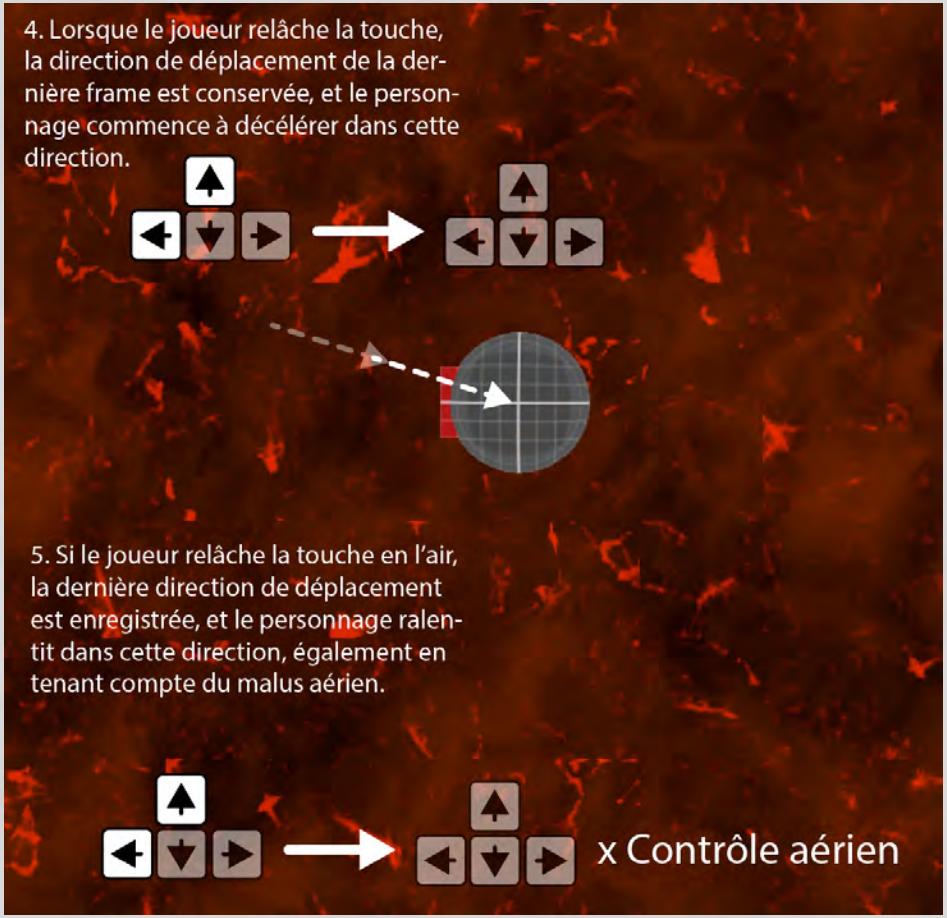
Cette section détaille plus précisément les mécanismes d'accélération et de ralentissement liés au mouvement du personnage.




Play controller : Deceleration
Input detected
Vector 3 : Input Direction
Update
Vector 3 : Last move direction
_Controller.Move with
current Input direction
Input detected
Vector 3 : Last move direction
_Controller.Move with Last
move direction


Play controller : Air control
GroundCheck
GroundCheck
Movement logic
was grounded ?
Yes
No
Save direction value if input
direction.magnitude>0.1
else direction = 0
Save speed value
was grounded = false
Input direction >0,1f ?
Update speed
Yes
Update direction
Apply movement


1. Lorsque le joueur appuie sur une touche, la direction de l'entrée est détectée et la vitesse augmente progressivement dans cette direction.
2. Lorsque le joueur est en l'air et appuie sur une touche, la direction est également détectée, la logique d'accélération est la même, mais affectée par un malus de contrôle aérien.
x Contrôle aérien
Avec une courbe ajustée
3. Lorsque le joueur change de direction, le personnage commence à pivoter avec un léger délai avant que la nouvelle direction de déplacement soit pleinement prise en compte.


4. Lorsque le joueur relâche la touche, la direction de déplacement de la dernière frame est conservée, et le personnage commence à décélérer dans cette direction.
5. Si le joueur relâche la touche en l'air, la dernière direction de déplacement est enregistrée, et le personnage ralentit dans cette direction, également en tenant compte du malus aérien.
x Contrôle aérien

Systeme d'énergie :

1. Vue d’ensemble
Le système énergétique repose sur deux scripts : S_EnergyStorage, qui enregistre la quantité d’énergie, gère l’augmentation ou la diminution et expose des méthodes publiques pour les autres compétences ; et S_EnergyAbsorption_Module, qui détecte les orbes d’énergie dans la scène, les attire vers le joueur puis dépose leur valeur dans le stockage. Ensemble, ils assurent tout le cycle « collecter -> stocker -> consommer ».

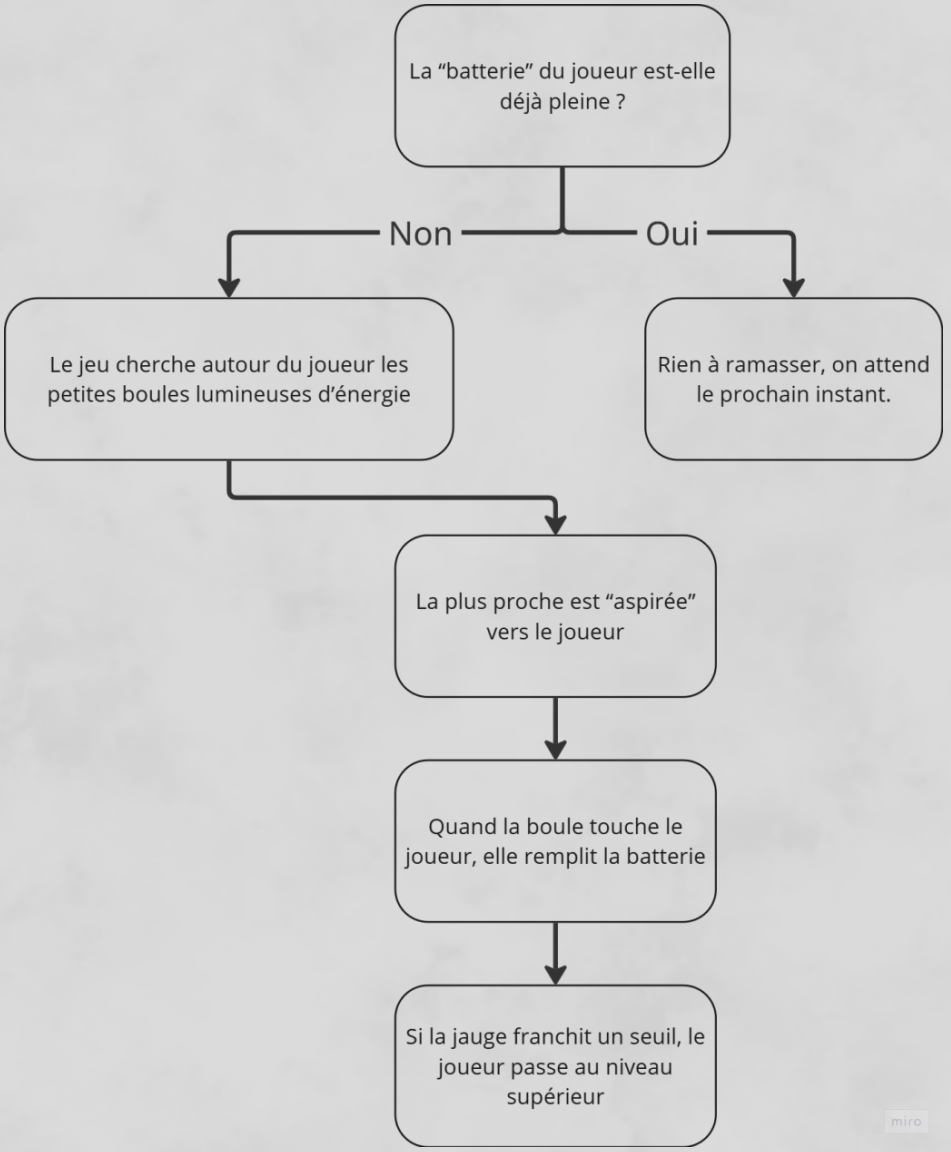
2. Rôle précis de chaque script
Le script S_EnergyStorage maintient les variables maxEnergy et currentEnergy, déclenche les montées ou descentes de niveau d’après un tableau d’objets EnergyLevel et émet l’évènement OnLevelChange chaque fois que le niveau change. Il applique aussi une « période de grâce » pendant laquelle l’énergie peut redescendre sans faire immédiatement régresser le niveau.

Le module S_EnergyAbsorption_Module balaie, à chaque image, un rayon défini autour du joueur grâce à Physics.OverlapSphereNonAlloc. Les orbes valides sont triés par distance ; chacun est alors rapproché par une coroutine : dès qu’il se trouve à moins de 4,5 m, on calcule l’énergie absorbée (givenPoint × currentComboMultiplier), puis on appelle AddEnergy du stockage. L’orbe est enfin renvoyé dans un pool ou détruit.

3. Logique d’exécution,
À chaque frame, le module d’absorption saute l’étape de détection si l’énergie est déjà pleine ; sinon, il détecte les orbes, filtre ceux qui ne portent pas de composant EnergyType, puis les classe par proximité. Une coroutine par orbe l’anime jusqu’au joueur, applique la quantité d’énergie et remet l’objet dans son pool. Parallèlement, S_EnergyStorage.Update vérifie si la réserve actuelle atteint les pré-requis d’un niveau supérieur ; si oui, il monte de niveau et redistribue, selon percentageGiveaway, un bonus d’énergie. Dans le cas contraire, si l’énergie descend sous le seuil du niveau courant, une minuterie de grâce démarre ; lorsque celle-ci s’achève, un niveau peut être perdu. Les autres scripts consomment simplement l’énergie par RemoveEnergy, tandis que l’évènement OnLevelChange alimente l’interface, les effets visuels ou sonores.

4. Interface publique exposée aux autres systèmes
AddEnergy(float amount) ajoute une quantité positive, par exemple lorsqu'un objet est ramassé ou qu'une régénération passive s'active. RemoveEnergy(float amount) débite la réserve, typiquement au lancement d'une compétence. L'évènement On-LevelChange(EnergyLevelState state, int levelIndex) signale la montée, la descente, le début ou la fin d'une période de grâce ; l'UI peut ainsi adapter la couleur des jauges, déclencher un son, etc.

5. Pistes d'extension,
Le système peut être généralisé à plusieurs types d'énergie (feu, glace, foudre) en stockant un dictionnaire <Type, float>. Côté interface, on peut lier chaque niveau à une variation de couleur, d'écran shake ou de musique de fond. Dans un contexte multi-joueur, currentEnergy deviendrait une NetworkVariable<float> afin de synchroniser les valeurs et les évènements de niveau sur tous les clients.



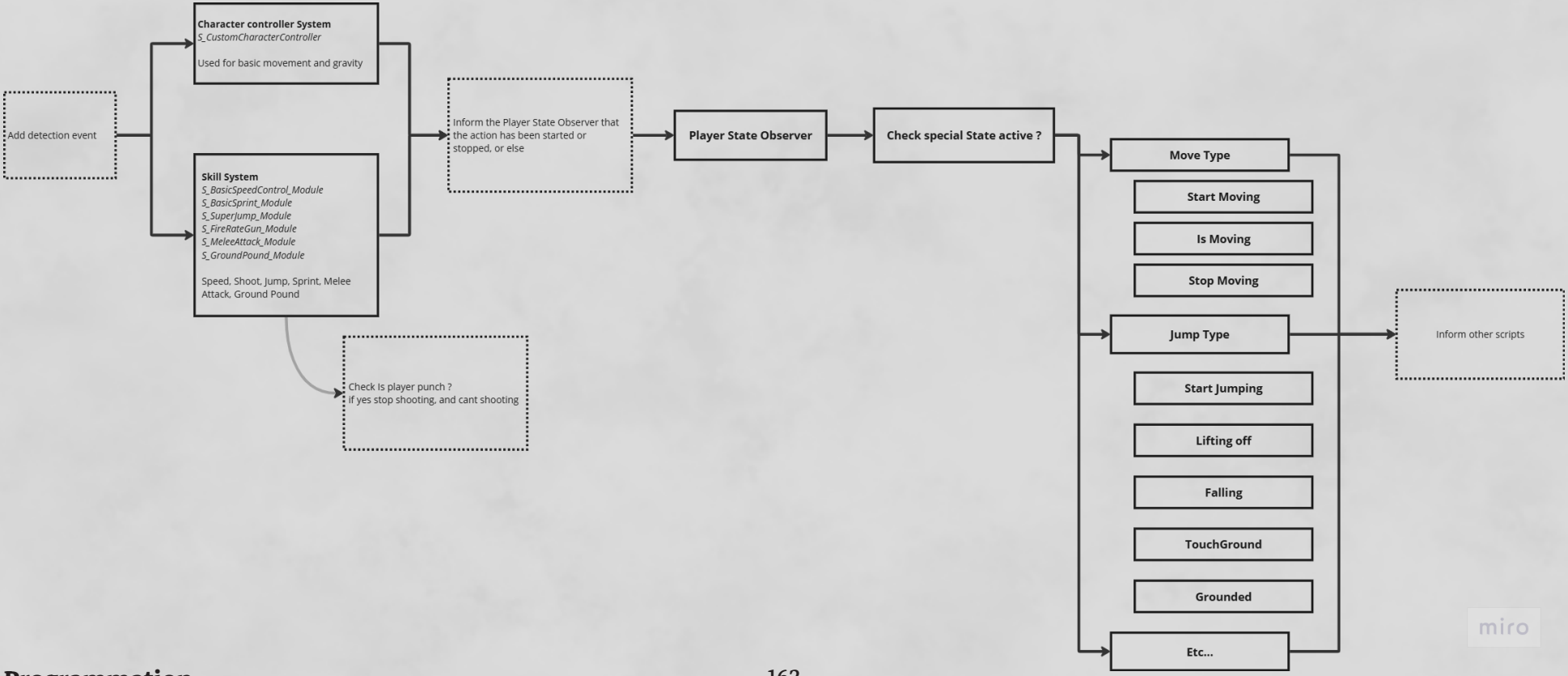
Systeme Compétence :

Introduction

Le système de compétences repose sur plusieurs modules indépendants, chacun attaché à une capacité du joueur. Ces modules interagissent avec deux éléments centraux :

le stockage d'énergie (classe S_EnergyStorage) pour consommer ou vérifier la ressource nécessaire à l'exécution,

l'observateur d'état (S_PlayerStateObserver) pour transmettre des événements aux systèmes extérieurs (interface, audio, effets).
Ce document décrit pour chaque module le fonctionnement interne, l'enchaînement logique des appels, et ses interactions avec ces deux systèmes.



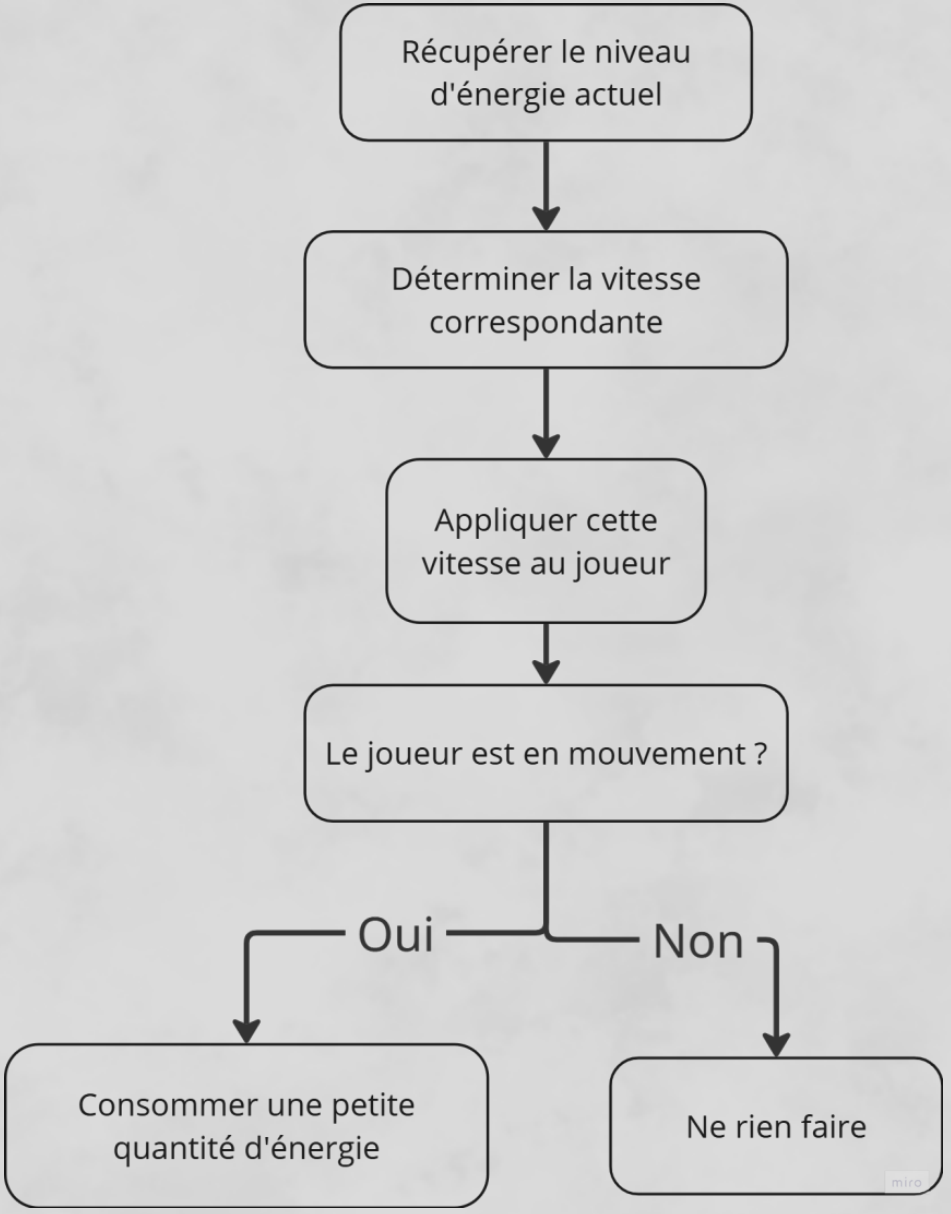
S_BasicSpeedControl_Module

Ce module est actif uniquement lorsque le joueur ne sprint pas. À chaque Update(), il récupère le niveau d'énergie actuel via le stockage, sélectionne une vitesse de déplacement depuis un tableau associé, et applique cette vitesse à la structure de mouvement. Si le joueur est en déplacement, une consommation passive d'énergie est déclenchée en continu.

Il ne communique pas directement avec l'observateur, mais dépend du niveau défini par S_EnergyStorage.

-> Interaction : energyStorage.GetCurrentLevel() + energyStorage.RemoveEnergy()

Schéma simplifiée:



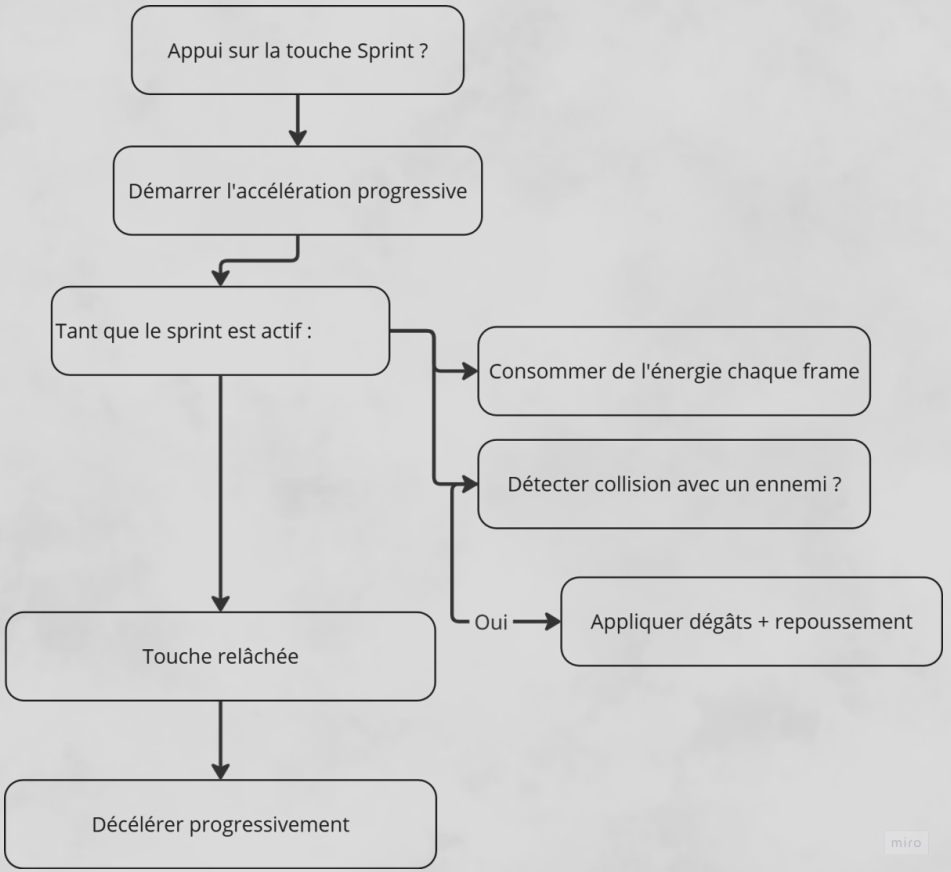
S_BasicSprint_Module

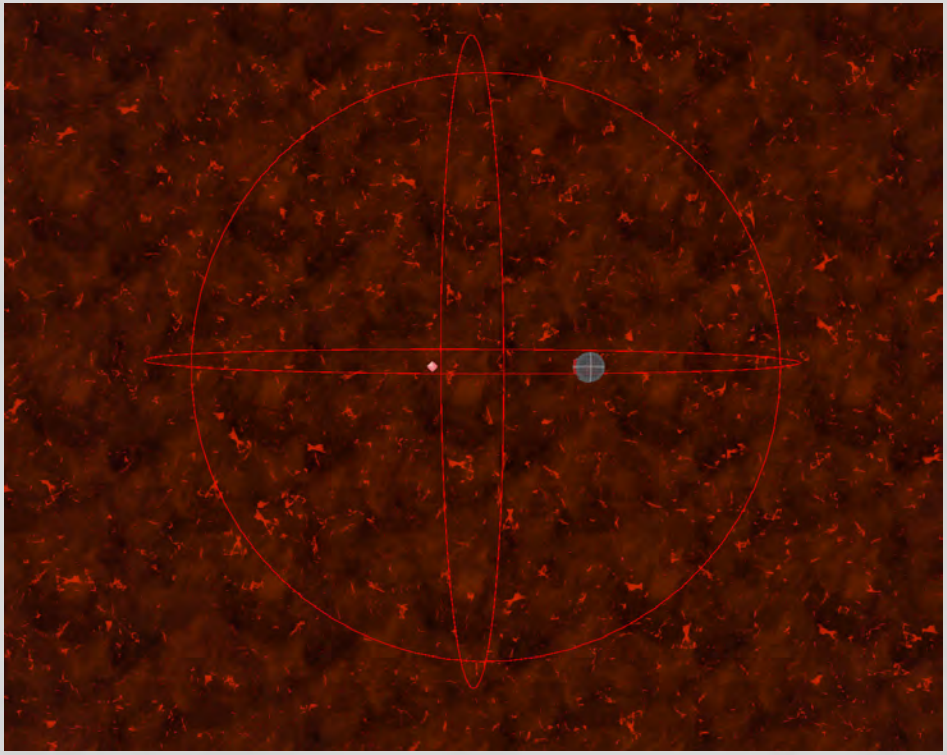
Ce module vérifie si le joueur appuie sur la touche de sprint et si l'énergie est suffisante. Lors de l'activation, il appelle energyStorage.RemoveEnergy() à chaque frame pour maintenir le sprint. Un système d'accélération progressive modifie la vitesse cible du joueur. Simultanément, une détection de collision frontale est effectuée (via Physics.SphereCast), et si un ennemi est touché, le module applique des dégâts et force, puis déclenche un événement OnSprintHit.

Lorsque le sprint s'interrompt (manque d'énergie ou relâchement de la touche), la vitesse revient progressivement à la valeur standard, et un événement OnStopSprinting est envoyé à l'observateur.

-> Interaction : consommation d'énergie continue + envoi de messages à S_PlayerStateObserver.

Schéma simplifiée:





Sprint Levels
Sprint Levels 3
Element 0
Level 2
Sprint Speed 45
Energy Consumption Rate 250
Sprint Damage 65
Drop Bonus 2
Element 1
Level 3
Sprint Speed 60
Energy Consumption Rate 800
Sprint Damage 300
Drop Bonus 0
Element 2
Level 4
Sprint Speed 80
Energy Consumption Rate 4500
Sprint Damage 2000
Drop Bonus -10

Sprint Settings
Enemy Layer Enemy
Range Distance 3.5
Range Offset X 0 Y 0 Z 0
Push Force 20
Sprint Range 10
Sprint Damage Cooldown 0.5

Acceleration/Deceleration Settings
Acceleration Time 0.5
Acceleration Curve
Deceleration Time 0.5
Deceleration Curve

S_FireRateGun_Module

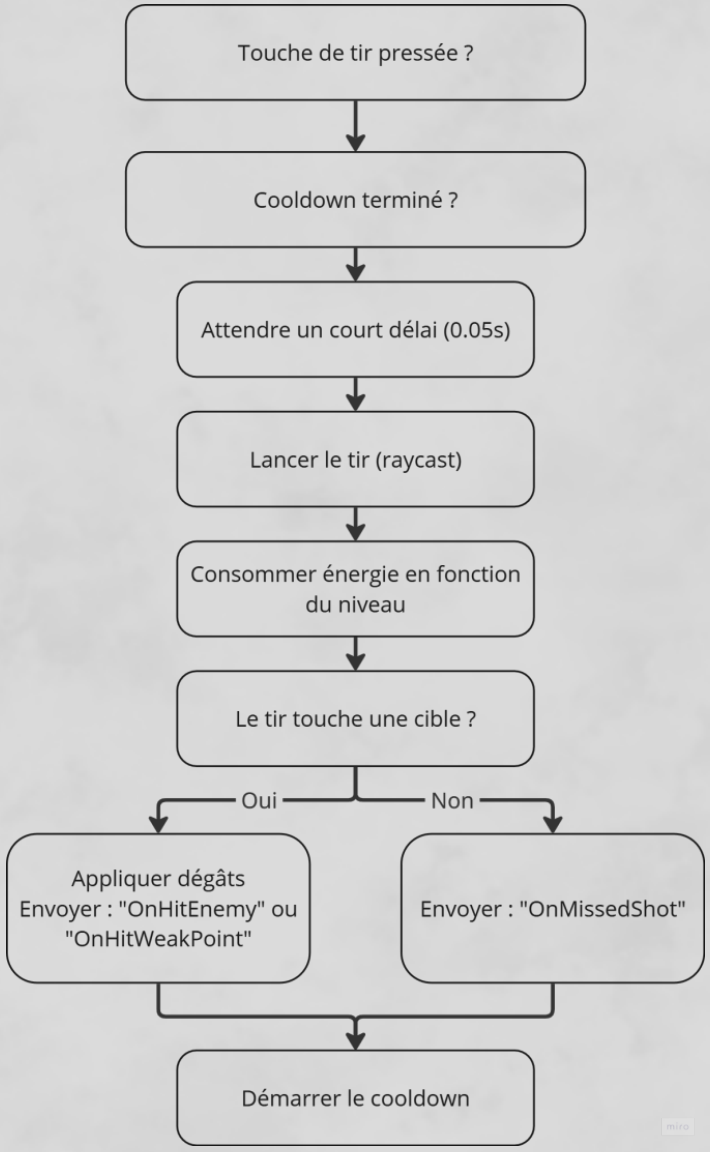
Ce module gère une logique de cadence de tir, en lien avec le niveau d'énergie. Lorsqu'un tir est lancé, le module :

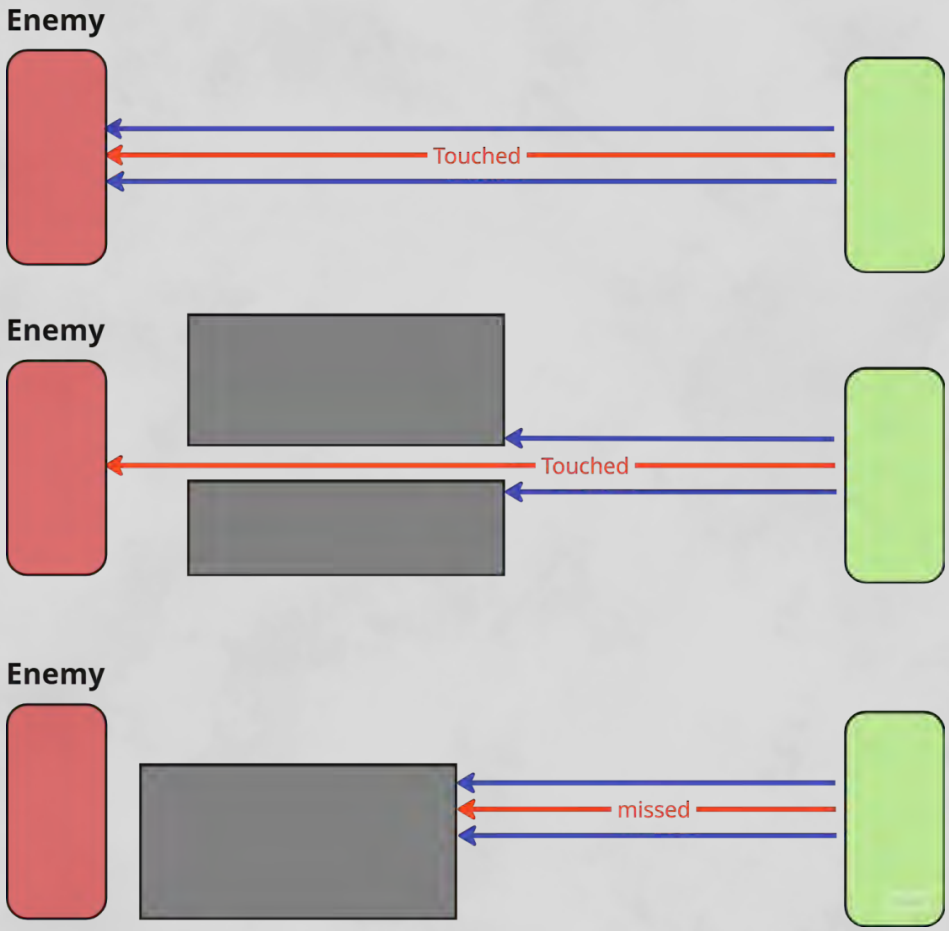
1. vérifie si le cooldown est terminé
2. consomme l'énergie nécessaire (energyStorage.RemoveEnergy())
3. déclenche un court délai d'animation (via coroutine)
4. applique des dégâts via un système de raycast
5. démarre un cooldown spécifique au niveau d'énergie actuel

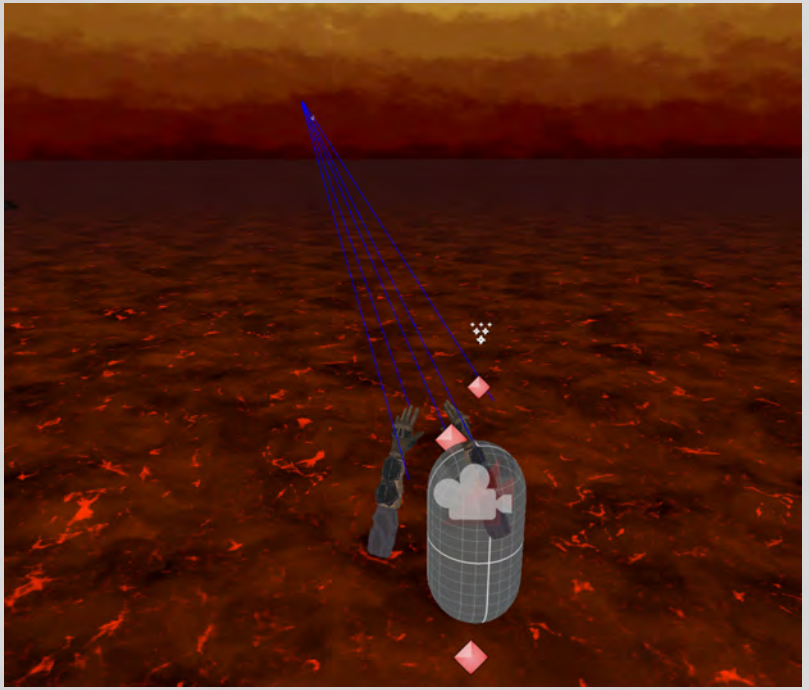
Durant chaque étape, il envoie les événements OnStartShoot, OnHitEnemy ou OnHitWeakPoint, puis OnStopShoot, pour permettre à l'observateur de relayer ces informations.

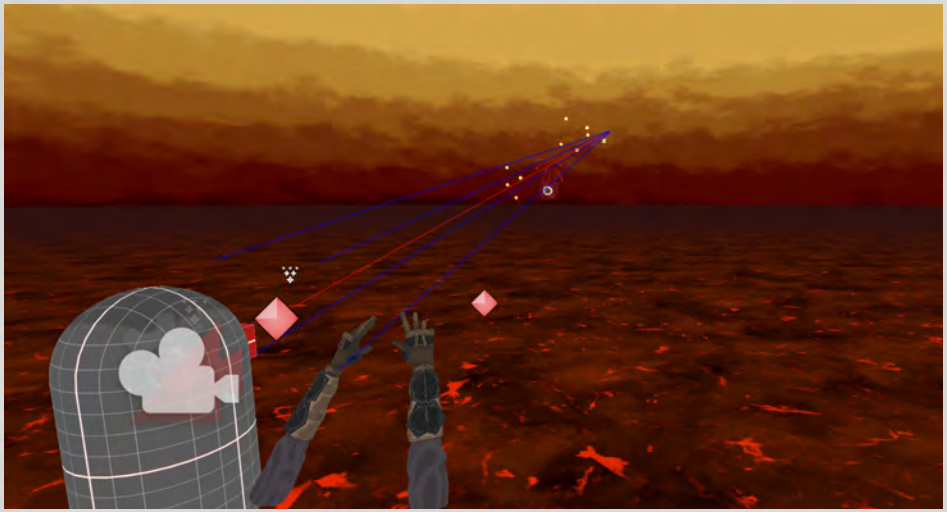
-> Interaction complète avec S_EnergyStorage pour la cadence et la consommation, et avec l'observateur pour les états de tir.

Schéma simplifiée:




Enemy
Touched
Enemy
Touched
Enemy
missed





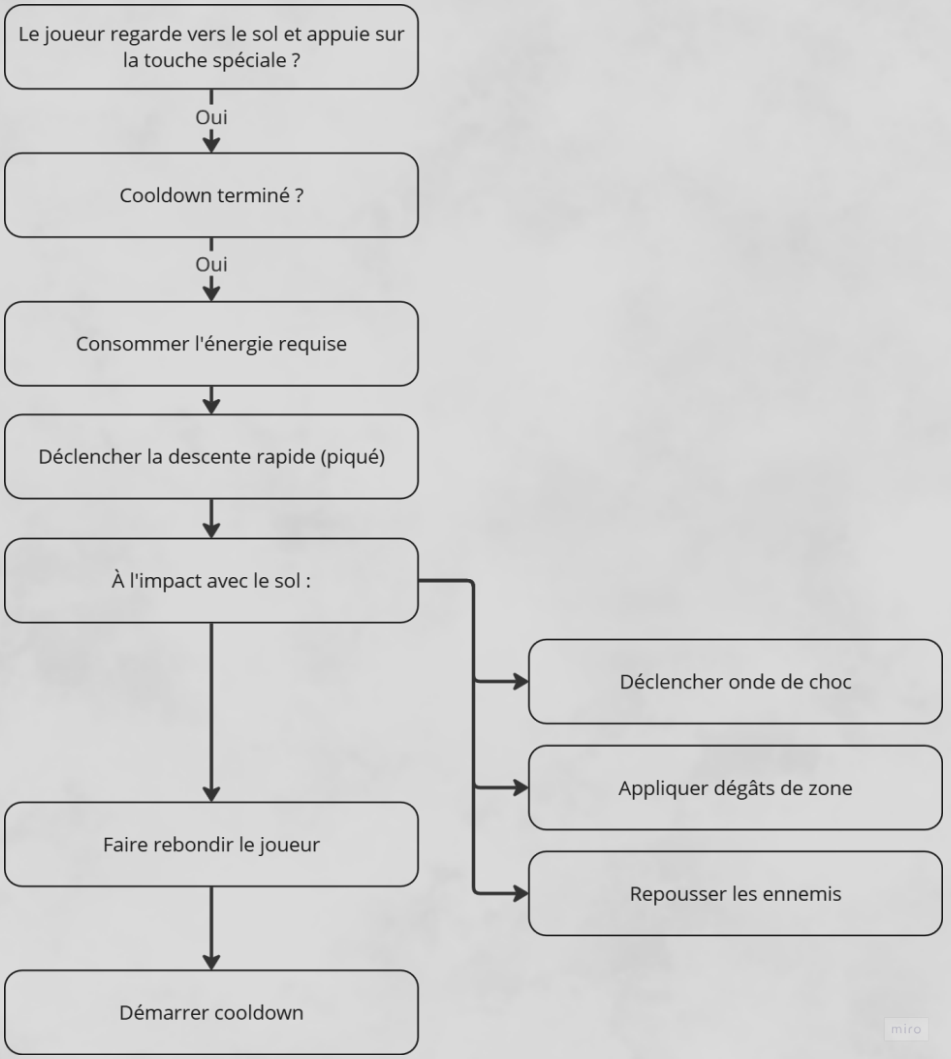
S_GroundPound_Module
Ce module déclenche une attaque verticale si le joueur est en l'air et presse une touche spécifique.
Lors de l'activation :

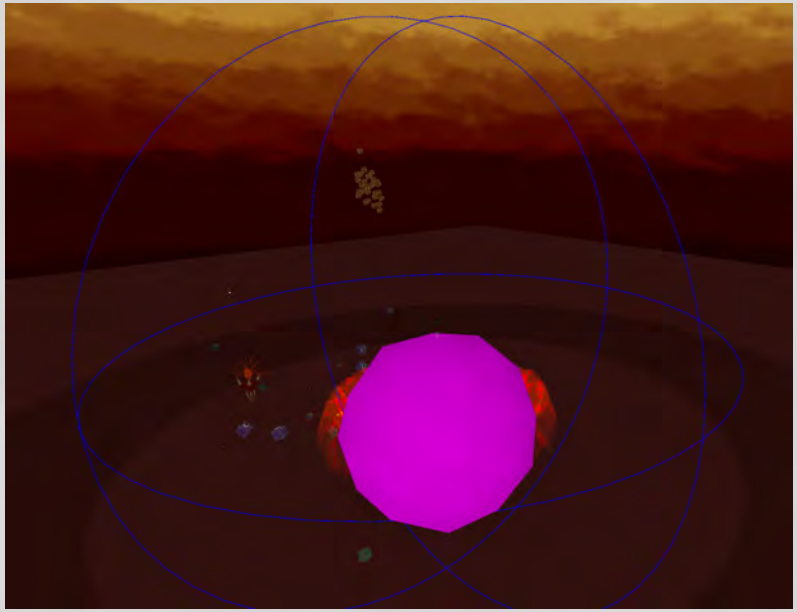
1. une quantité fixe d'énergie est retirée immédiatement
2. un mouvement de descente est appliqué (StartFallCoroutine())
3. à l'impact, un effet de zone est déclenché (DamageArea())
4. un rebond est appliqué au joueur

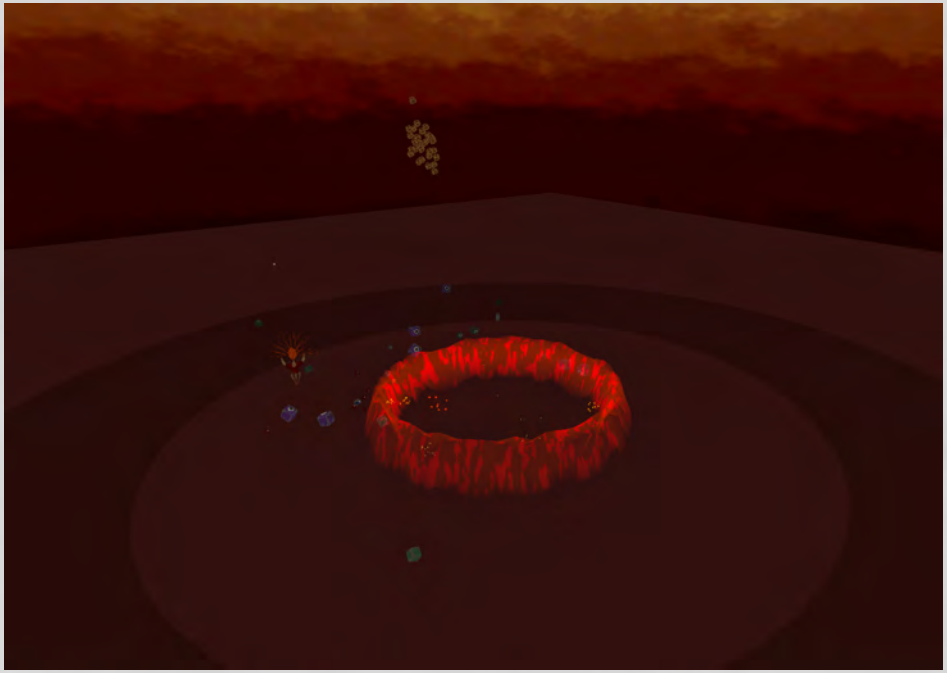
Un cooldown empêche toute nouvelle activation pendant une période définie.

-> Interaction : consommation d'énergie immédiate, effet physique local, pas de message observateur explicite.

Schéma simplifiée:





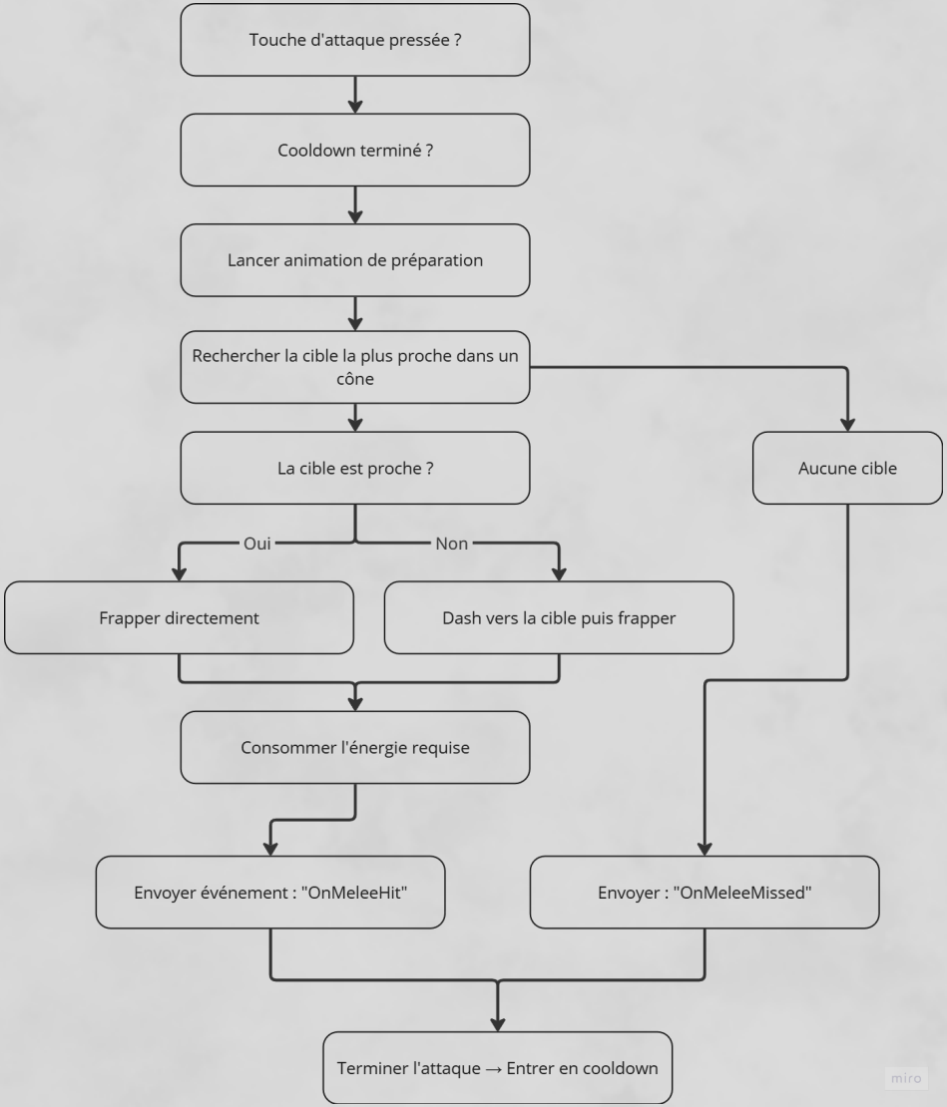


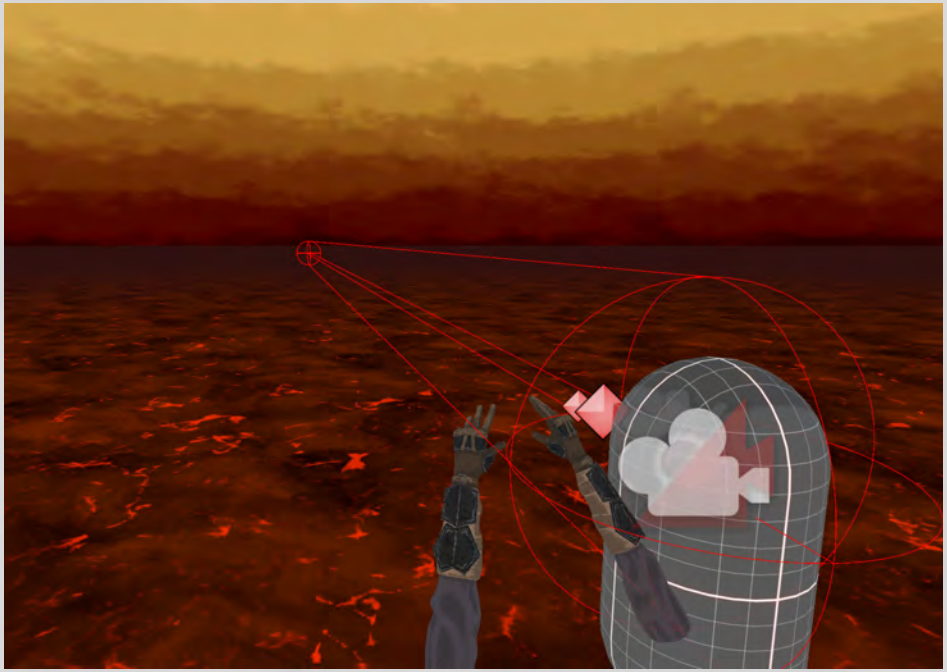
S_MeleeAttack_Module
Au moment de l'entrée de commande, le module exécute une coroutine :

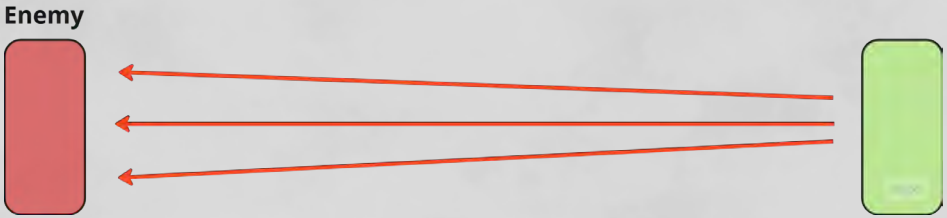
1. lance une animation de préparation
2. recherche la cible la plus proche dans un cône (fonction FindTarget())
3. si la cible est éloignée, le joueur dash automatiquement
4. une fois en portée, l'attaque est déclenchée, des dégâts sont appliqués
5. les événements OnStartMelee, OnMeleeHit, ou OnMeleeMissed sont émis

L'énergie est consommée juste avant le dash ou l'impact, selon les cas.

-> Interaction : logique complète avec stockage + émission d'événements pour l'observateur.


Touche d'attaque pressée ?
Cooldown terminé ?
Lancer animation de préparation
Rechercher la cible la plus proche dans un cône
La cible est proche ?
Aucune cible
Oui
Non
Frapper directement
Dash vers la cible puis frapper
Consommer l'énergie requise
Envoyer événement : "OnMeleeHit"
Envoyer : "OnMeleeMissed"
Terminer l'attaque → Entrer en cooldown
miro



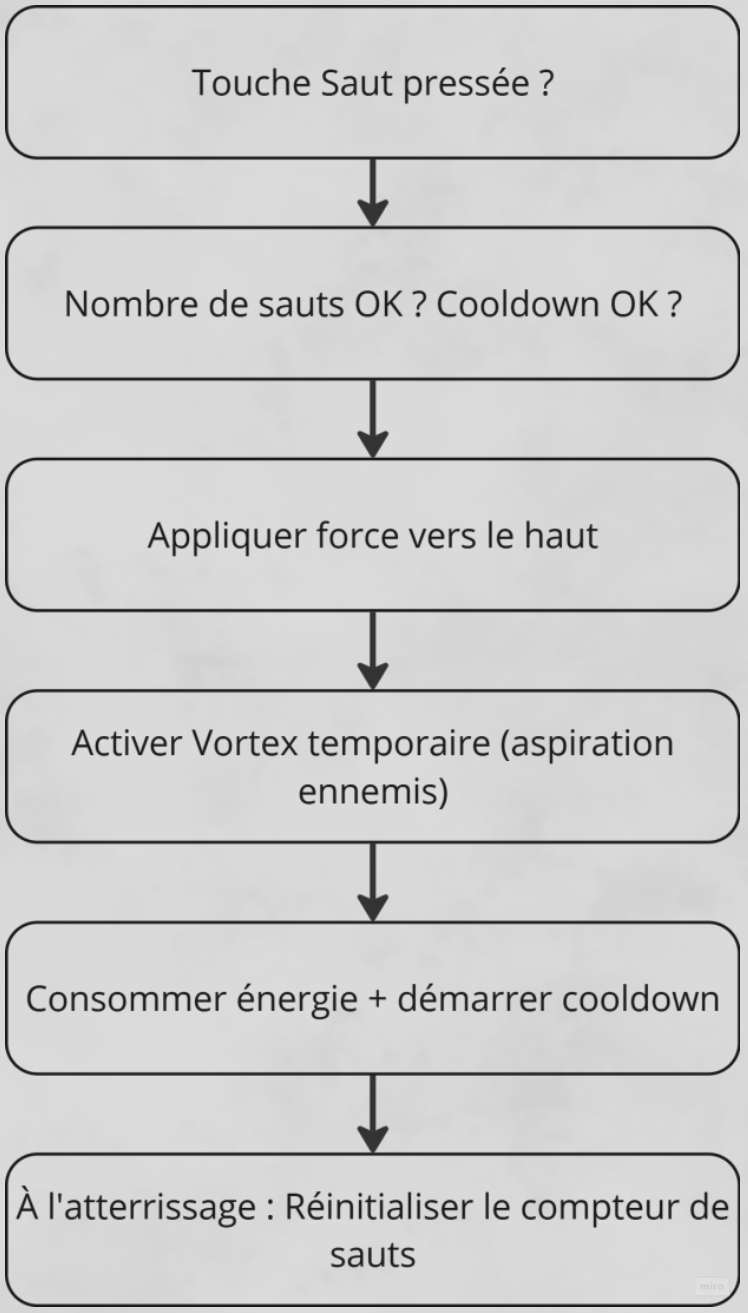

Enemy

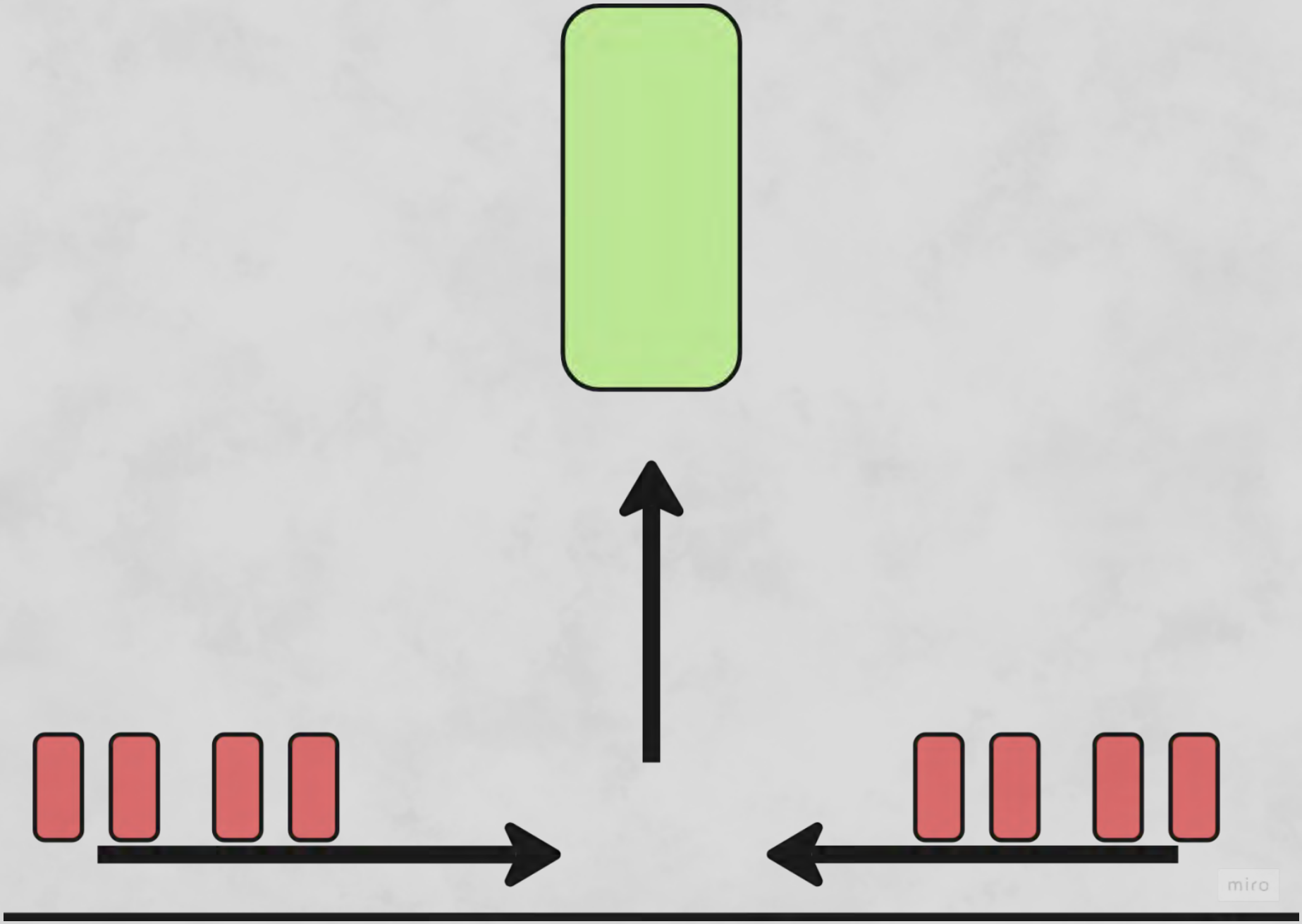
S_SuperJump_Module
Ce module permet au joueur d'effectuer un ou plusieurs sauts supplémentaires. Il garde en mémoire le nombre de sauts effectués, comparé à une limite dépendant du niveau d'énergie.
Lors de l'activation :

1. l'énergie est consommée (RemoveEnergy())
2. une force verticale est appliquée (_customCC.velocity.y = Mathf.Sqrt(currentLevel.jumpHeight * -2f * _customCC.gravity);)
3. un vortex temporaire est activé pour attirer les ennemis proches
4. un cooldown est démarré
5. un compteur de sauts est incrémenté

Le compteur est remis à zéro automatiquement lorsque le joueur touche le sol (OnLanding() appelé via un autre système).

-> Interaction : vérification du niveau et des limites, consommation d'énergie, gestion d'état local.




miro

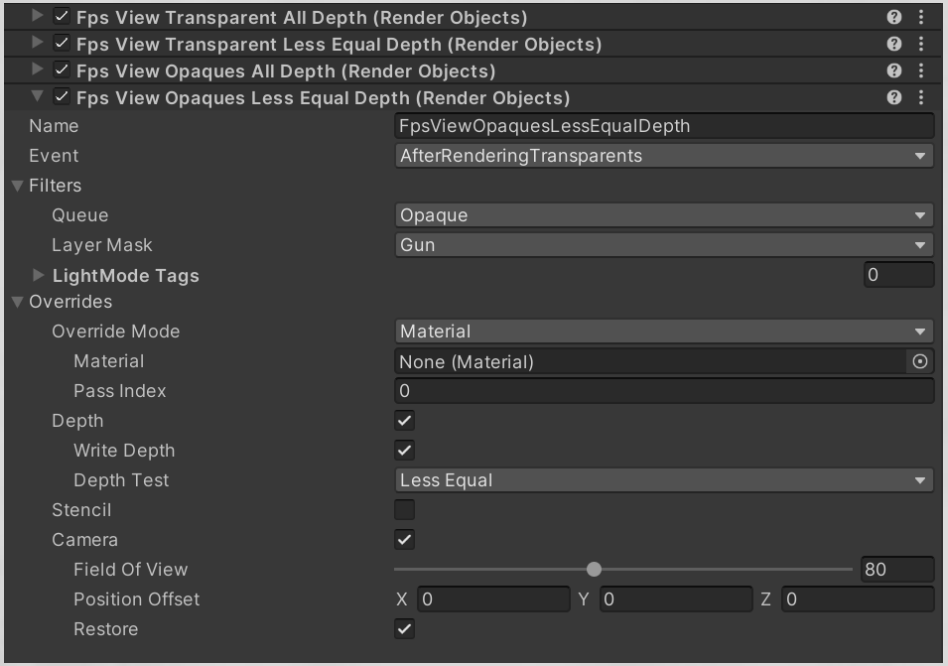
Basé sur URP Render Features et Cinemachine :

Objectif

Ce système vise à garantir un rendu cohérent des objets attachés à la vue FPS (armes, bras, viseurs, etc.), indépendamment du reste de la scène. Il permet notamment :

- d'éviter les problèmes de clipping avec les objets proches (murs, sols),
- de forcer un ordre de rendu correct pour les matériaux opaques et transparents,
- de découpler le FOV de la caméra principale et celui des objets FPS.

Le système repose sur l'Universal Render Pipeline (URP) avec une configuration personnalisée du Forward Renderer, en s'appuyant sur plusieurs Render Features de type "Render Objects". La logique de mouvement et de champ de vision est gérée via Cinemachine.



Structure des Render Features

Le système utilise quatre Render Features séparées, toutes appliquées uniquement sur le Layer Gun, qui contient les objets de la vue FPS. Chacune est responsable d'un cas précis de rendu (matériau opaque ou transparent, avec ou sans test de profondeur classique).

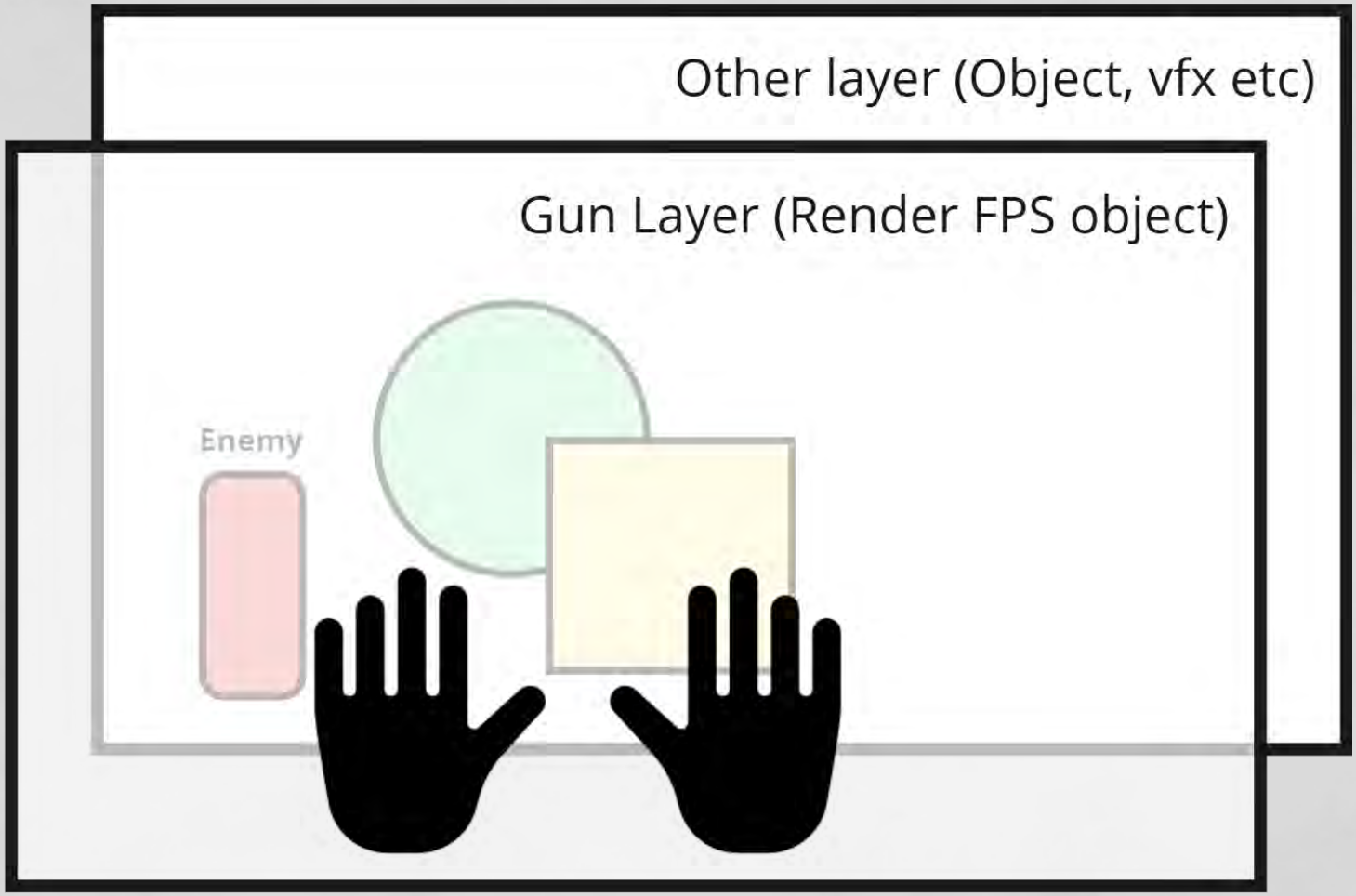
Fps View Opaques All Depth:
Cette feature force le rendu des objets opaques du layer Gun sans tenir compte de la profondeur. Elle permet aux objets FPS d'être rendus au-dessus de toute autre géométrie opaque, indépendamment de leur position réelle dans l'espace.

Fps View Opaques Less Equal Depth:
Cette version alternative permet un rendu respectant les contraintes classiques de profondeur, tout en conservant les effets de surcharge de champ de vision. Elle est utile dans les cas où le test Always provoquerait des artefacts.

Fps View Transparent All Depth
Ce pass est destiné aux objets transparents (viseurs, effets spéciaux) qui doivent être rendus systématiquement par-dessus tout le reste. Le test Always garantit leur visibilité même à travers d'autres objets.

Fps View Transparent Less Equal Depth
Ce pass gère les objets transparents tout en respectant le Z-buffer classique, permettant ainsi un rendu réaliste si nécessaire (par exemple pour éviter de rendre un effet si un mur est réellement devant).

Can recceive from the same lighting



Gestion performante des ennemis et des effets

L'un des enjeux techniques majeurs de notre projet repose sur la capacité à afficher, animer et gérer en temps réel des centaines, voire des milliers d'objets actifs, tout en conservant une fluidité constante sur des machines cibles grand public. Pour atteindre cet objectif, nous avons mis en œuvre une approche combinant deux stratégies complémentaires : l'utilisation systématique d'un système de pooling et la répartition des traitements lourds sur plusieurs frames.

Pooling : éviter l'instanciation coûteuse

Plutôt que d'instancier ou détruire dynamiquement des objets en jeu (ennemis, VFX, projectiles énergétiques...), nous avons centralisé leur gestion dans des managers de pool spécialisés. Ces gestionnaires conservent une réserve d'objets inactifs, prêts à être réutilisés à tout moment. Trois composants principaux illustrent cette approche :

S_EnemyPoolManager : récupère ou recycle des ennemis selon leur type (défini via EnemyType). Lorsqu'un ennemi meurt, il est désactivé et replacé dans la file d'attente correspondante plutôt que détruit

S_VFXPoolManager : permet d'activer des effets visuels (explosions, impacts, halos...) en les réutilisant depuis un stock préchargé. Une fois le délai d'affichage écoulé, les objets sont désactivés et remis en file d'attente

S_EnergyPointPoolManager : utilisé pour les objets d'énergie générés pendant le gameplay. Chaque activation est traitée comme une demande mise en file, puis réutilisée ou instanciée si besoin

Ce système évite ainsi les coûts de création/destruction fréquents dans le heap, réduisant drastiquement les allocations mémoire et le déclenchement du garbage collector, principal responsable de freezes inattendus dans Unity.

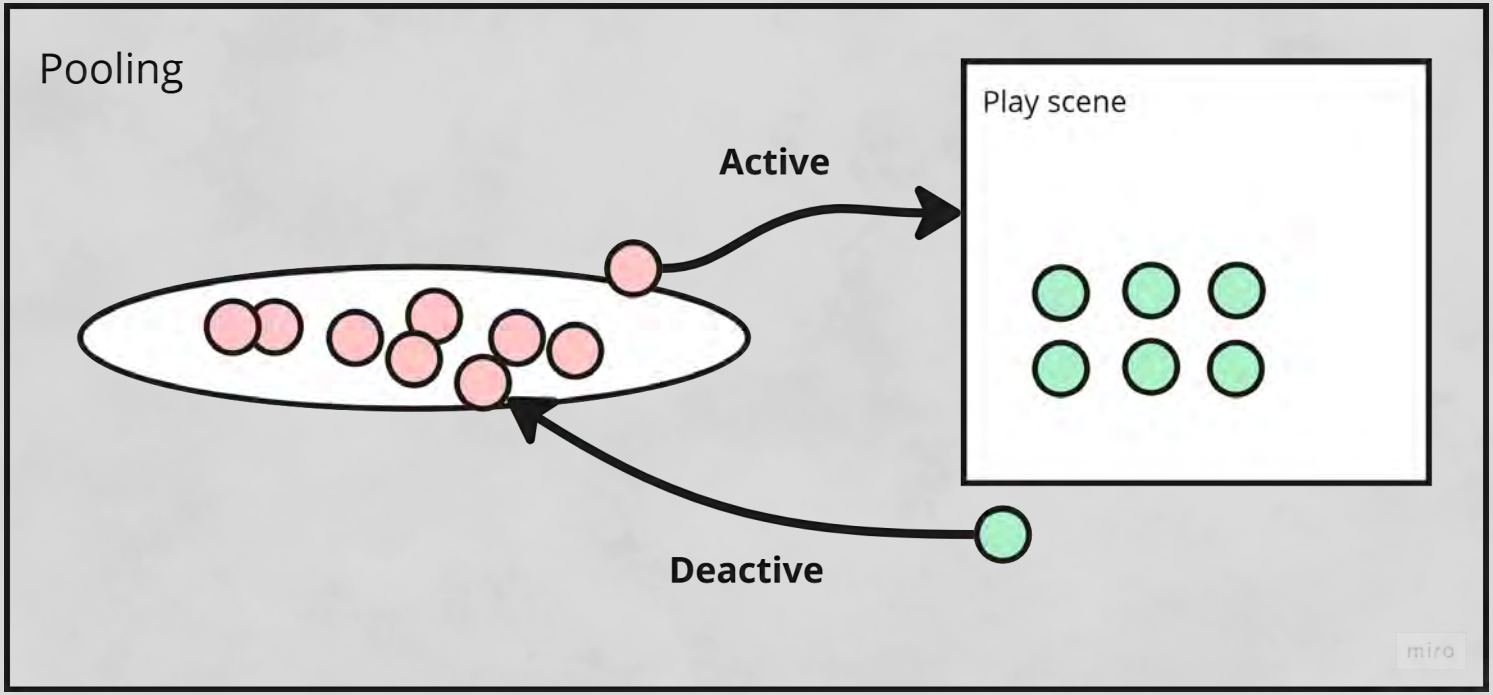
Répartition temporelle : lisser la charge sur plusieurs frames

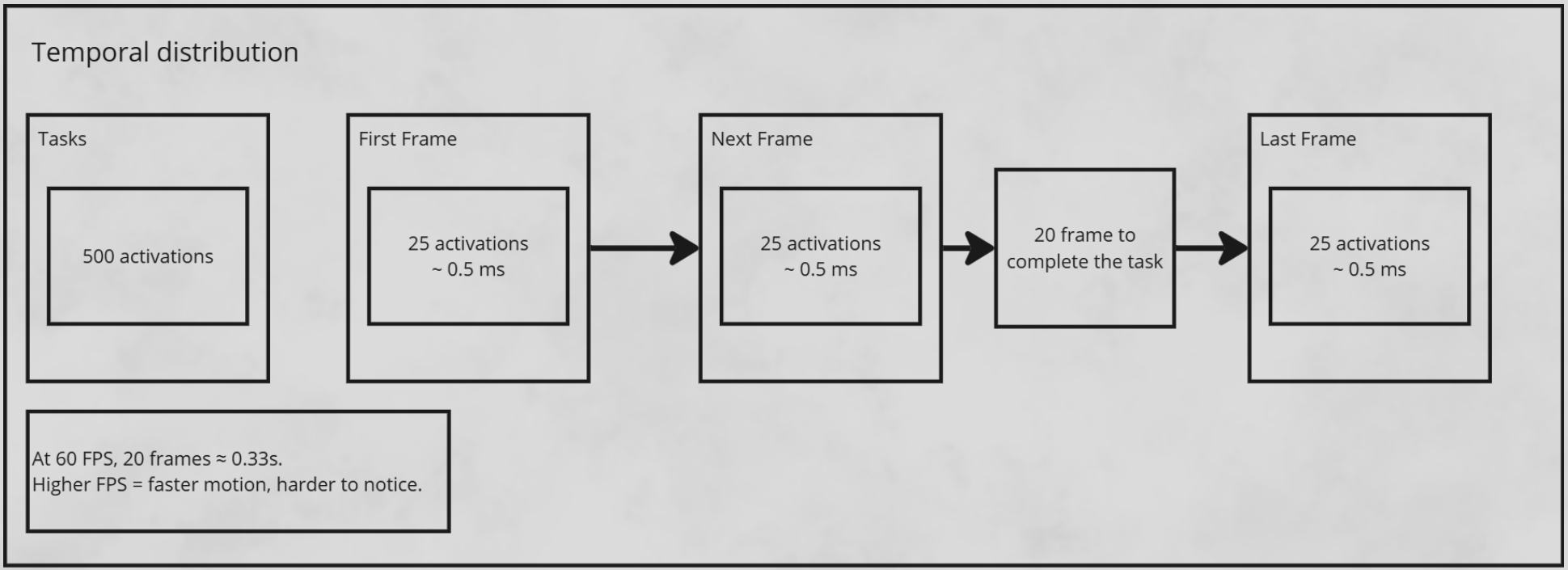
La deuxième stratégie clé consiste à répartir intelligemment les traitements sur plusieurs frames.
Au lieu de traiter toutes les demandes de création en une seule frame (ce qui peut provoquer un pic de charge CPU ou GPU), chaque pool manager limite volontairement le nombre d'activations autorisées par frame (maxActivationsPerFrame).

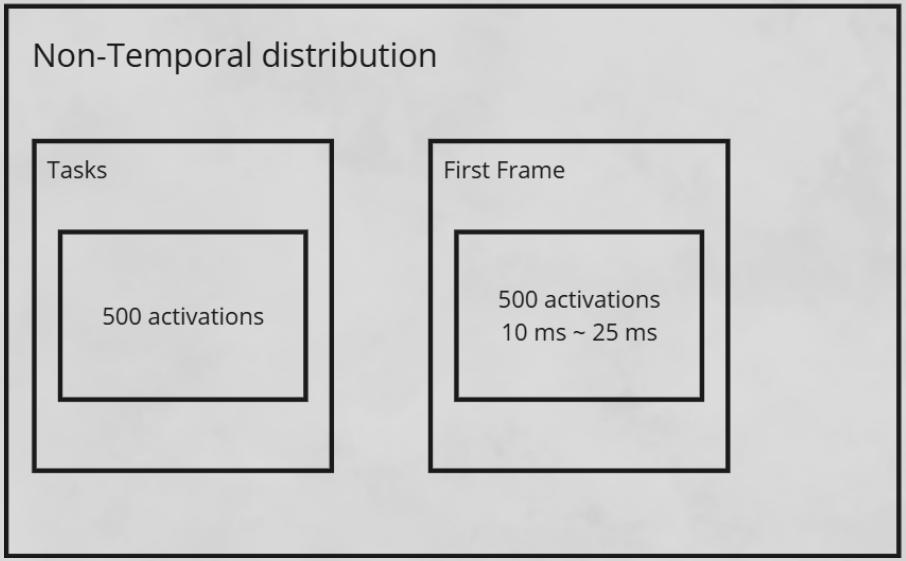
Par exemple :
S_EnergyPointPoolManager limite à max 25 activations par frame d'objets énergétiques

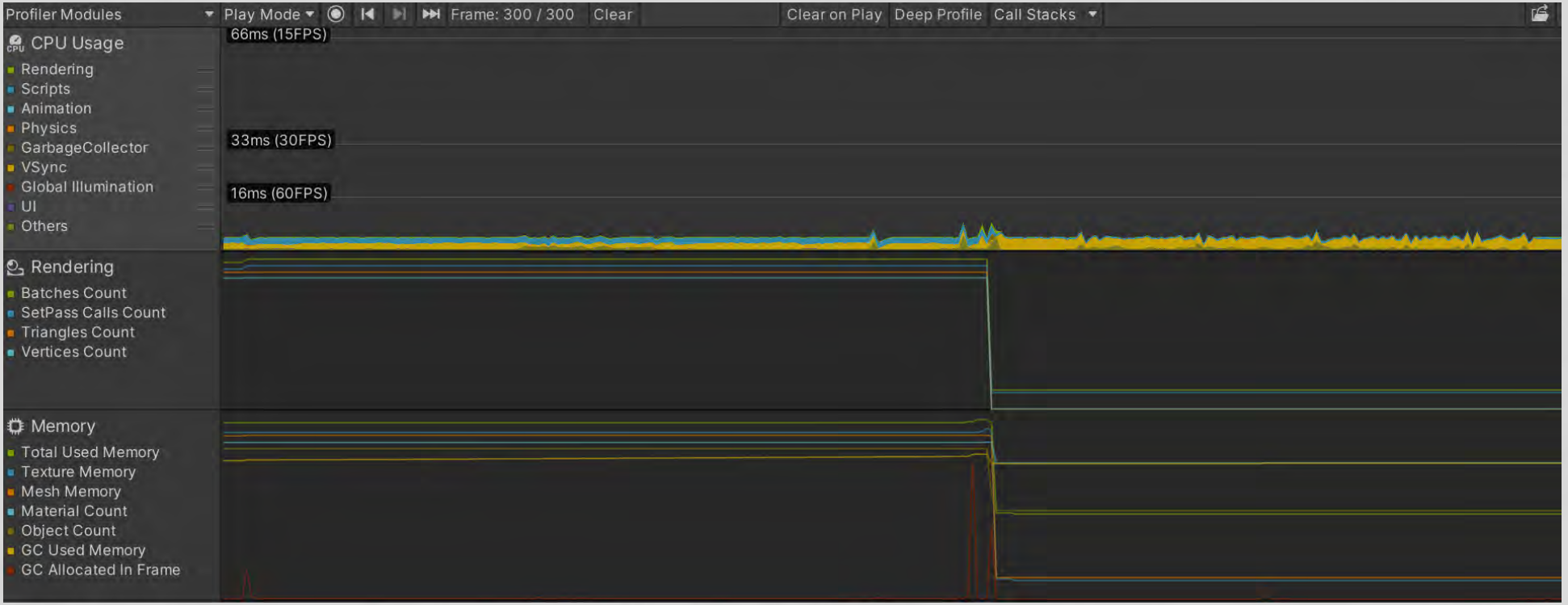
S_VFXPoolManager traite jusqu'à 35 effets visuels par frame, ce qui est suffisant pour les pics d'intensité visuelle sans surcharger le rendu

Cette approche contrôle activement la densité des traitements graphiques ou logiques à l'intérieur d'un seul cycle Update, et empêche que l'arrivée massive d'ennemis ou d'explosions n'entraîne une chute brutale du frame-rate.




Temporal distribution
Tasks
500 activations
First Frame
25 activations
~ 0.5 ms
Next Frame
25 activations
~ 0.5 ms
20 frame to
complete the task
Last Frame
25 activations
~ 0.5 ms
At 60 FPS, 20 frames ≈ 0.33s.
Higher FPS = faster motion, harder to notice.


Non-Temporal distribution
Tasks
500 activations
First Frame
500 activations
10 ms ~ 25 ms


Profiler Modules
Play Mode
Frame: 300 / 300
Clear
Clear on Play
Deep Profile
Call Stacks
CPU Usage
Rendering
Scripts
Animation
Physics
GarbageCollector
VSync
Global Illumination
UI
Others
66ms (15FPS)
33ms (30FPS)
16ms (60FPS)
Rendering
Batches Count
SetPass Calls Count
Triangles Count
Vertices Count
Memory
Total Used Memory
Texture Memory
Mesh Memory
Material Count
Object Count
GC Used Memory
GC Allocated In Frame

Génération continue d'ennemis avec logique adaptative

Le système d'apparition des ennemis est entièrement géré via S_SpawnRequester, qui repose sur une coroutine asynchrone par type d'ennemi. Chaque coroutine observe les paramètres suivants :

1. Le niveau du joueur (via S_EnergyStorage)
2. Le nombre d'ennemis actifs (S_EnemyTracker) pour éviter la surcharge
3. Une AnimationCurve par type, qui définit une fréquence d'apparition dynamique en fonction de la densité d'ennemis déjà présents dans la scène

Cette logique permet d'adapter dynamiquement la pression ennemie en fonction du rythme du joueur, tout en garantissant que le système ne génère jamais plus d'unités que le moteur ne peut gérer.

Suivi de charge en temps réel

Enfin, un gestionnaire auxiliaire S_EnemyTracker maintient à jour, de manière extrêmement légère, une base de données des ennemis actifs par type. Cela permet de :

1. Déclencher ou ralentir l'apparition d'un type spécifique
2. Nettoyer régulièrement les références à des objets détruits
3. Offrir un retour instantané au game designer sur la densité d'ennemis pendant une phase donnée

Conclusion

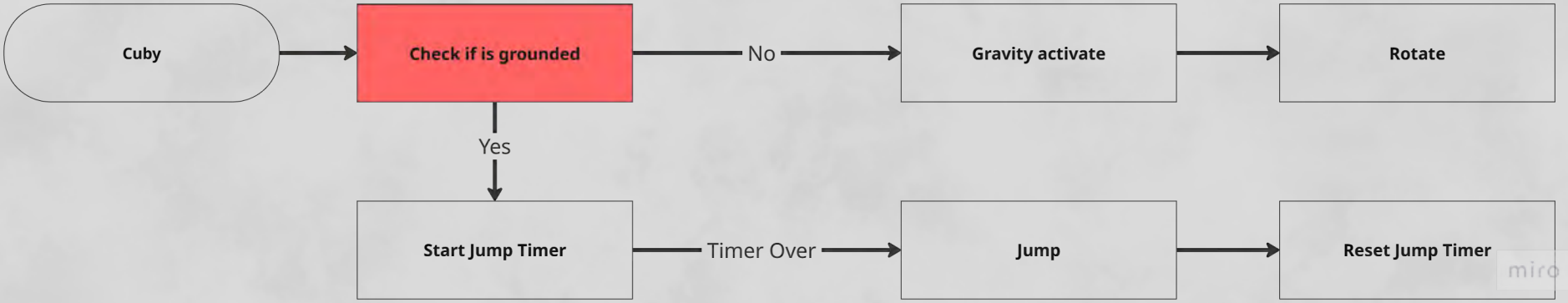
Grâce à une combinaison de pooling strict, de traitement par lot limité par frame et d'un contrôle intelligent du rythme d'apparition, notre projet parvient à gérer des scènes riches de plusieurs milliers d'objets dynamiques sans perte de performance ni instabilité. Ce choix architectural constitue l'un des piliers techniques du projet, et illustre notre volonté de concevoir un gameplay à grande échelle fluide, contrôlé et évolutif.

Les ennemis :

Le Cuby vérifie s'il est au sol :

- Au sol : le timer de saut démarre
- En l'air : il devient soumis à la gravité et exécute une rotation

Si le timer de saut termine, il saute en direction du joueur. S'il collisionne avec le joueur, lui inflige des dégâts.

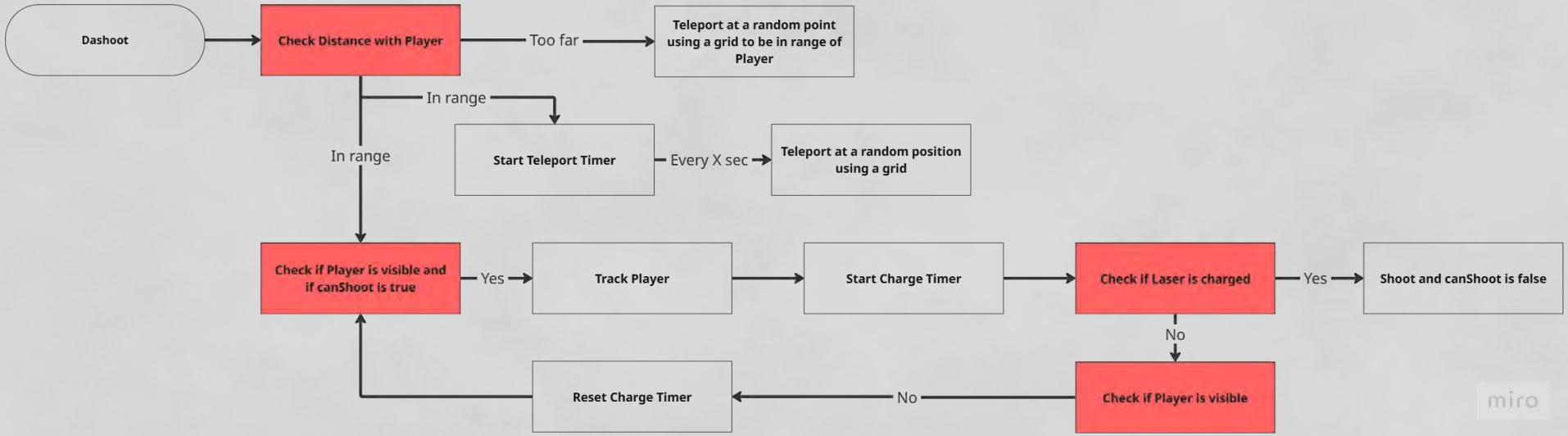


Toutes les X secondes au sol, le Dashoot vérifie la distance entre lui et le joueur :

- Trop loin : il cherche autour du joueur un point où il pourra se téléporter pour être à portée du joueur sans être trop proche ni trop loin
- À portée : il cherche dans une grille autour de lui un point valide, lui permettant de rester à une distance raisonnable du joueur :
    - Trop proche : s'éloigner,
    - Trop loin : se rapprocher,
    - Bonne distance : se déplace vers la gauche/droite

S'il voit le joueur et que canShoot est true, il le traque, démarre un timer de charge, puis tire un laser prédictif une fois chargé. Il vérifie l'objet touché par le laser et si le joueur est touché, lui inflige des dégâts.

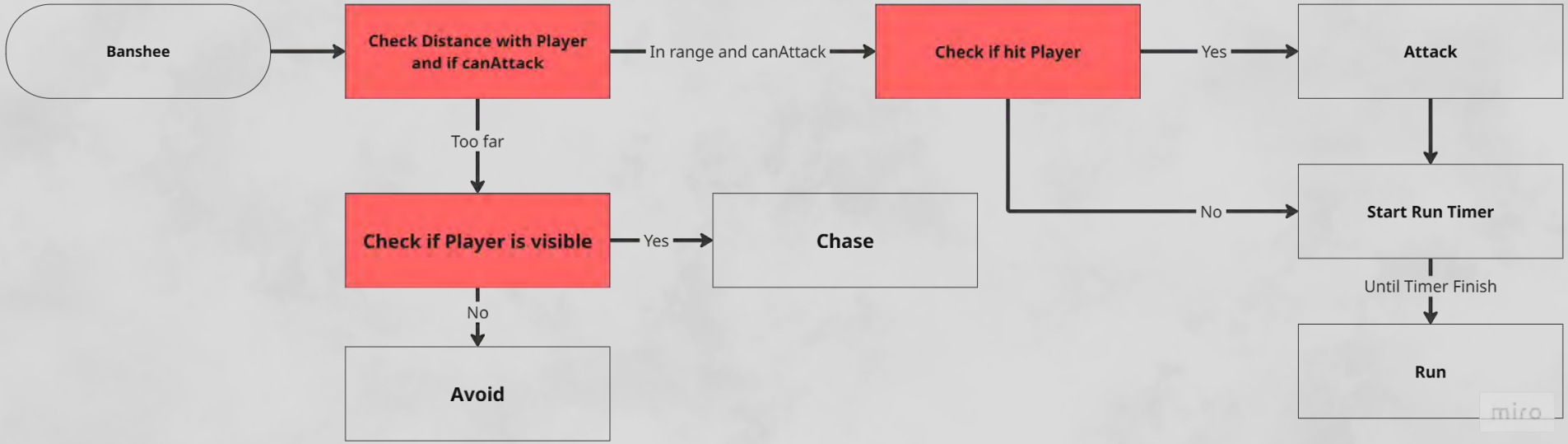
Si la visibilité est perdue pendant la charge, le timer est réinitialisé. Après le tir, canShoot devient false. Il ne peut tirer qu'une fois par téléportation.



Le Banshee vérifie sa distance entre lui et le joueur :

- Trop loin : se déplace et s'oriente vers le joueur
- À portée d'attaque : il attaque le joueur

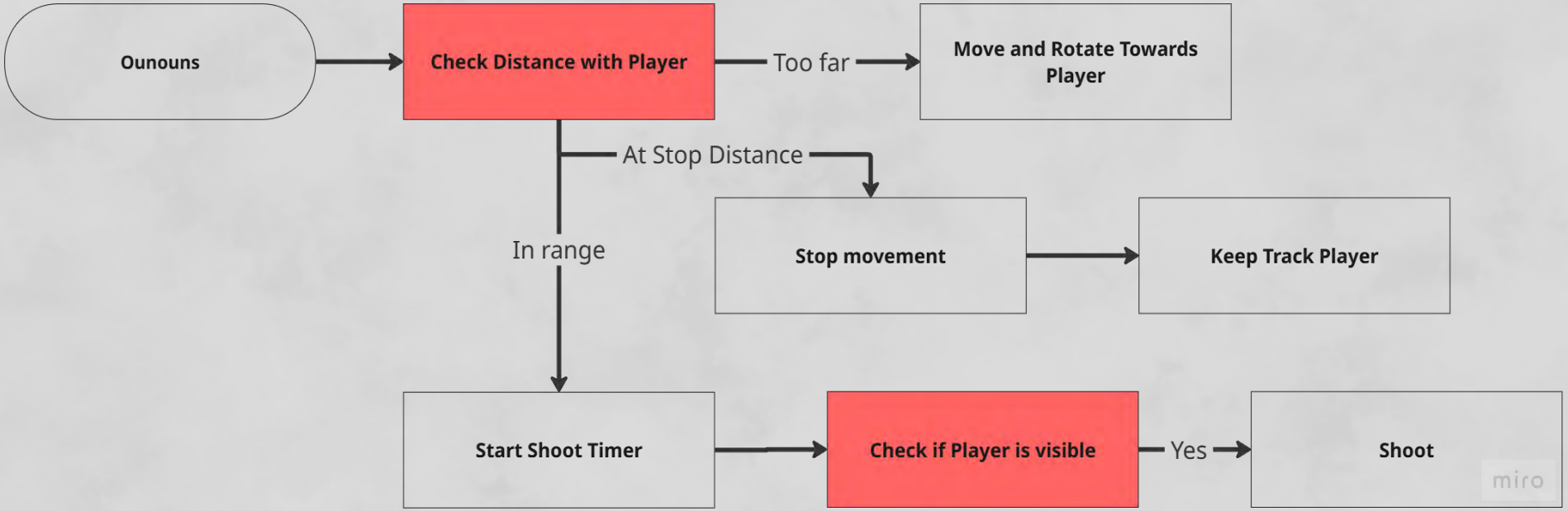
Après l'attaque, un timer de fuite démarre et fuit en direction opposée pendant la durée du timer. S'il y a un obstacle pendant sa chasse, il essaie de l'éviter.



Le Ounoun se rapproche du joueur jusqu'à une distance d'arrêt :

- Trop loin : se déplace et s'oriente vers le joueur
- À distance d'arrêt : il s'arrête et continue à suivre la position du joueur

Si le joueur est dans la portée de tir, un timer de tir démarre. Si le joueur est visible et le timer se termine, il tire.



Objectif Manager :

L'Objectif Manager est constitué de 4 éléments dont 3 composants de fonctions et 1 composants pour l'UI :

- Objective Class : il gère la définition d'un objectif, son état, les fonctions de progression, ainsi que les events liés (complétion, mise à jour, etc).

- Objective Manager : il contient la liste des objectifs actifs et sert de point d'accès pour ajouter, supprimer et interagir avec les objectifs de jeu.

- Objective : le script de l'objectif, définie les paramètres spécifique (description, condition de réussite, déclencheurs), l'instancie dans le Manager et met à jour l'état de l'objectif via les events.

- Objective UI : il gère l'affichage visuel des objectifs à l'écran et reçoit les events depuis la Class pour mettre à jour l'affichage.

Pour le fonctionnement, le script de l'Objective doit instancier et ajouter l'objectif dans le Manager puis utilise les fonctions et les events de la Class pour gérer l'état d'avancement de l'Objective.

Ce système est un élément important pour la gestion du score et des conditions de victoire.

Objective Class

C# Events :
Action OnComplete
Action OnValueChanged

Properties :
ClassSetter (Objective(string eventTrigger, string statusText, int maxValue))
ClassGetter
readonly string statusText

Methods :
CheckCompletion()
AddProgress()
GetStatusText()

Objective Manager

C# Events :
Action OnObjectiveAdded

Properties :
Objective list
Dictionnary

Methods :
AddObjective()
AddProgress()

Objective UI

UI Properties :
TextMeshProUGUI objectiveText

Properties :
Objective objective

Methods :
Init()
OnObjectiveComplete() (Overline objective when complete)
OnObjectiveValueChange() (Update objective value)

Objective

Properties :
Objectives parameters
ObjectiveDisplay
ObjectiveManager
Objective

Methods :
Start() : Create Objective + Setup UI and Trigger method
AddProgress()

Direction Artistique Visuelle

Intention :

L'expérience que nous souhaitons transmettre à travers **KILL DE'MON** est celle d'un joueur surpuissant, une menace absolue évoluant dans un monde où il incarne le danger.

Il ne doit pas se sentir comme une victime luttant pour survivre, mais plutôt comme un prédateur implacable, dominant et destructeur, capable d'anéantir tout ce qui se dresse sur son chemin.

Pour amplifier ce sentiment de toute-puissance, le joueur dispose de pouvoirs magiques évolutifs, dont la puissance fluctue pour dynamiser le gameplay et lui faire ressentir différents états de puissance. Cette mécanique alterne entre des moments de frénésie brutale et des phases plus stratégiques, renforçant ainsi l'impact et la satisfaction de chaque action.

L'environnement doit justifier cette destruction massive et permettre au joueur de la pratiquer librement. Le joueur doit pouvoir anéantir ses ennemis avec une brutalité, et pour que cet acte soit viscéral et percutant, leur apparence doit être pensée pour ce type de mise à mort.

Nous avons choisi l'enfer et les démons comme cadre principal, car ils incarnent un univers sale, organique et ensanglanté, où la chair est vouée à être déchiquetée. Ce monde chaotique et oppressant renforce l'idée que le joueur ne lutte pas pour survivre, mais pour dominer et exterminer.

Image Gameplay du jeu

Rappel S1

Lors du premier semestre, la direction artistique du projet était encore au stade de design et de réflexion. Des choix importants marquant la suite de la production ont été décidés, et des veilles de pratiques ont été produites. En voici un rapide résumé :

Références graphique :

Notre univers visuel et gameplay s'inspire d'une sélection de jeux où puissance, énergie et destruction sont au cœur de l'expérience.
Nous puisons d'abord dans **Devil Daggers** et **Hyper Demon**, qui incarnent une esthétique infernale et impitoyable. Leur ambiance démoniaque et leurs combats nerveux nourrissent notre vision d'un enfer brutal et déstructuré, où notre personnage puissant utilise une énergie surnaturelle pour annihiler ses ennemis avec une efficacité redoutable.

L'influence de **Doom** se traduit par la fluidité et la puissance du gameplay, avec un focus sur l'action rapide et la précision létale. Cette inspiration nous aide à créer une sensation de rage contrôlée, où chaque attaque doit être maîtrisée.

Pour la diversité visuelle et l'identité des ennemis, nous nous inspirons particulièrement de **Risk of Rain 2**. Son bestiaire varié et son style unique nous permet d'enrichir notre univers avec des créatures aux apparences distinctes et marquantes, renforçant la richesse visuelle et l'immersion.

Enfin, **Boomerang X** influence notre approche de la mobilité et du maniement précis d'une arme centrale, ici, l'énergie destructrice de notre personnage, renforçant l'efficacité et la dynamique du combat.

En combinant ces références, nous bâtissons un univers où l'enfer est à la fois chaotique et maîtrisé, où la puissance brute et l'efficacité sont les clés pour dominer un bestiaire varié dans un ballet infernal d'énergie et de destruction.


HYPER
DEMON


DEVIL DAGGERS


DOOM

Image Gameplay Risk of Rain 2

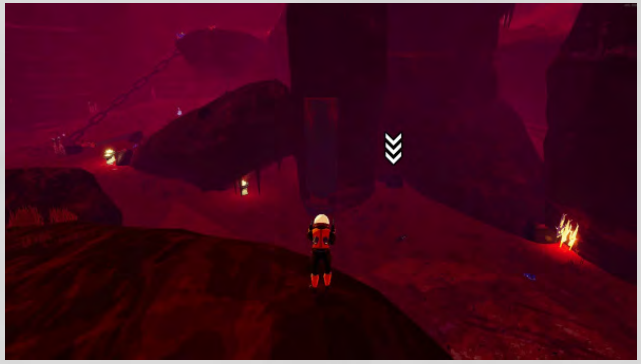
Image Gameplay BoomRang X

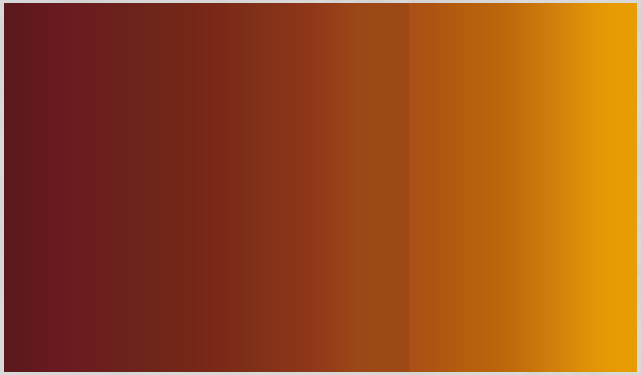
Ambiance Chromatique :

Grâce au moodboard et aux références sélectionnées, nous avons défini une ambiance chromatique cohérente, destinée à unifier l'ensemble de la direction artistique du jeu. Cette palette de couleurs vise à créer une atmosphère immersive et percutante, qui s'étendra sur l'ensemble du jeu. Avec notre thème démoniaque et onirique, il était naturel de nous orienter vers des teintes sombres et rougeâtres, renforçant ainsi l'aspect oppressant et viscéral de l'univers. Pour affiner cette direction, nous avons rassemblé plusieurs références visuelles servant de base pour établir cette palette chromatique.

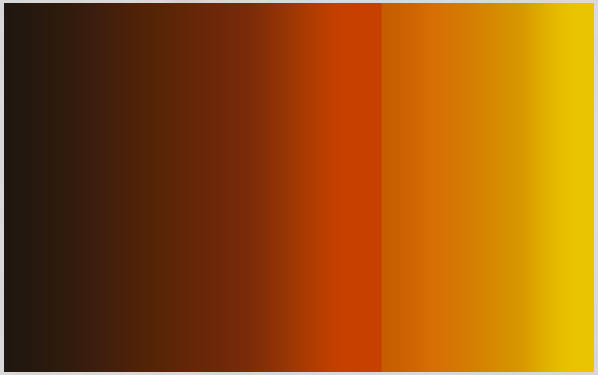
Ces palettes de couleurs servent de référence principale pour la conception des environnements et ingrédients du jeu. Elles permettent de maintenir une cohérence esthétique globale, en harmonisant les teintes utilisées dans les décors et l'atmosphère visuelle.

En s'appuyant sur ces couleurs, nous garantissons une identité forte et immersive, plongeant le joueur dans un monde à l'ambiance sombre et pesante, en accord avec notre direction artistique.





Gradient Blessed Burden



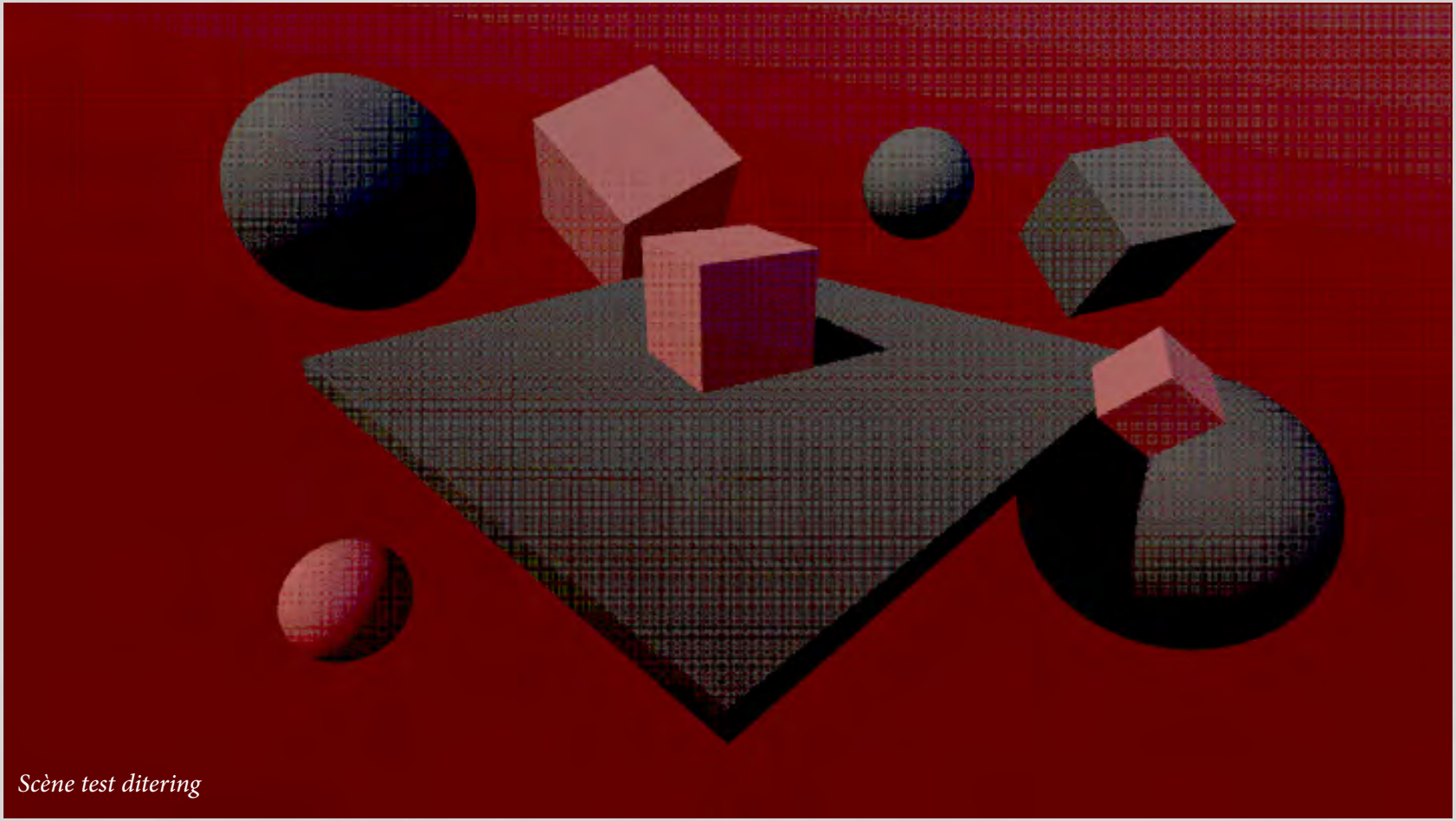
Gradient Devil Dager

Gradient Risk Of Rain 2

Shader Dithering (Intention de cassé l'image)

Pour renforcer l'ambiance infernale et chaotique de notre jeu, nous avions l'intention d'intégrer des shaders, dont l'un des plus marquants était le dithering. Cet effet permet de créer des textures, déstructurer l'image et créer du contraste entre les couleurs, accentuant ainsi l'aspect sombre et oppressant de l'environnement.

Visuellement, le dithering apporte un effet rétro et low-poly, conférant au jeu un style brut et distinctif. En plus de son impact esthétique, il optimise les dégradés et les ombres, rendant les scènes plus dynamiques, même dans des environnements très sombres.

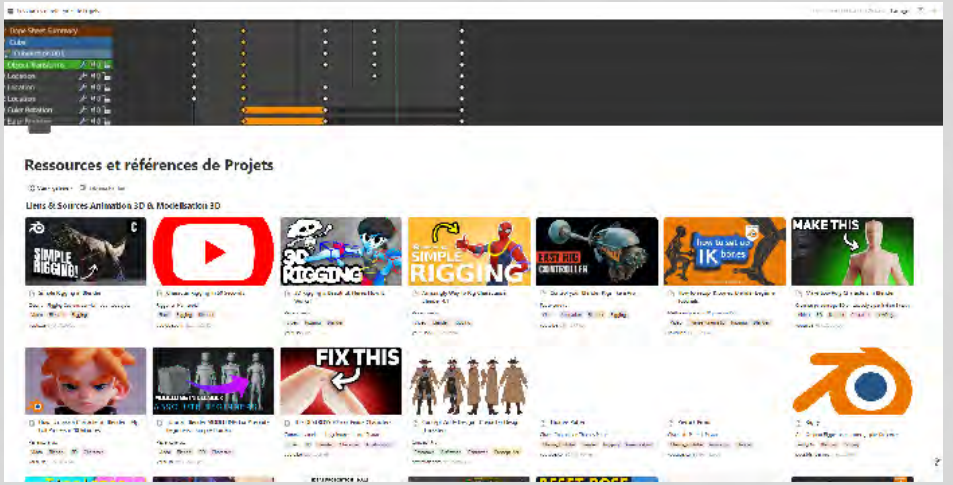


Scène test ditering

Études de l’animation

Pour optimiser notre processus de création d’animations, nous avons mis en place une veille dédiée à l’animation 3D. Nous avons utilisé le bloc «Galerie» de Notion pour recenser et organiser les ressources les plus pertinentes en matière d’animation.

Les fiches Notion sont taguées selon leur contenu, ce qui permet de filtrer rapidement les ressources en fonction des besoins spécifiques du moment. Chaque fiche est accompagnée d’une description résumant le contenu et les points clés, facilitant ainsi l’apprentissage et la pratique de l’animation. Cette base de données a permis de s’exercer à l’animation sur Blender en appliquant différentes techniques sur divers éléments. L’animation ne se limite pas à Blender, mais inclut également l’exportation vers Unity, où de nombreux paramètres doivent être maîtrisés. Les vidéos référencées nous ont permis de mieux comprendre ces réglages et d’optimiser notre workflow.



Notion veille animation

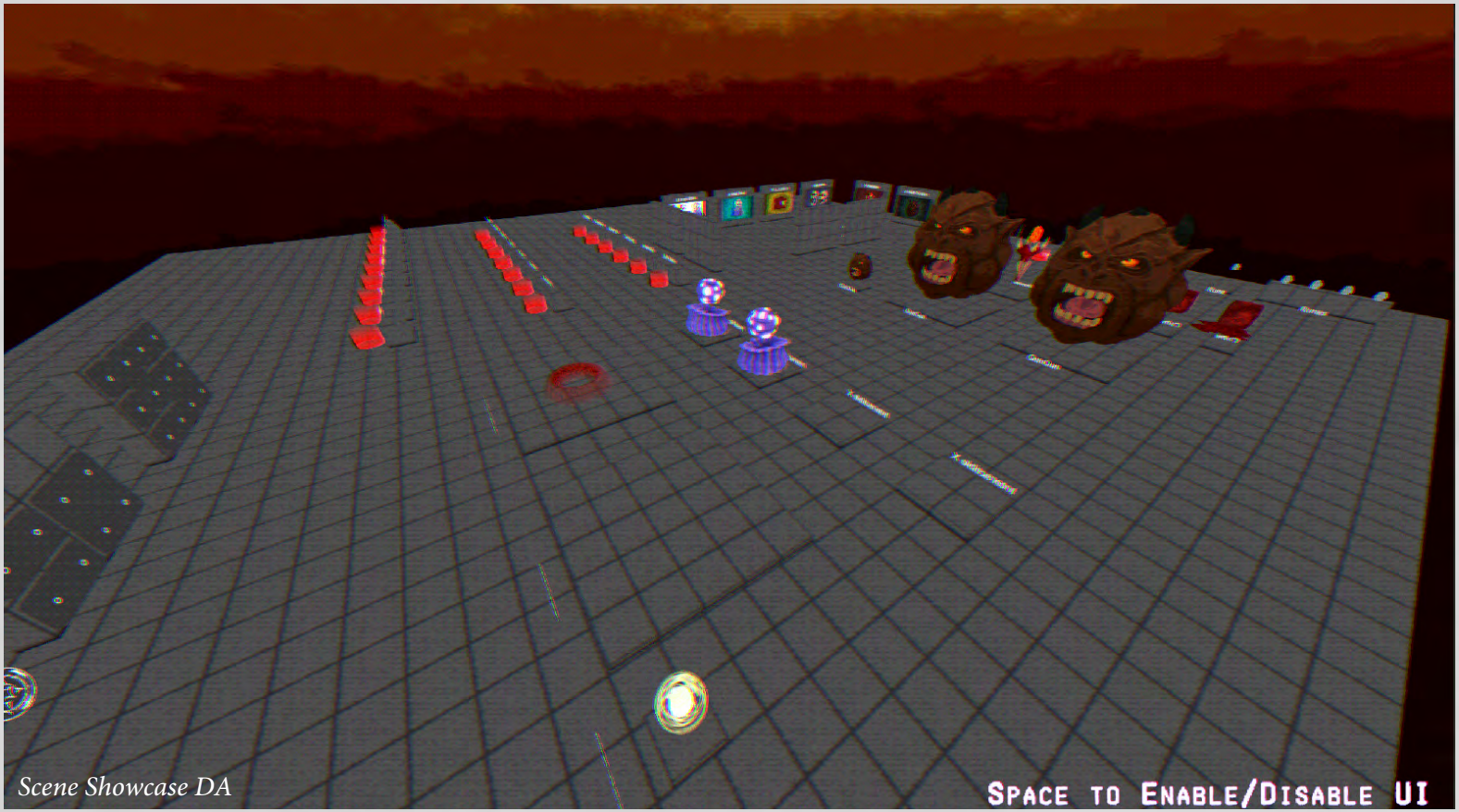
Scene Test Anim

Production S2 :

Scène ShowcaseDA :

Pour s’assurer une Direction Artistique uniforme et donc un univers cohérent, nous avons mis en place une scène Unity nommée “ShowcaseDA” permettant d’y placer nos différents ingrédients côte à côte.

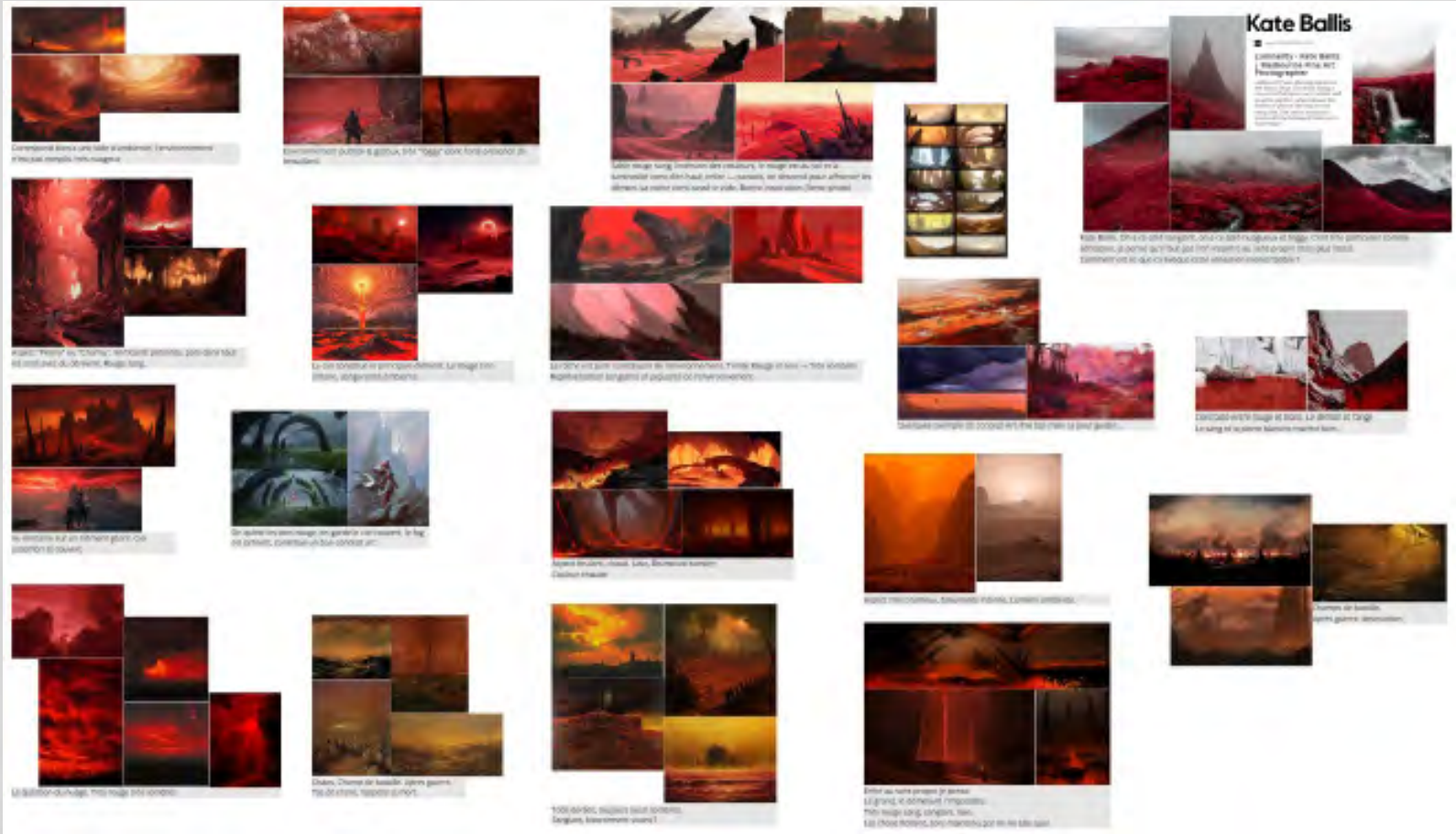
Cette proximité entre les éléments du jeu permet d'identifier des différences qui seraient beaucoup trop marquées et de les réadapter pour améliorer cette cohérence recherchée. On y retrouve les modélisations 3D, les feedbacks sonores et visuels (VFX et SFX).

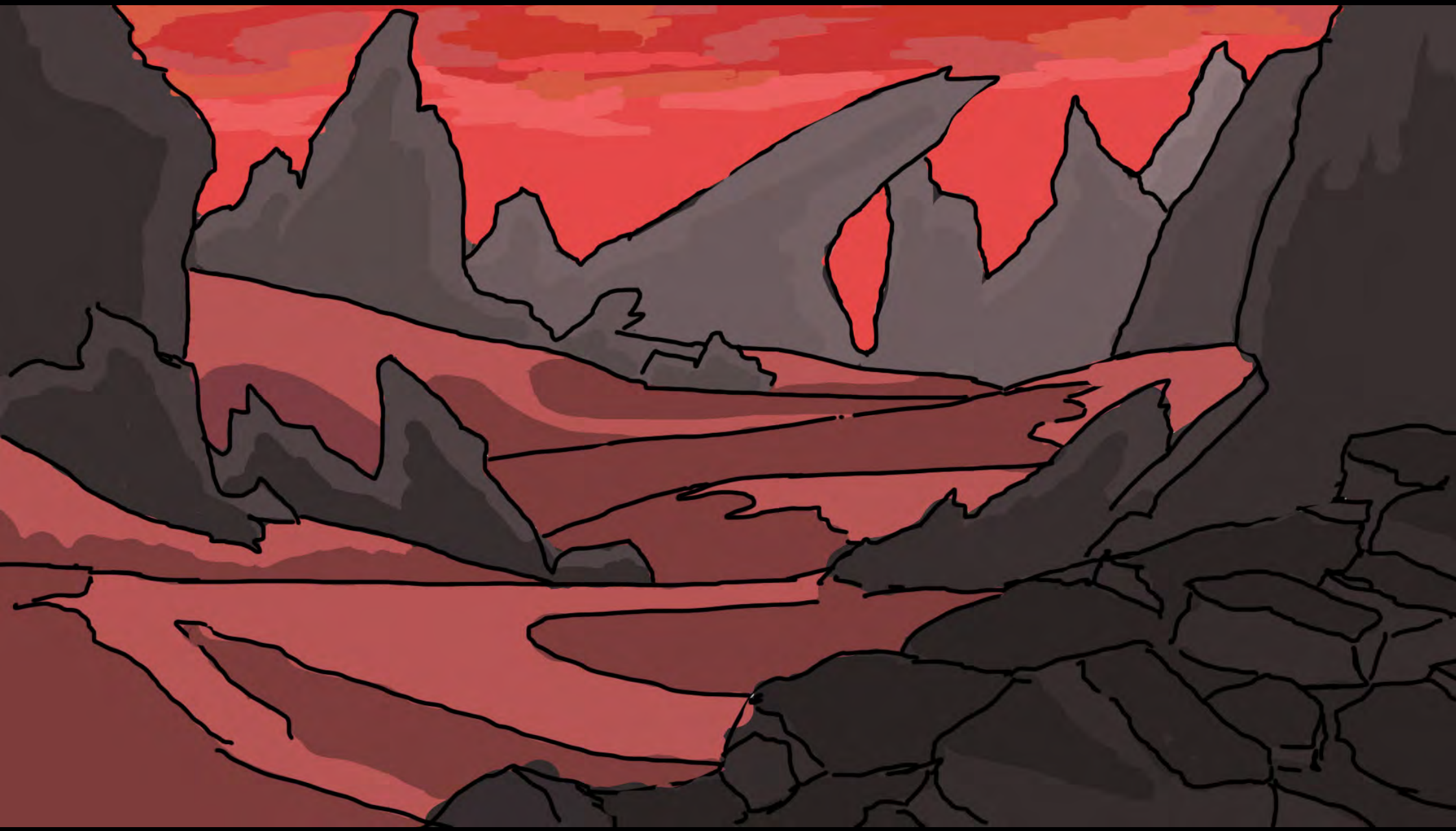


Scene Showcase DA

Ambience Global

Pour nous permettre de nous projeter dans la DA et particulièrement dans l'environnement qui constituait le potentiel espace de jeu, nous avons produit un Concept Art d'environnement. Ce dernier nous a permis de dégager des idées majeures, comme la texture des éléments dans le jeu, par exemple.





Concept Art Ambience Prototype

Concept Art Ambience Final

Modélisation 3D

Particularité des ennemies

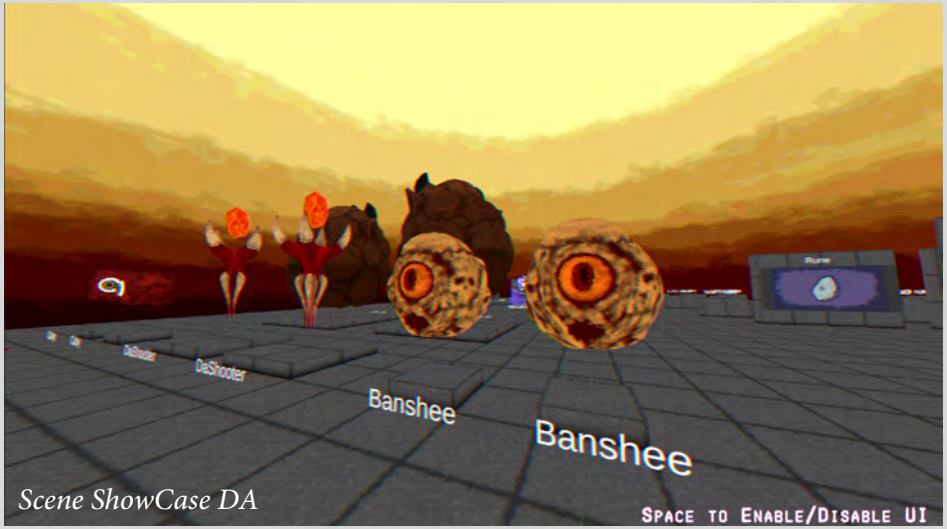
Les ennemis dans notre jeu adoptent des formes géométriques simples : cubes, cylindres et sphères. Ce choix de design n'est pas uniquement esthétique, il est motivé par des raisons pratiques et fonctionnelles :

Lisibilité visuelle : En utilisant des formes nettes et différenciées, le joueur peut facilement identifier les types d'ennemis, même à distance ou en plein mouvement. Cela améliore la clarté de l'action et permet une prise de décision plus rapide.

Simplicité de modélisation : Les formes simples réduisent la complexité de la modélisation 3D, ce qui permet un gain de temps en production tout en assurant des performances optimales en jeu.

Cohérence stylisée : Ce choix s'intègre dans une direction artistique minimaliste et cohérente, où chaque forme peut être associée à un comportement ou une menace spécifique.

En résumé, l'usage de formes géométriques de base pour les ennemis est un choix stratégique de gameplay et de production, visant à équilibrer clarté, performance et identité visuelle.



Scene ShowCase DA

Scene ShowCase DA

Runes

Les runes dans le jeu sont la représentation des capacités que le joueur peut exécuter. L'idée initiale était que la pierre s'illuminait en fonction de sa charge (soit le niveau du joueur) et que des craquelures seraient apparues au niveau maximal de charge pour évoquer un afflux d'énergie. Le concept des runes posait des problèmes physiques, puisque ces dernières, devant se situer sur les bras du joueur, auraient eu des problèmes de visibilité à cause des animations des bras. Ainsi, les runes sont passées d'objets physiques à un objet UI plus facile à maîtriser.

Boules d'énergie

Les boules d'énergie constituent la ressource principale du jeu puisqu'elles permettent au joueur de tirer et d'utiliser ses compétences. On les représente par une sphère en mouvement car elle crée du dynamisme et ajoute de la vie à cet objet, pourtant statique dans la scène.

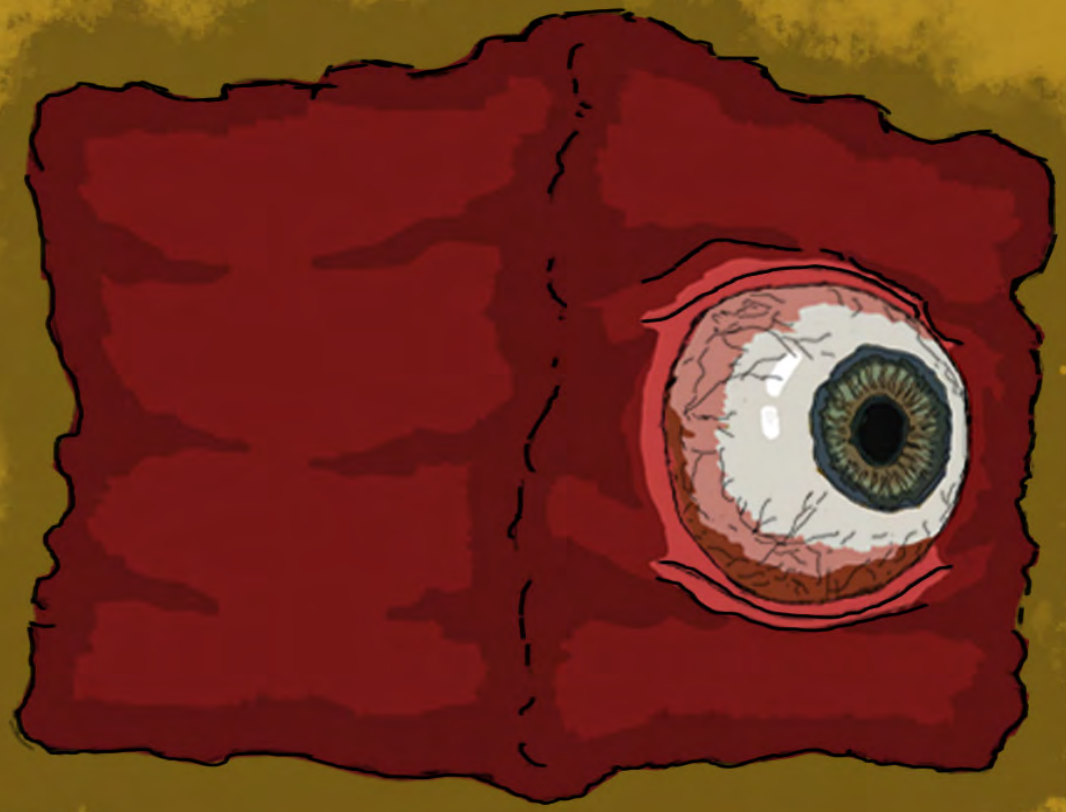
Gantelet

Le gantelet est l'arme principale du joueur, un artefact magique capable d'accumuler et de libérer de l'énergie pour alimenter ses attaques.

Il agit comme un canalisateur de puissance, amplifiant les capacités du joueur à mesure que l'énergie s'accumule.

Lorsqu'il est chargé, le gantelet augmente progressivement en puissance, débloquant de nouvelles capacités. En absorbant davantage d'énergie, il peut activer des runes supplémentaires, améliorant ainsi ses pouvoirs et son impact en combat.

Son design robuste et énigmatique traduit cette montée en puissance, soulignant son rôle central dans le gameplay et l'identité du personnage.



Cuby

Le Cuby constitue la première menace que le joueur rencontre lorsqu'il lance une partie. On remarque son design très simpliste, très peu caractérisé, ce qui accentue sa faiblesse et traduit son comportement basique : il saute. Cet ennemi est bien évidemment présent dans les paliers supérieurs sous forme de variante avec une taille plus grande, indiquant premièrement un marqueur visuel d'une force augmentée et d'un gain d'énergie plus élevé.

Dashooter

Le Dashooter est le deuxième ennemi que le joueur rencontre. Il est d'abord présent au premier palier en petits échantillons, puis beaucoup plus au deuxième palier. Son comportement impacte grandement son design, puisque ce dernier est un ennemi statique qui se téléporte et dont l'attaque est représentée par un laser qu'il tire sur le joueur. Pour créer une transition suivant le Cuby, son corps garde un aspect charnier accompagné d'une structure osseuse. Cette forme de tour nous a apporté l'idée d'une gemme au-dessus de ce dernier, créant un point de tir pour le laser.

Banshee

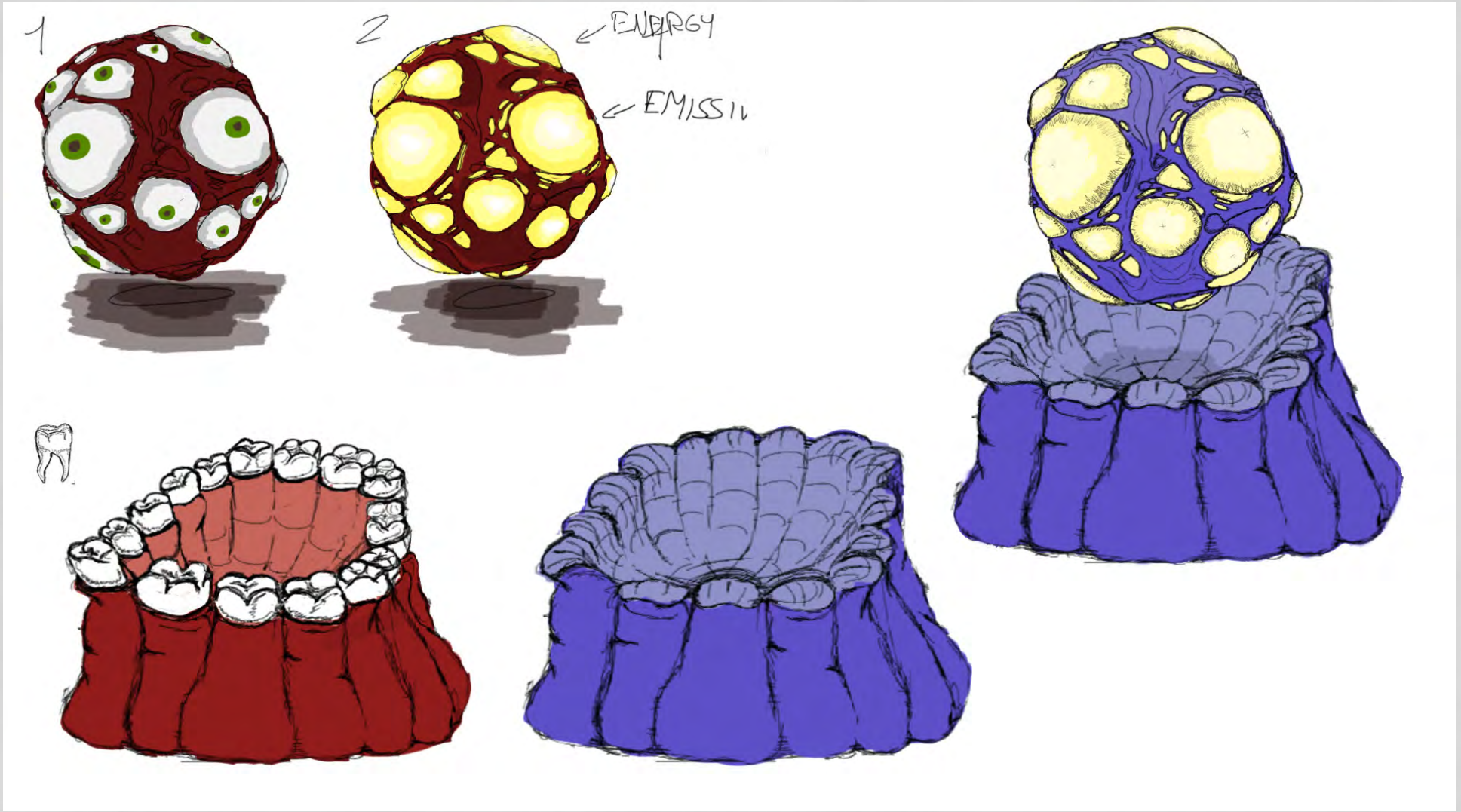
Jusqu'à présent, les ennemis sont majoritairement présents au sol, ces derniers n'exploitant que très peu le saut du joueur. La banshee vient pallier ce souci, son design très sphérique lui permet de garder une forme simple à différencier. D'ailleurs, sa forme ronde a permis de lui créer une caractéristique fortement présente chez nos ennemis : un gros œil globuleux, ce qui permet de garder une cohérence visuelle chez ces derniers.

OunOun

Le OunOun est le dernier ennemi du jeu, son comportement et sa puissance en jeu ont grandement influencé son apparence puisqu'il a des traits physiques très marqués, menaçants, et que sa taille est trois fois plus grande que celle des autres. D'ailleurs, sa taille traduit parfaitement son taux de drop d'énergie, grandement supérieur aux autres.

FeedSeed

La FeedSeed est le premier ingrédient passif du jeu, elle est statique dans la map en attendant d'être détruite. Lorsque le joueur lui tire dessus, elle relâche l'énergie contenue dans son bulbe. Parmi les designs produits, 2 ont retenu notre attention de par leur apparence :



FeedSeed

Nous avons opté pour une apparence plus organique avec une couleur très violette, permettant de contraster avec le décor et permettant au joueur de bien la remarquer.

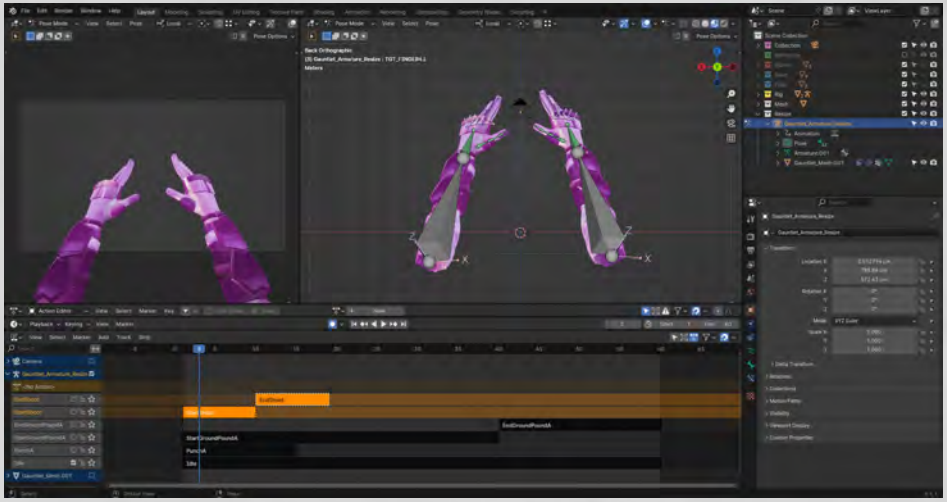
Animation & Rig

Animation

Les animations ont été conçues pour traduire visuellement une sensation de force, de puissance et de rapidité. On a opté pour un style rigide et nerveux, qui colle à l'identité du jeu : des gestes secs, percutants, sans superflu, qui appuient l'efficacité de l'action.

L'objectif n'était pas de viser une animation fluide ou réaliste, mais plutôt de renforcer le gameplay en offrant des feedbacks clairs et impactants. Chaque animation vient soutenir la lecture du jeu, en accentuant le tempo et l'intensité des interactions.

Il n'y a pas eu de références précises utilisées : tout a été développé en interne en fonction des besoins du projet et de la direction artistique que notre équipe de 6 personnes a définie ensemble.



Rig et Animation dans Blender

RIG

Le rig utilisé dans le jeu est un rig maison, réalisé sur mesure en partant du modèle 3D des avant-bras du joueur, qui est le seul personnage animé du projet. L'approche a été simple mais fonctionnelle, avec quelques expérimentations (comme l'IK) rapidement abandonnées, car peu adaptées à notre configuration uniquement basée sur les avant-bras.
Le rig repose sur une base simple mais efficace, avec des contrôleurs dédiés pour les doigts, et des bonnes spécifiques servant à fixer les accessoires utilisés en jeu.
Le rig et les animations ont été faites par la même personne, ce qui a permis de garder une bonne cohérence entre les deux aspects, même sans pipeline complexe.
La seule contrainte que nous nous sommes imposée a été de maintenir une nomenclature claire et rigoureuse sur les bones, notamment pour distinguer les bras gauche et droit, en vue d'une intégration propre.

Post Processing :

Couleurs et éclairage

Dans notre scène, nous ne plaçons pas de sources lumineuses classiques, car la lumière modifie souvent les couleurs que nous définissons pour les objets 3D. Pour conserver l'exactitude des teintes voulues, l'éclairage de l'environnement ne repose pas sur la skybox, mais sur une couleur fixe. Cette méthode permet d'afficher les couleurs des objets telles qu'elles ont été pensées, sans altération liée à la lumière dynamique. Elle garantit également une uniformité dans les jeux de couleurs à travers la scène, ce qui renforce la cohérence visuelle globale du jeu.

Avant ambiant lighting

Après ambiant lighting

Intention :
Le post-processing occupe une place essentielle dans notre direction artistique : il ne sert pas à embellir l'image, mais à la casser. Dans la lignée du travail entamé avec le dithering au premier semestre, notre intention est de déconstruire l'image lisse et propre du rendu classique pour lui donner du caractère, de la texture, presque une rugosité visuelle. En salissant et en déstructurant volontairement l'image, on cherche à renforcer l'ambiance du jeu, à accentuer les tons et à rendre l'univers plus vivant, plus incarné.

Dithering :
Le choix d'utiliser du dithering dès le début du projet s'inscrit dans notre volonté de casser l'image trop nette ou trop lisse d'un rendu 3D classique. En introduisant volontairement une forme de «bruit» visuel, le dithering vient perturber la lecture des dégradés et uniformités, créant ainsi une texture plus brute, presque imparfaite. Cette approche nous permet de déstructurer l'image, de lui retirer un certain confort visuel pour plonger le joueur dans une ambiance plus marquée, en cohérence avec l'univers du jeu.

Posterize :
L'effet de posterize nous permet de simplifier les couleurs en réduisant le nombre de teintes visibles à l'écran. En écrasant les dégradés, on casse l'image lisse et réaliste pour obtenir un rendu plus dur, plus stylisé. Les transitions deviennent plus franches, les aplats plus marqués, ce qui donne un aspect visuel plus tranché, moins "propre". Cet effet renforce notre intention de déstructurer l'image et d'apporter une forme de violence visuelle qui colle à l'ambiance du jeu.

Color Split :
Le color split vient renforcer notre volonté de casser la netteté de l'image en introduisant un léger décalage entre les couleurs. Ce décalage crée une sensation de flou coloré, presque comme si l'image était instable. C'est un moyen de perturber la perception du joueur, de rendre l'image moins confortable à regarder, tout en apportant un effet visuel marquant. On l'utilise pour amplifier le côté déséquilibré de l'univers qu'on met en place.

Pixellisation (Render Scale Personnalisé) :
L'effet de pixellisation consiste à réduire la résolution de l'image pour créer un rendu plus granuleux, où les pixels sont bien visibles. Cela donne un aspect plus brut, qui renforce l'impression que l'image est "cassée". Plutôt que d'utiliser un shader dédié à la pixellisation, nous passons par le Render Scale de Unity, qui permet de diminuer la résolution globale du rendu avant son affichage. Ce choix technique permet d'appliquer cet effet à toute l'image, y compris l'UI, permettant ainsi une cohérence visuelle globale. La pixellisation est notamment l'effet majeur du jeu : c'est lui qui donne son style unique, et qui uniformise le reste des effets appliqués à l'image.

Dithering

Posterise

Color Split

Pixelisation

Skybox :

La skybox est une image panoramique composée d'un dégradé de couleurs par étapes, allant du rouge sombre au jaune clair. Ce contraste fort entre ces teintes reflète les ambiances chromatiques que nous avons définies en amont, symbolisant une transition entre une zone plus sombre vers une zone plus lumineuse. En tant que panorama sphérique, elle joue un rôle important pour recentrer le joueur dans l'espace, sans éléments flottants comme des nuages qui pourraient suggérer une profondeur ou un lointain. La skybox reste toujours visible et joue un rôle purement décoratif, en renforçant l'identité visuelle du jeu. Elle sert également de support à l'application des effets de post-processing (pixelisation, dithering, posterize), qui viennent accentuer l'ambiance globale en s'y appliquant.

UI

L'UI adopte une DA marquée, cohérente avec l'esthétique crasseuse, agressive et infernale du jeu. Elle n'est pas pensée pour être épurée, mais pour s'imbriquer dans l'univers visuel global, jusqu'à être partiellement «polluée» par les effets de post-process. Ce choix volontaire gomme les frontières entre HUD et environnement, renforçant l'immersion et l'agressivité visuelle.

Jauge d'énergie (gauche) : une barre verticale indique l'énergie du joueur. Elle évoque une montée en tension constante. La barre change de couleur en montant de pallier (du jaune vers le rouge), créant un code couleur simple mais brutal, presque «crasse», qui accompagne l'intensité du gameplay. La forme est très instable pour rappeler la dynamique du jeu

Score (haut-centre) : affiché sous forme d'un nombre encadré de deux têtes de démons, ce compteur représente le nombre d'ennemis restant à éliminer. L'iconographie utilisée permet d'associer rapidement cet élément à la notion de carnage.

Multiplicateur et combo (haut-droit) : l'affichage blanc sur rouge agressif attire l'œil. Les chiffres sursaturés et l'effet de distorsion du post-process donnent un style corrompu, amplifiant l'aspect frénétique.

Pierres de compétence (droite) : trois icônes de pierres, visuellement proches de roches taillés, sont affichées à droite. Initialement conçues pour être sur les bras du joueur, elles ont été transposées à l'écran, car elles obstruaient la vision du joueur. A partir du modèle 3D, la forme a été gardée mais adaptée à la fois pour l'UI et pour garder une identité visuelle.

Main du joueur : pas d'armes visibles, seulement des mains gantées, en posture agressive. Cela soutient la DA : combat primal, sans filtres.

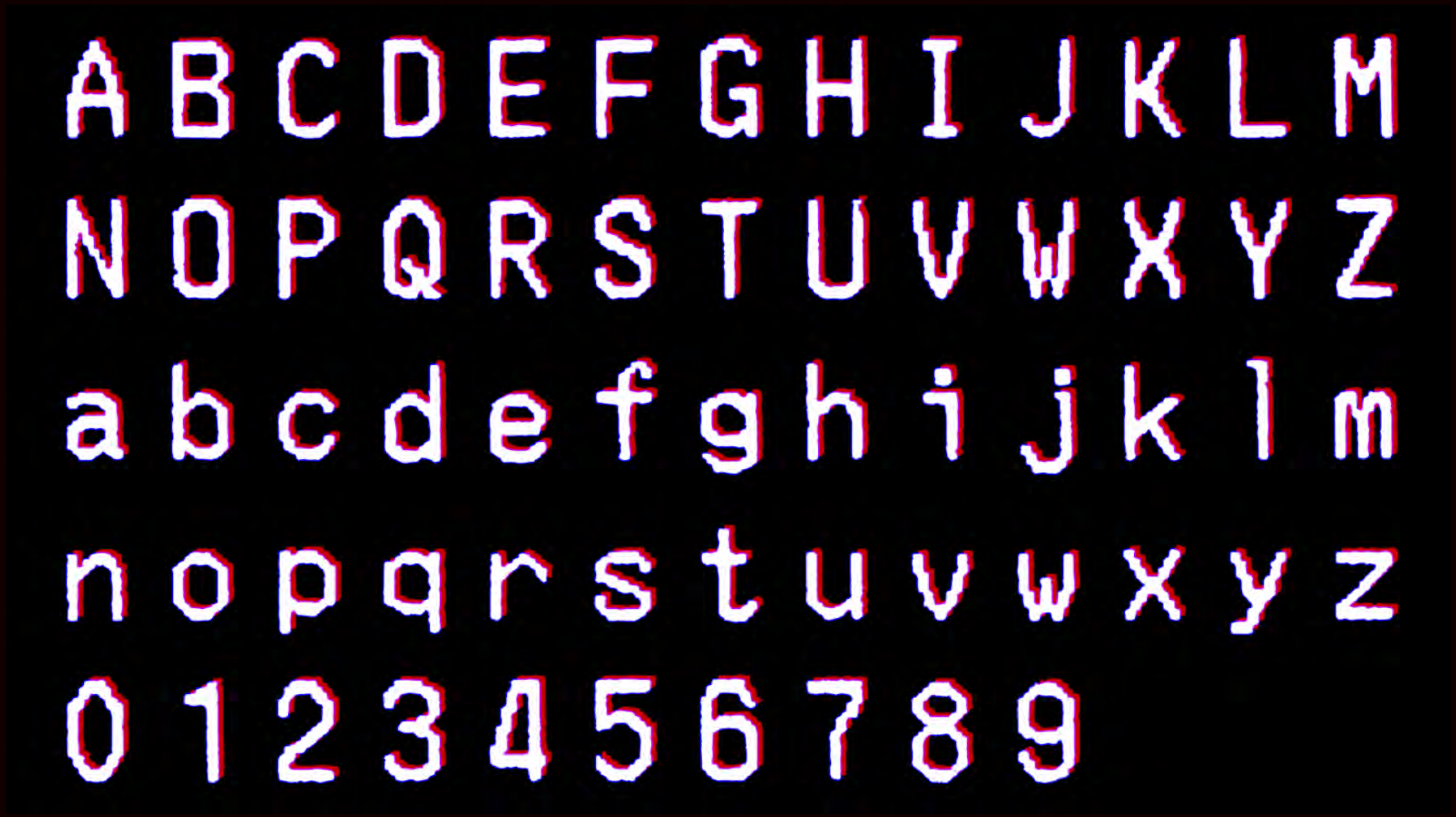
Typographie et DA

La typographie choisie pour Kill Dem'On est une police à empattement instable, qui semble à la limite de l'effondrement ou de la corruption numérique. Elle évoque immédiatement un univers brut, rétro et chaotique, en parfaite adéquation avec l'ambiance gore et infernale du jeu.

Caractéristiques visuelles :
Empattements irréguliers : ils donnent une impression de déséquilibre, comme si chaque lettre était en train de se désagréger. Cela crée un sentiment de tension permanente, un inconfort graphique qui colle parfaitement au gameplay violent.

Instabilité des formes : les lettres semblent parfois "mal imprimées", mal rendues, comme si elles provenaient d'un vieux terminal CRT glitché. On est ici dans une DA qui évoque un brutalisme visuel.

Couleur blanche sur fond rouge sang, avec un léger effet de glow/saturation : ce contraste dur accentue l'agression visuelle et rappel le color split, effet appliqué par le post processing.



typographie du jeu


991
1
x1.00
0 K/Sec
14 E/Sec


455
0 K/Sec
-2308 E/Sec


1216
x3,00
2 K/Sec
-987 E/Sec


995
5
x1,40
0 K/Sec
5 E/Sec


983
9
x1,64
1 K/Sec
6984 E/Sec


0 K/Sec
-4714 E/Se


33
149
x2,80
+112 pts
30 K/Sec
3779 E/Sec

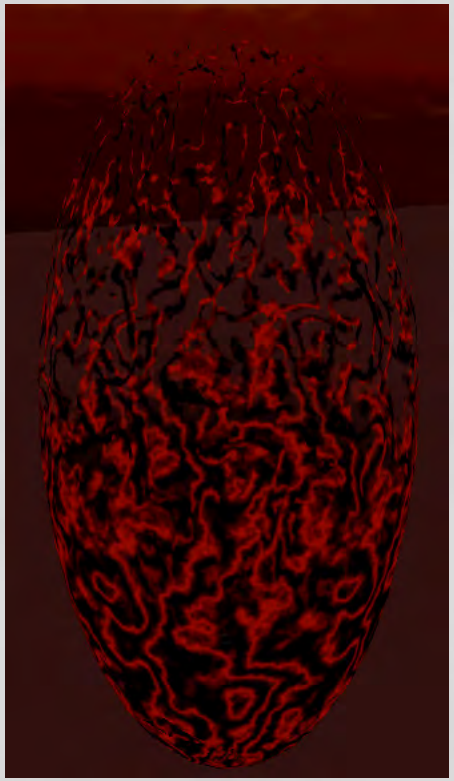
Shader :

Notre jeu repose largement sur l'utilisation de shaders pour produire une grande variété de visuels et d'effets. L'un des principaux atouts d'un shader réside dans sa maniabilité et sa polyvalence : un même shader peut être réutilisé sur de nombreux assets différents. De plus, les shaders offrent une grande flexibilité dans leur conception, pouvant remplir divers rôles : textures, effets de feedback visuel, voire éléments d'asset à part entière.

Révélation/effacement :

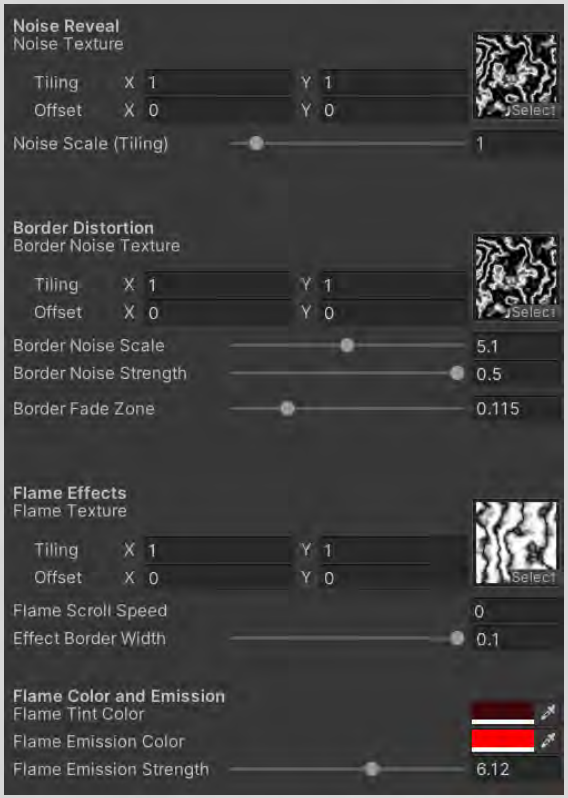
Le shader que nous avons le plus utilisé est un shader de révélation/effacement de textures, capable de faire apparaître ou disparaître dynamiquement des éléments visuels sur un objet. Il nous a permis de créer des effets marquants, comme les boules d'énergie ou encore l'apparition progressive de l'œuf du boss.

Boss Egg



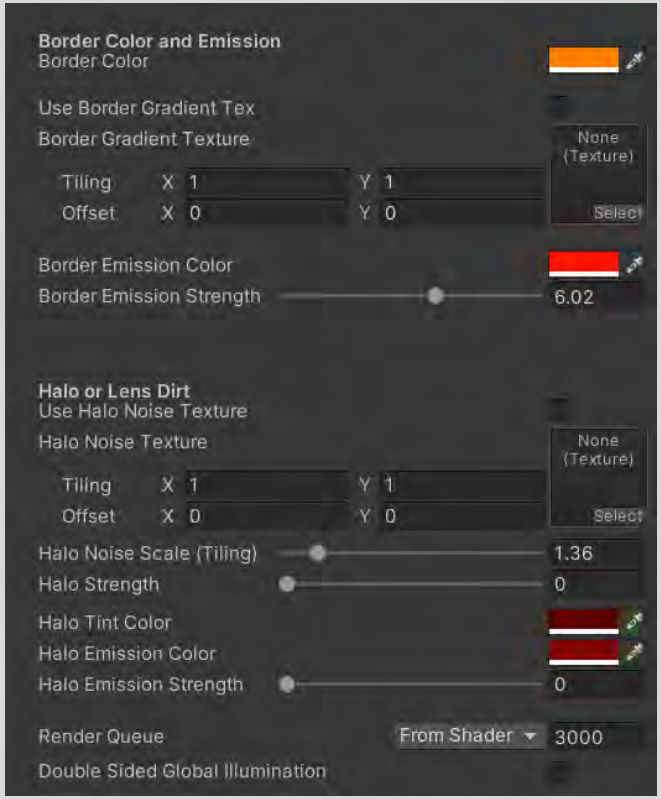
Boss Egg en train d'éclore

Ce shader est l'outil principal que nous utilisons dans notre pipeline visuel. Grâce à l'utilisation de bruits procéduraux (noise) et de textures personnalisées, il nous permet de générer une grande variété de matériaux à appliquer sur des objets très différents. Sa flexibilité nous a permis de le décliner pour répondre à plusieurs besoins spécifiques. Par exemple, la barre d'énergie utilise une version adaptée en 2D de ce même shader, conservant ainsi la cohérence visuelle tout en répondant à des contraintes différentes.



Propriété du Shader

Propriété du Shader

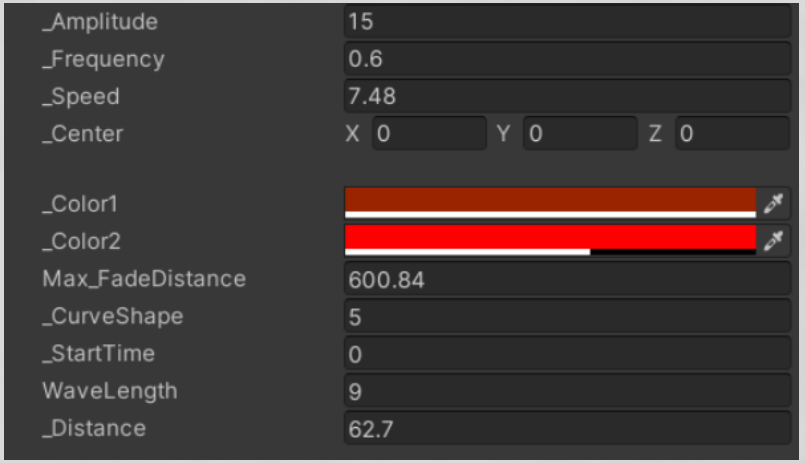


Propriété du Shader

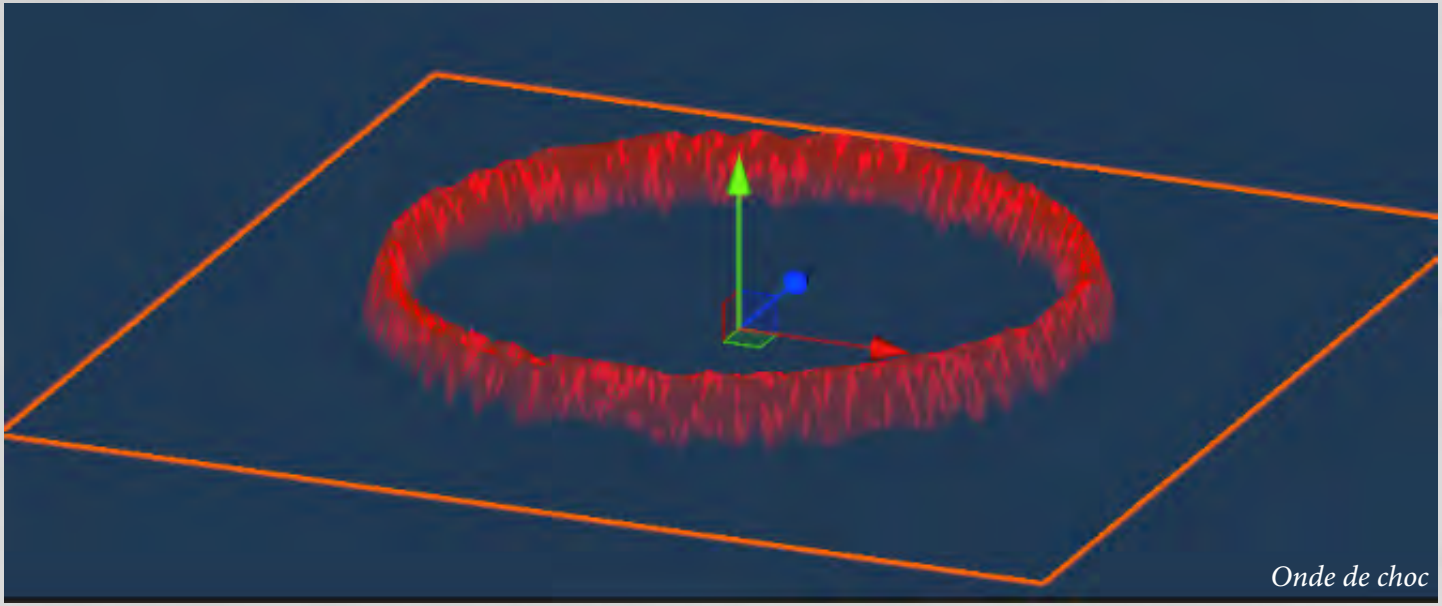
Onde de choc :

Nous avons également développé un shader dédié à notre effet de pilonnage (Stomp). Son objectif est de courber dynamiquement le plan sur lequel il est instancié, afin de simuler une onde de choc visuelle. Ce shader donne une impression de force et d'impact, renforçant la lisibilité et l'intensité de l'action à l'écran.

Ce shader dispose de plusieurs paramètres personnalisables, tels que la taille, la vitesse et la couleur de l'onde. Ces paramètres étant exposés publiquement, ils peuvent être ajustés dynamiquement via le code, ce qui nous permet de moduler l'effet selon le contexte ou le gameplay.



Onde de choc propriété

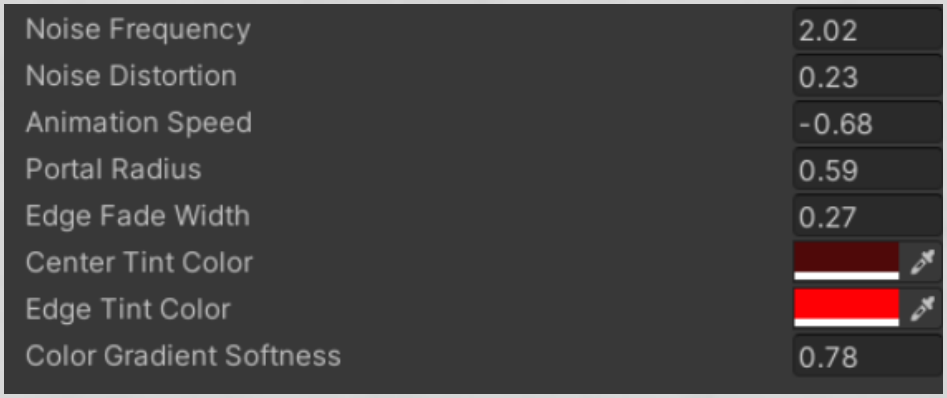


Onde de choc

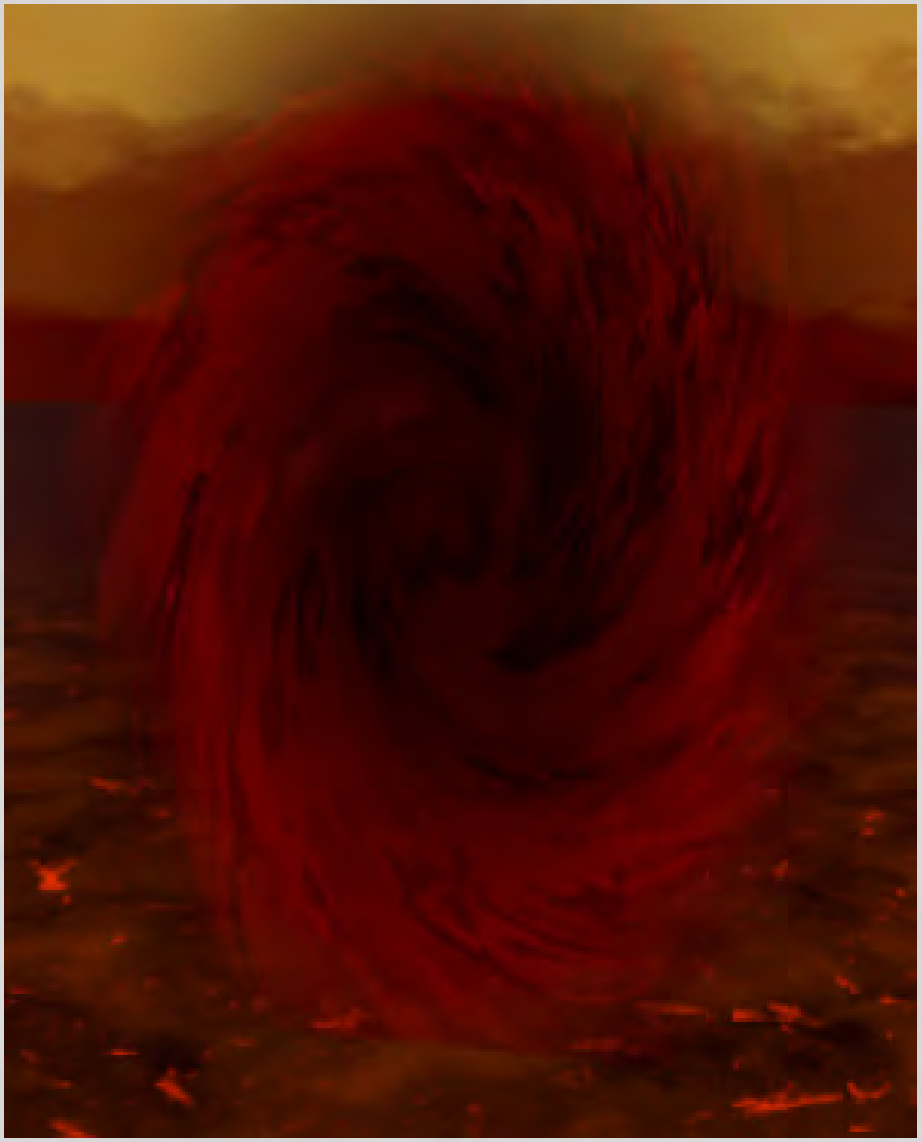
Portail :

Enfin, le dernier shader “unique” que nous utilisons est celui du portail. Celui-ci sert à faire apparaître les démons ou à transporter le joueur, selon sa couleur, qui détermine sa fonction. Ce shader combine effets de matière et animation pour renforcer son caractère magique et mystérieux.

Ce shader de portail dispose de plusieurs paramètres configurables, tels que le rayon, la vitesse de rotation, la fréquence du bruit (noise), ainsi que d’autres paramètres secondaires permettant d’affiner son apparence. Bien que ces paramètres soient accessibles via le code, ils ne sont pas modifiés dynamiquement en jeu : nous instancions directement le portail avec son matériau préconfiguré, ce qui garantit une apparence cohérente et maîtrisée à chaque apparition.



Portail propriété



Portail

Direction Artistique Sonore

Intentions

Le sound design de notre jeu repose sur une direction claire : soutenir la direction artistique, renforcer la montée en puissance des paliers, et sublimer le Game Feel. Chaque son doit non seulement être fonctionnel, mais aussi émotionnellement évocateur, pour faire ressentir au joueur toute la brutalité et la fluidité de son ascension.

Un son sale et distordu en cohérence avec la DA

Notre direction artistique repose sur un univers sombre, brutal et surnaturel. Le sound design s'aligne sur cette esthétique en proposant des textures sonores rugueuses, distordues, presque sales, à l'image d'un monde où le pouvoir est instable, sauvage et dangereux. Saturations, filtres lo-fi, distorsions et réverbérations angoissantes sont autant d'outils utilisés pour transmettre cette ambiance. Le but n'est pas la propreté sonore, mais la rugosité expressive.

Des sons directement liés aux paliers

Les paliers sont au cœur de la progression du joueur. Chaque changement de palier est accompagné d'un changement sonore perceptible et identifiable, pour signifier clairement la montée en puissance. À chaque nouveau niveau, les sons gagnent en richesse, profondeur, intensité et spatialisation. Cette hiérarchie sonore permet au joueur d'identifier instantanément son état de puissance, même sans interface visuelle, et participe à l'immersion sensorielle dans le gameplay.

Un Game Feel puissant et organique

Chaque action du joueur doit produire un feedback sonore immédiat, riche et satisfaisant. Le sound design est pensé pour renforcer la sensation de contrôle total, avec des sons synchronisés, dynamiques et percutants. Le moindre son participe au plaisir de jouer. Ce Game Feel sonore est essentiel pour immerger le joueur dans un état de flow, et lui faire ressentir qu'il devient progressivement l'ultime menace du jeu.

Point Technique :

D'un point de vue technique, le principal enjeu est le nombre de sons qui se joue en même temps. En effet, on tue énormément d'ennemie en même temps et cela peut poser des problèmes de superpositions de son.
De plus, il y a un enjeu de localisation pour les ennemis, on doit sentir d'où ils viennent et où on les tue. Pour cela, on va localiser chaque son sur les ennemis que ce soit les kills ou les déplacements ou les Hits.

Références :

Doom Eternal :

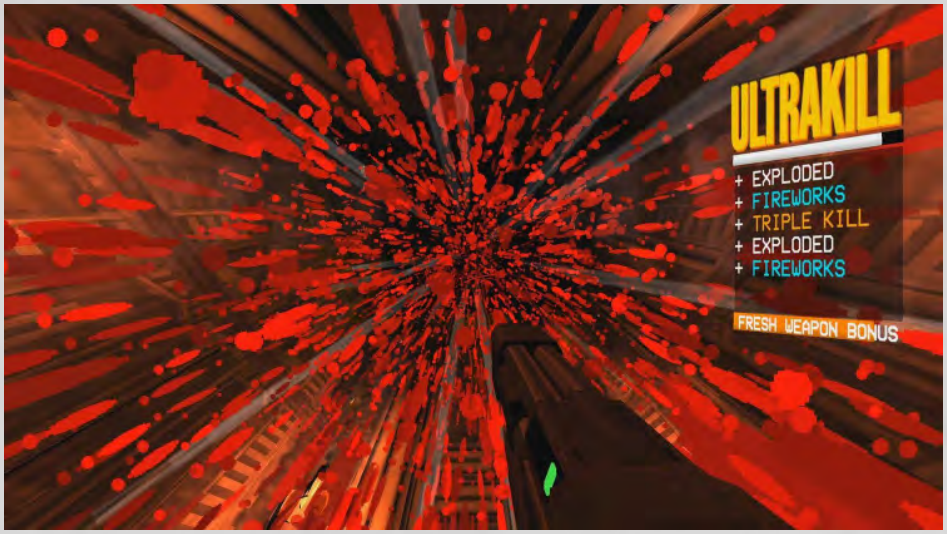
- Effets sonores d'ennemis avec des grognements profonds et saturés.
- Des sons explosifs et réverbérants pour les attaques magiques.
- Des ambiances infernales, comme des bruits de lave ou des échos démoniaques.

Elden Ring :

- Des sons de sorts magiques réverbérants, souvent graves et puissants (parfaits pour ton gantlet magique).
- Des ambiances menaçantes, avec des bruits de vent lugubre, de flammes mystiques et de gémissements lointains.
- Des feedbacks audio bien dosés lors de l'utilisation des armes magiques (explosions énergétiques, vibrations puissantes).

Diablo :

- Sons puissants et percutants pour les attaques magiques
- Des bruits d'invocation démoniaque, avec des échos graves et résonants.



Brotato :

- Tirs courts et saccadés (sons "pop" ou "zap") pour les tirs rapides de base.
- Un effet sonore de montée progressive à mesure que la cadence ou la puissance des tirs augmente.
- Ajoute un feedback visuel/audio lorsque les tirs atteignent un seuil critique (ex. bruit de surcharge ou distorsion).

Ultrakill :

- Des sons électriques saturés qui augmentent la tension avant chaque tir.
- Un feedback sonore rapide, presque arcade.
- Des impacts violents avec un effet de compression sonore, donnant l'impression d'une énorme décharge.

Un mixage Globalisé :

Dans notre projet, le Global Parameter de FMOD joue un rôle central dans notre intégration sonore. Ce paramètre est directement lié à l'un des éléments clés de notre gameplay : les paliers de compétence. Grâce à lui, on a pu mettre en place une solution à la fois simple, efficace et cohérente, parfaitement adaptée à notre production sans sound designer attitré.

Un gain de temps précieux :

Normalement, chaque palier de compétence devrait avoir son propre son, mixé individuellement. Mais cette méthode demande beaucoup de temps, de ressources et une gestion fine du mix. Ici, on a opté pour une approche bien plus maligne : on utilise un seul son par compétence, enrichi de modificateurs FMOD (modifiers) qui varient selon le palier actuel. Résultat : des variations sonores dynamiques, sans multiplier les fichiers audio ni alourdir le pipeline.

Une intégration fluide dans Unity :

Un autre avantage clé de cette méthode, c'est sa simplicité d'intégration dans Unity. Une fois le Global Parameter configuré dans FMOD, tout se gère côté FMOD. Dans Unity, il suffit juste de jouer l'événement sonore au bon moment : plus besoin de lier manuellement chaque son à son palier. C'est propre, rapide, et centralisé.

Une expérience sonore claire et cohérente :

Ce système a un autre avantage majeur : il permet de garder une unité sonore entre les différentes compétences. Chaque compétence garde sa propre identité sonore, tout en évoluant subtilement selon le palier. Cela permet au joueur de reconnaître immédiatement l'action effectuée, tout en percevant une montée en puissance ou une transformation à travers le son. En utilisant les mêmes modifiers pour différentes compétences, on instaure aussi une logique sonore cohérente : les paliers «résonnent» de la même manière, peu importe la compétence. Cela renforce la lisibilité et l'immersion sans complexifier la production.

Changement d'effet en fonction du palier

Musique :

Intention musicale :

La musique dans notre jeu n'est pas un simple fond sonore : elle est un élément moteur du gameplay, pensé comme un outil de feedback, d'immersion et de progression. Elle guide le joueur tout au long de son ascension, en soulignant son état de puissance, tout en traduisant l'émotion propre à chaque palier. À chaque instant, elle doit renforcer le ressenti de montée en puissance, sans jamais nuire à la lisibilité ou à la fluidité du jeu.

Feedback essentiel :

Le changement de musique agit comme un retour immédiat et naturel pour informer le joueur du palier dans lequel il se trouve. C'est un feedback audio constant, qui évite d'avoir à afficher des informations visuelles supplémentaires. Ce changement musical marque les moments clés du gameplay :

Gain ou perte de palier :

- État de danger (énergie basse)
- Stabilité ou instabilité énergétique
- Ce système participe activement à la lisibilité du jeu et renforce la sensation de contrôle et de progression du joueur.

Ambiance évolutive au service de l'immersion

Chaque palier est associé à une ambiance musicale unique, qui transforme l'univers sonore du jeu au fur et à mesure que le joueur évolue. Cela permet de donner à chaque étape une identité sonore forte. Le monde évolue avec le joueur, s'adapte à son état de puissance, et souligne les transitions par des variations de rythme, d'intensité, de richesse instrumentale ou de texture sonore. La musique agit ainsi comme un narrateur invisible, qui accompagne le joueur dans sa montée en puissance, sans interrompre le gameplay.

Les différentes musiques en fonction du palier

Palier 1 : Tension et fragilité

Une musique oppressante, avec des nappes graves et étouffées. Le rythme est lent, la dynamique faible. L'objectif est de faire ressentir la vulnérabilité du joueur, au bord de la mort. Cela crée un état de tension constante, en phase avec une posture défensive.

Palier 2 : Accélération et affirmation

Le rythme s'accélère légèrement, la texture sonore s'enrichit. On perçoit un retour positif pour le joueur, plus affirmé. La musique reste discrète mais offre une sensation de maîtrise. C'est un palier de transition où le joueur commence à reprendre le dessus.

Palier 3 : Flow et intensité

La musique devient plus musicale et rapide, avec des motifs rythmiques plus marqués. La sensation de flow est maximale : tout s'enchaîne naturellement, et la bande-son renforce ce sentiment par sa cadence soutenue et son harmonie dynamique. Le joueur est dans sa zone.

Palier 4 : Puissance absolue et climax sonore

Véritable breakdown musical, ce palier introduit une musique puissante, saturée, expansive. On entend de nouveaux instruments, des percussions massives, des nappes intenses. C'est l'apogée sonore du gameplay, un moment de domination totale. Le joueur est devenu une force incontrôlable.

Pitch descendant en cas de danger de perte de palier :

Un mécanisme subtil mais crucial renforce encore l'expérience : la baisse temporaire du pitch de la musique lorsque le joueur tombe sous le seuil énergétique de son palier actuel. Ce «sustain altéré» agit comme un signal de danger immédiat :

- Si le joueur remonte rapidement en énergie, la musique reprend son pitch normal, soulignant une reprise de contrôle.
- S'il échoue à retrouver son énergie, alors le jeu bascule dans la musique du palier inférieur, marquant une vraie perte de puissance.

Ce système de musique adaptative permet de rendre le gameplay plus vivant, plus lisible et plus immersif, sans ajouter de surcharge visuelle. Le joueur ressent le danger, l'anticipe, et y réagit uniquement par l'écoute.


0 K/SEC
-595 E/SEC

Gestion de Projet

Planification initiale :

Lors des premières étapes, nous avons mis en place un calendrier global des livrables et structuré les grandes étapes du projet (Game Design Document, Level Design Document, prototype, etc.) dans un tableau collaboratif. Ce calendrier, estimant durée et dates de chaque tâche, nous a offert une vision d'ensemble et permis de répartir équitablement la charge de travail sur la période.

Game Overview Document

Numéro	Tâche	Pôle	Durée estimée	Date éstimée	Déscription	Tâches prérequises	État
Game Design Document (Vers Toys)							
1	Préparation Indesign	Game Design	0,5J	1/12/2025	Construit la doc, met les marge, les typo selectionné, appliqué les titre et sous titre, etc	/	To Do
2	GDD : Introduction	Game Design	0,5J	1/12/2025	Page Équipe, fiche d'identité	/	To Do
3	GDD : Intention et noyau système Partie 1	Game Design	0,5J	1/12/2025	Page Intention GD, Description noyau système, tendance sys, tension ludique, boucle de gameplay	/	To Do
4	GDD : Noyau système Partie 2	Game Design	2J	1/15/2025	Page 3Cs, Boucle de prediction, schéma de ventrice, boucle de motivation	/	To Do
5	GDD : Situation Gameplay et OCR	Game Design	1J	1/17/2025		/	To Do
6	GDD : Référence GD et Première itération	Game Design	1J	1/20/2025		/	To Do
7	GDD : Deuxième et troisième itération	Game Design	1J	1/20/2025		/	To Do
8	GDD : Game Feel	Game Design	1J	1/20/2025		/	To Do
						Durée estimée	Deadline
Game Design Document (Vers Toys)						7J	24/01/2025
Document LD							
9	Introduction et description	Game Design Level Design	2J	1/22/2025	Page macro LD, Ingrédient LD, Layout du blocking	/	To Do
10	Projection LD	Game Design Level Design	1J	1/25/2025	Description du Système Difficulté dynamique	/	To Do
						Durée estimée	Deadline
Document LD						3J	25/01/2025

Schedule GOD | Schedule Prototype | Schedule V1_legacy

Schedule V1

Définition progressive des tâches

En partant d'un organigramme de tâches (par pôle : GD, LD, UI, etc.), nous avons listé les éléments «Must Have» et les avons décomposés progressivement en sous-tâches précises. Chaque tâche a été détaillée (objectif, contenu, dépendances), puis intégrée dans notre tableau de suivi pour faciliter la priorisation.

Toutes les deux semaines, un objectif court-terme est fixé pour maintenir un bon rythme de production et créer des itérations testables.

Legacy

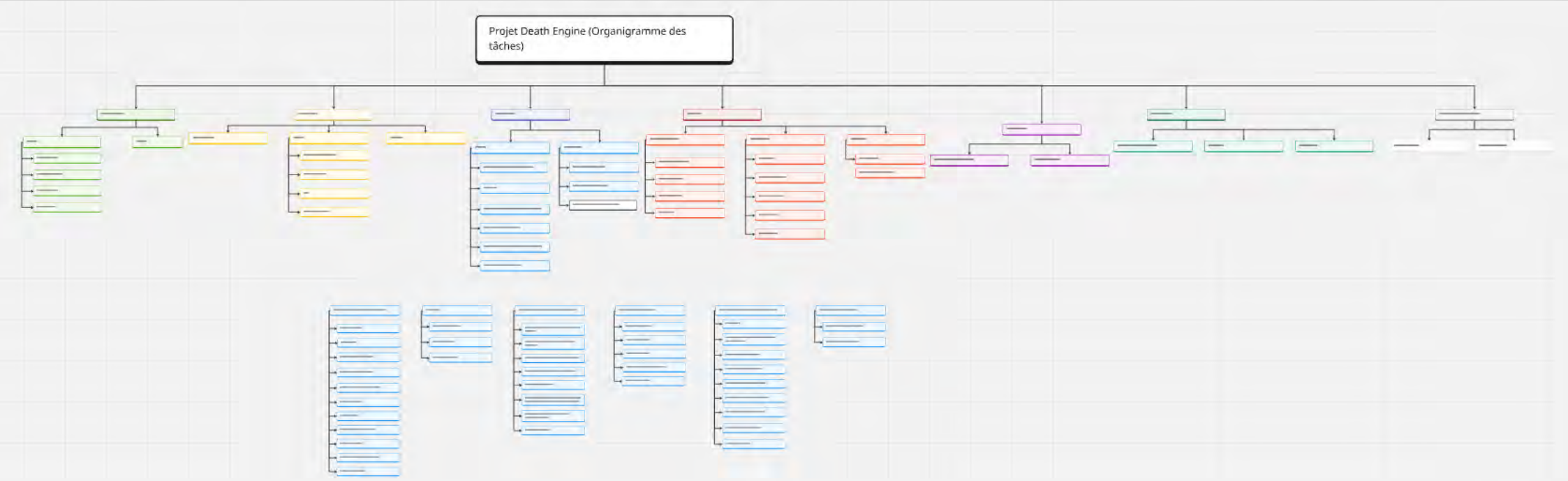
Numéro	Tâche	Pôle	Durée	Date éstimée	Déscription	Tâches prérequises	État
				10/14/2024			
8	Analyse du prototype (Variation)	Game Design		10/15/2024	Analyser le prototype actual, identifier tout ces mécaniques	6	Done
9	Peaufiner la tendance système	Game Design		10/22/2024	Revoir la Tension ludique, améliorer augmenter la tendance sys et la tendance du joueur (en Proposant ou modifier des mécaniques)	8	Done
10	Boucle de gameplay In and Out	Game Design		10/22/2024	Proposer des boucle de gameplay In and Out	8	Done
11	Création d'un outil de prototypage	Prog		10/22/2024	Création d'un outil de prototypage afin de permet au GD puis tester leur idée		Done
12	Conception des situation de Gameplay	Game Design		10/27/2024		10,9	Done
13	Documenter et schématiser le GDD (situation gameplay)	Game Design		10/27/2024	Documenter et schematiser tout les proposition des boucles de gameplay, les mécanique, le core système, la tendance système.		Done
~~14~~	~~Étudier et améliorer la mécanique de Grab & Throw~~	~~Game Design~~			~~Explorer cette mécanique, rendre cette mécanique exploitable à différent niveau~~		
15	Prototypage (Situation de gameplay)	Prog Game Design		11/3/2024	Lancement de la production du prototype	12,13	Done
16	Finition du prototype (situation de gameplay)	Prog		11/3/2024	Prototype jouable, testable		Done
				11/10/2024		27	
17	Analyse du prototype (Situation gameplay)	Game Design	1J	11/6/2024	Analyser le prototype actual, identifier tout ces mécaniques etc.	16	Done
18	Proposition du Fiche d'identité et Intention Global	All	1J	11/6/2024	Chaque personne propose un fiche d'identité et l'intention global		Done
19	Définir le première fiche d'identité et	All	1J	11/7/2024	Définir la fiche d'identité et		Done

Schedule GOD | Schedule Prototype | Schedule V1_legacy

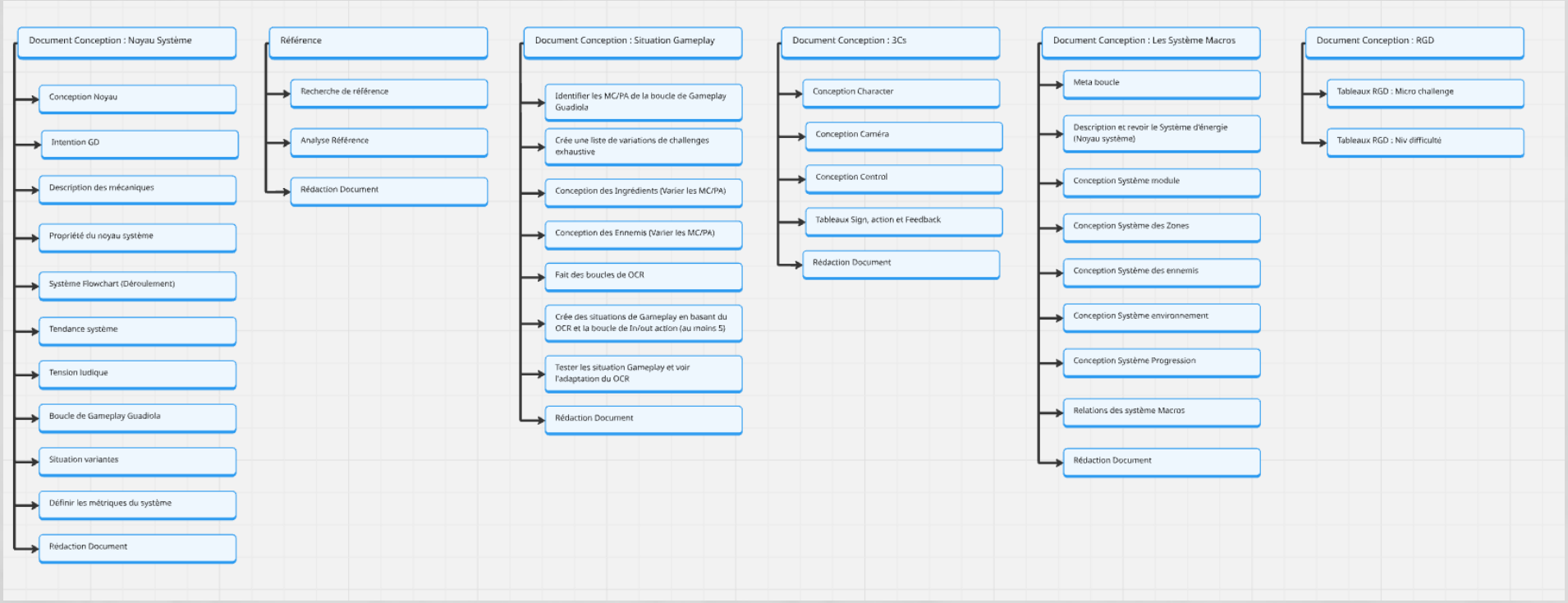
Schedule V1

Itérations jouables et retours utilisateurs

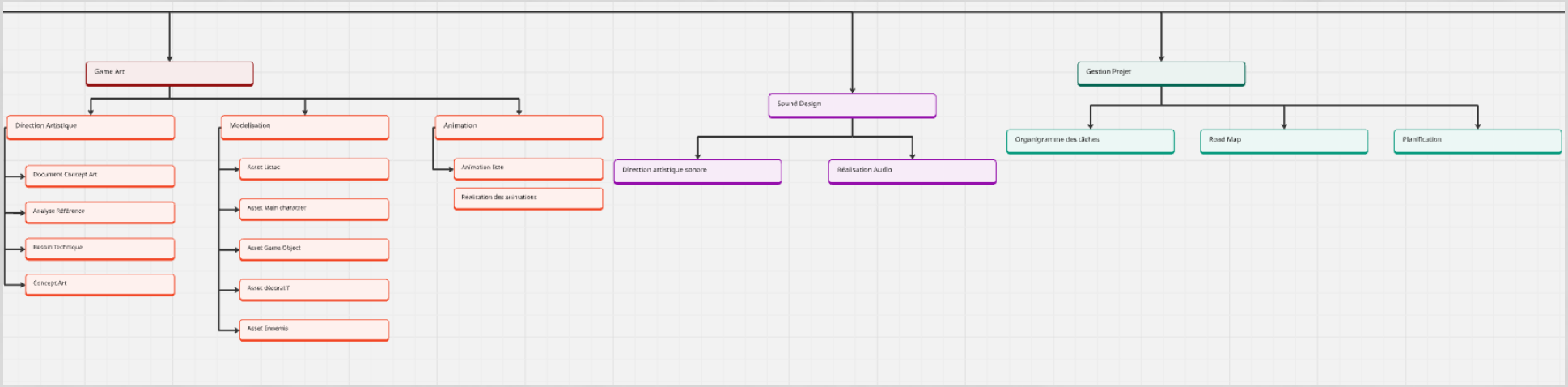
Dès que le jeu est devenu partiellement jouable, nous avons commencé à livrer des versions internes régulièrement. Ces builds ont été testés par des joueurs proches de l'équipe, ce qui nous a permis d'ajuster la priorité des tâches en fonction des retours (game feel, difficulté, lisibilité...).

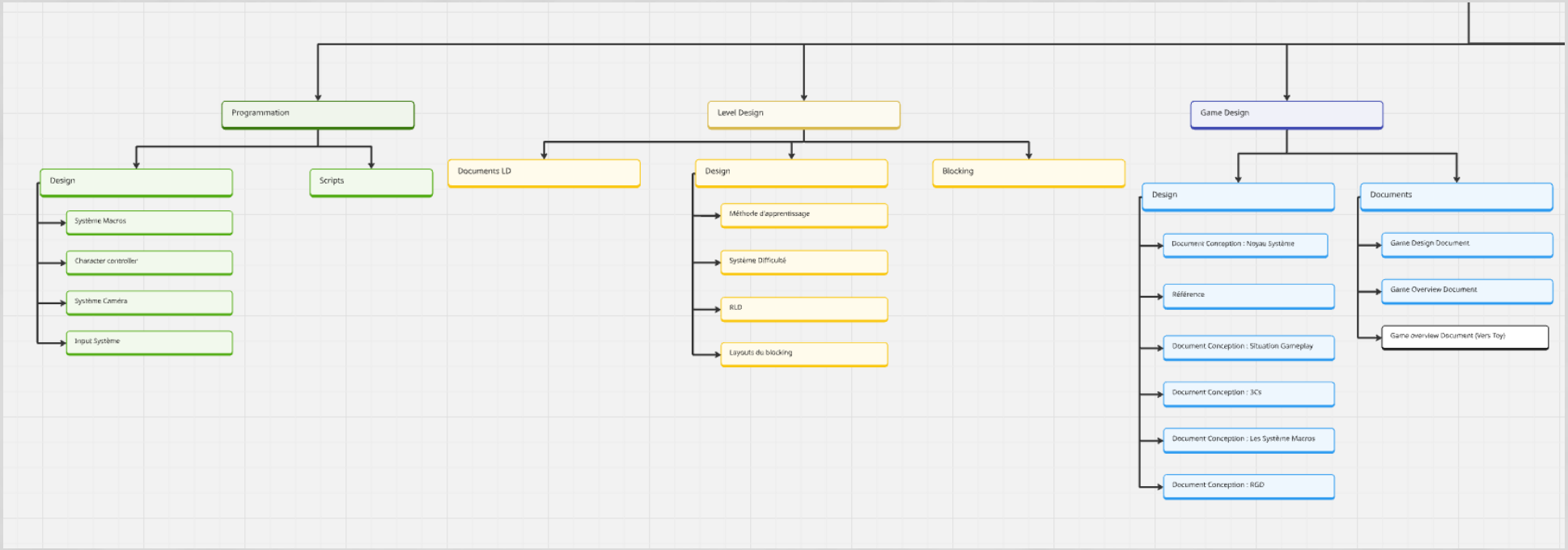


Organigramme Tache V1 vue d'ensemble

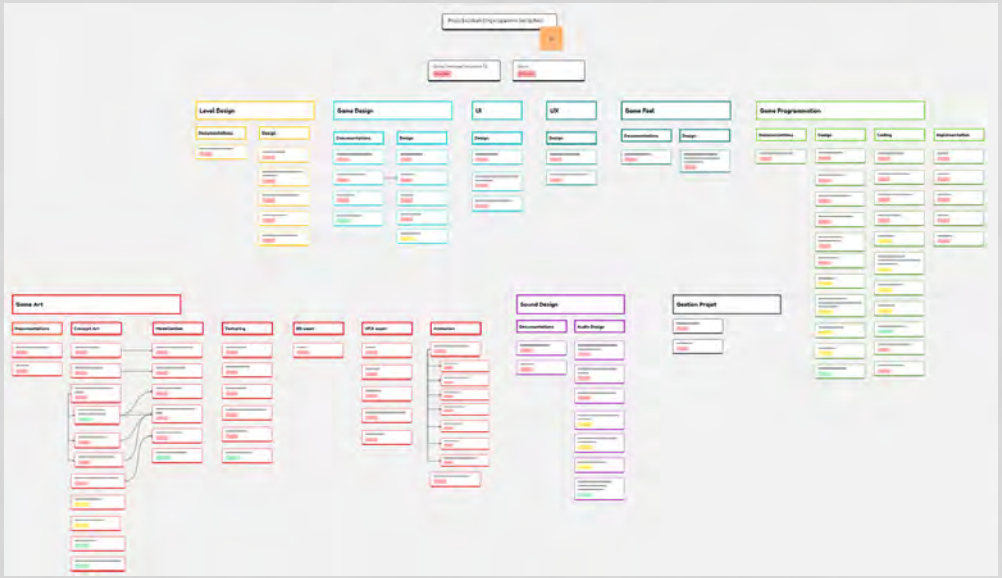


Organigramme Tache V1 vue par partie

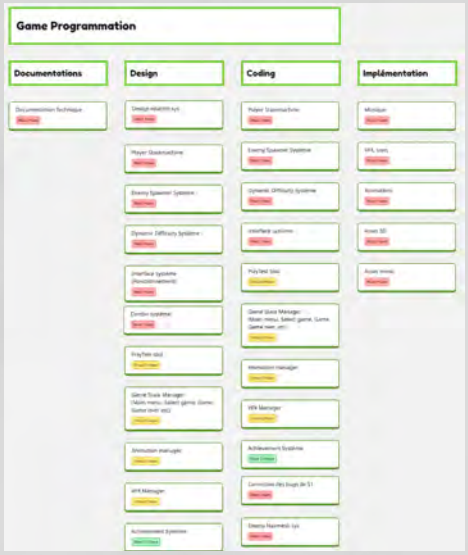




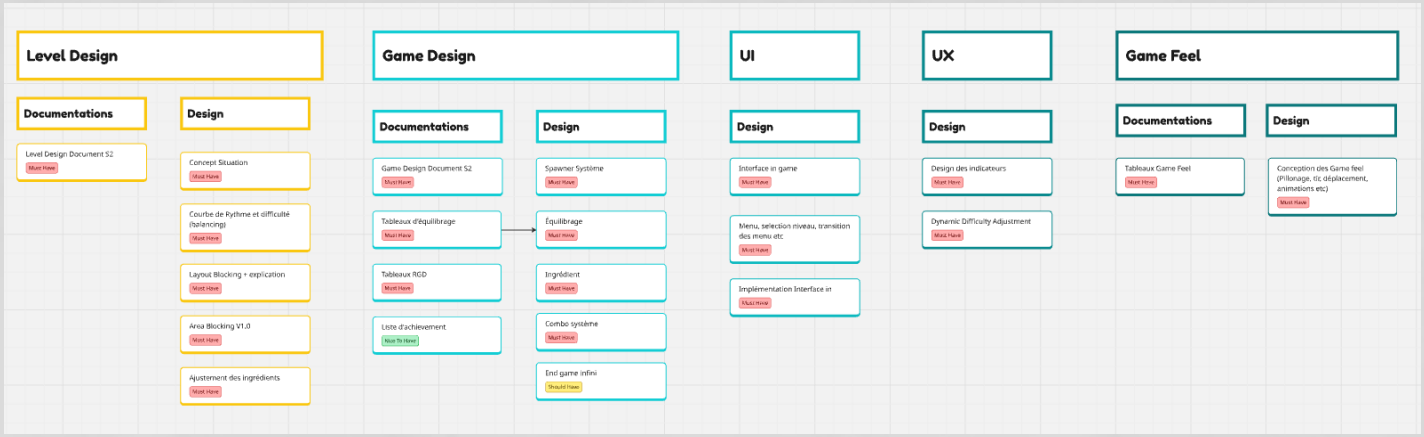
Organigramme Tache V1
vue par partie

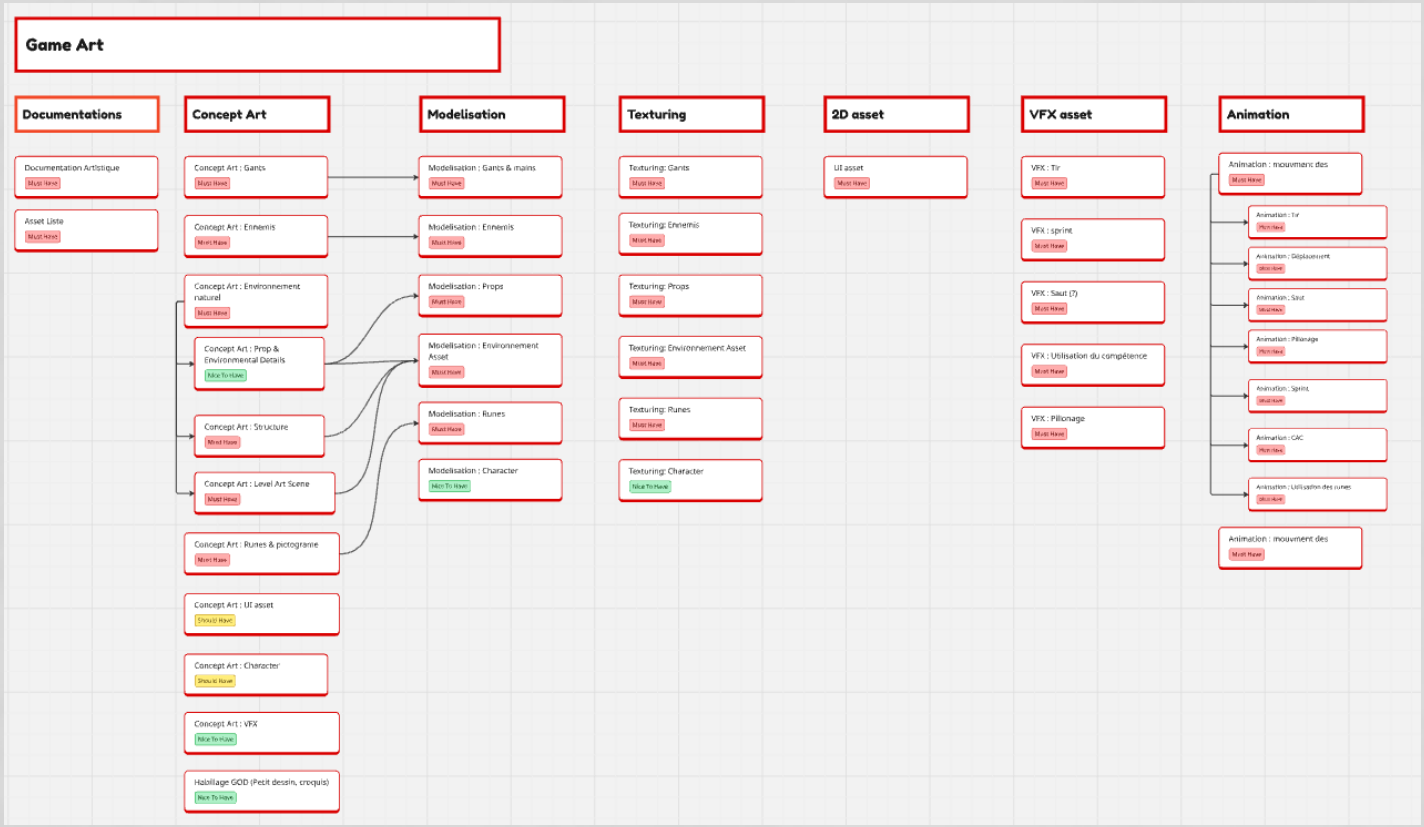


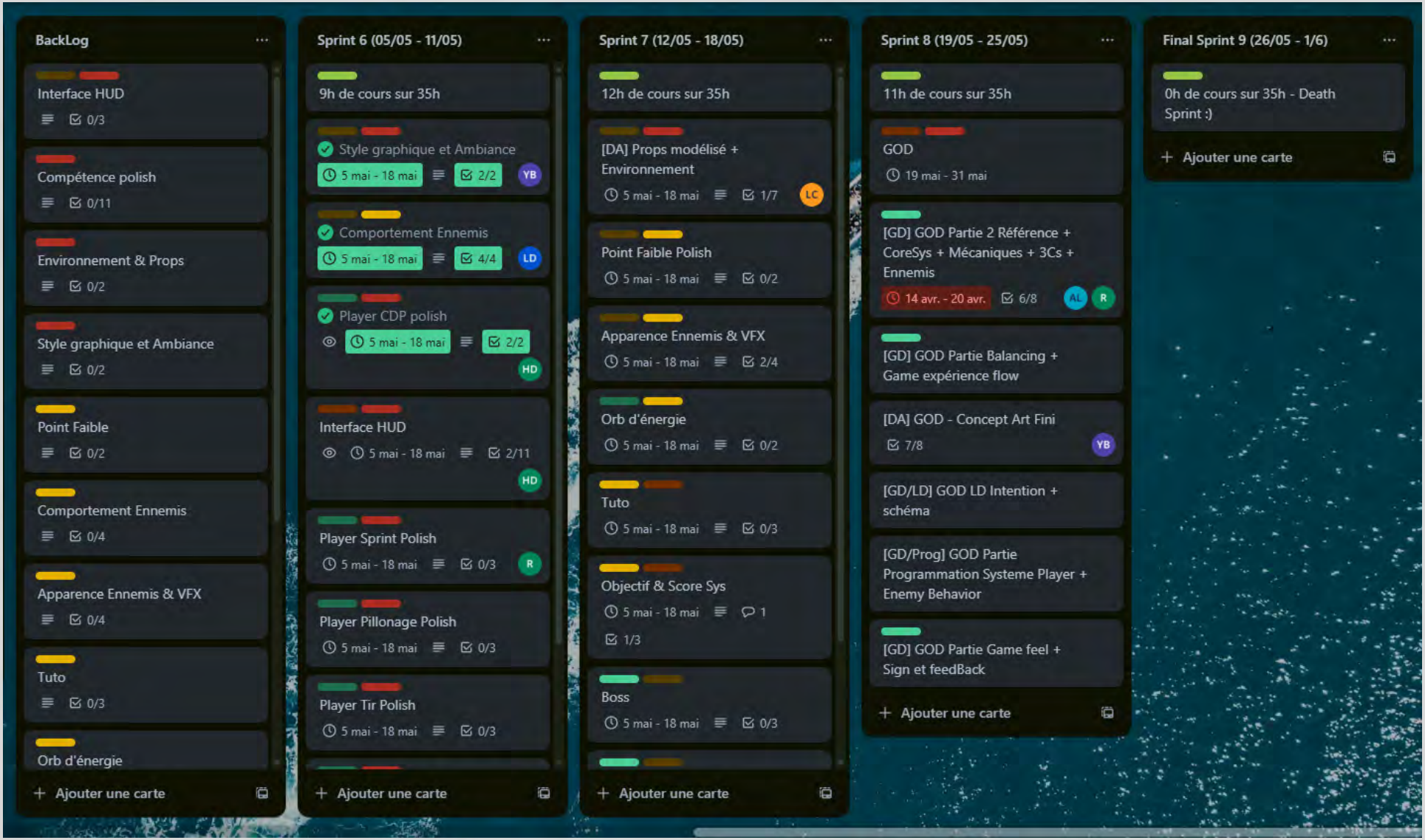
Organigramme Tache V2
vue d'emsemble



Organigramme Tache V2 vue par partie



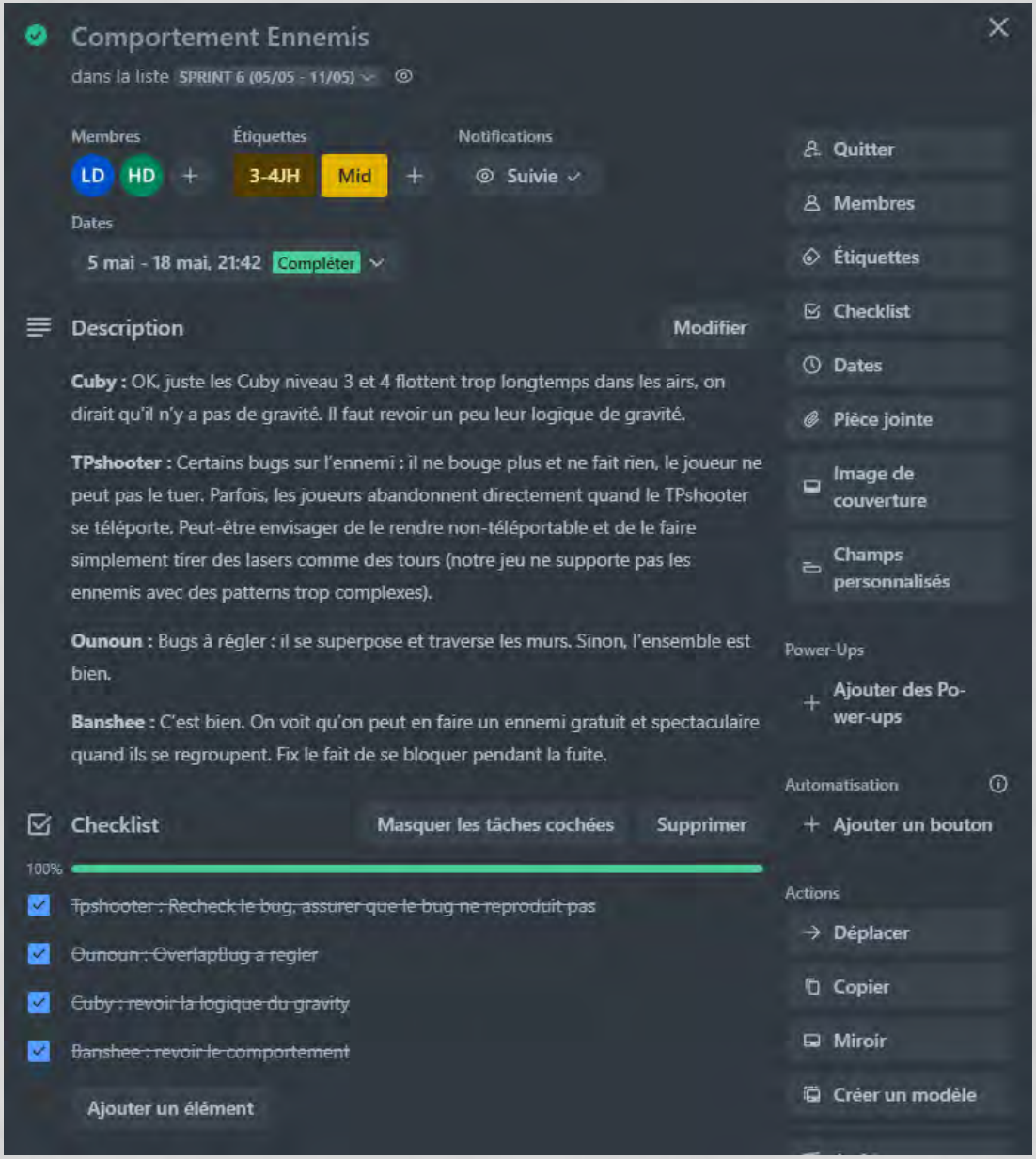




Sprint Trello

Phase de production complète

Une fois en phase de production active, nous avons structuré le travail autour de sprints de deux mois, divisés par semaine. Chaque sprint contient un volume de travail réaliste, avec des tâches ordonnées logiquement. Les tâches sont clairement assignées, accompagnées d'objectifs, de formats de rendu et de délais. Cela a permis de limiter l'ambiguïté, de favoriser l'autonomie, et de fluidifier les collaborations inter-pôles.



Tache trello

Communication au sein de l’équipe :

Discord comme hub central

Notre serveur Discord est structuré en plusieurs canaux thématiques :
- Échanges quotidiens (coordination, partage de captures, points rapides)
- Ressources et références (liens utiles, inspirations, articles de game design)
- Suivi des sprints et annonces

Un bot GitHub y est intégré afin de relayer les mises à jour du projet de manière automatique, assurant une bonne visibilité du travail technique.

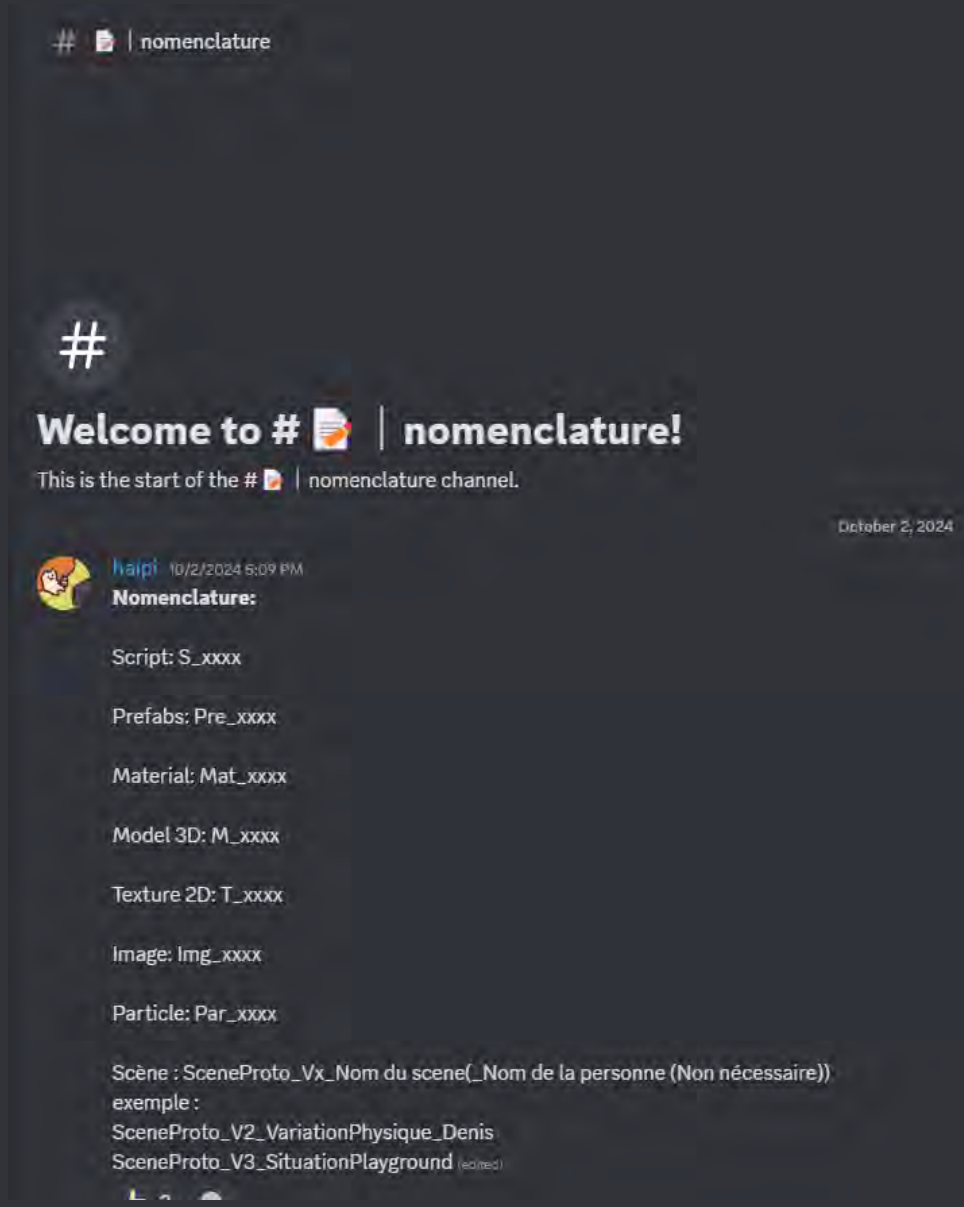
Suivi détaillé via Trello

Nous utilisons Trello comme outil de gestion de tâches.
Chaque carte comprend :
- Une description claire de l’objectif
- Des feedbacks de test intégrés
- Une checklist de polish quand nécessaire
- Une étiquette de priorité (Must Have, Should Have, etc.)
- Des tags de durée estimée
- Des deadlines par sprint

Cela nous permet de suivre à la fois la charge réelle et les besoins de révision (feedback loop) de manière fluide.
Exemple : la tâche « Player CDP polish » a été revue avec une checklist intégrée à partir des retours joueurs sur l’imprécision du ciblage lors du dash.

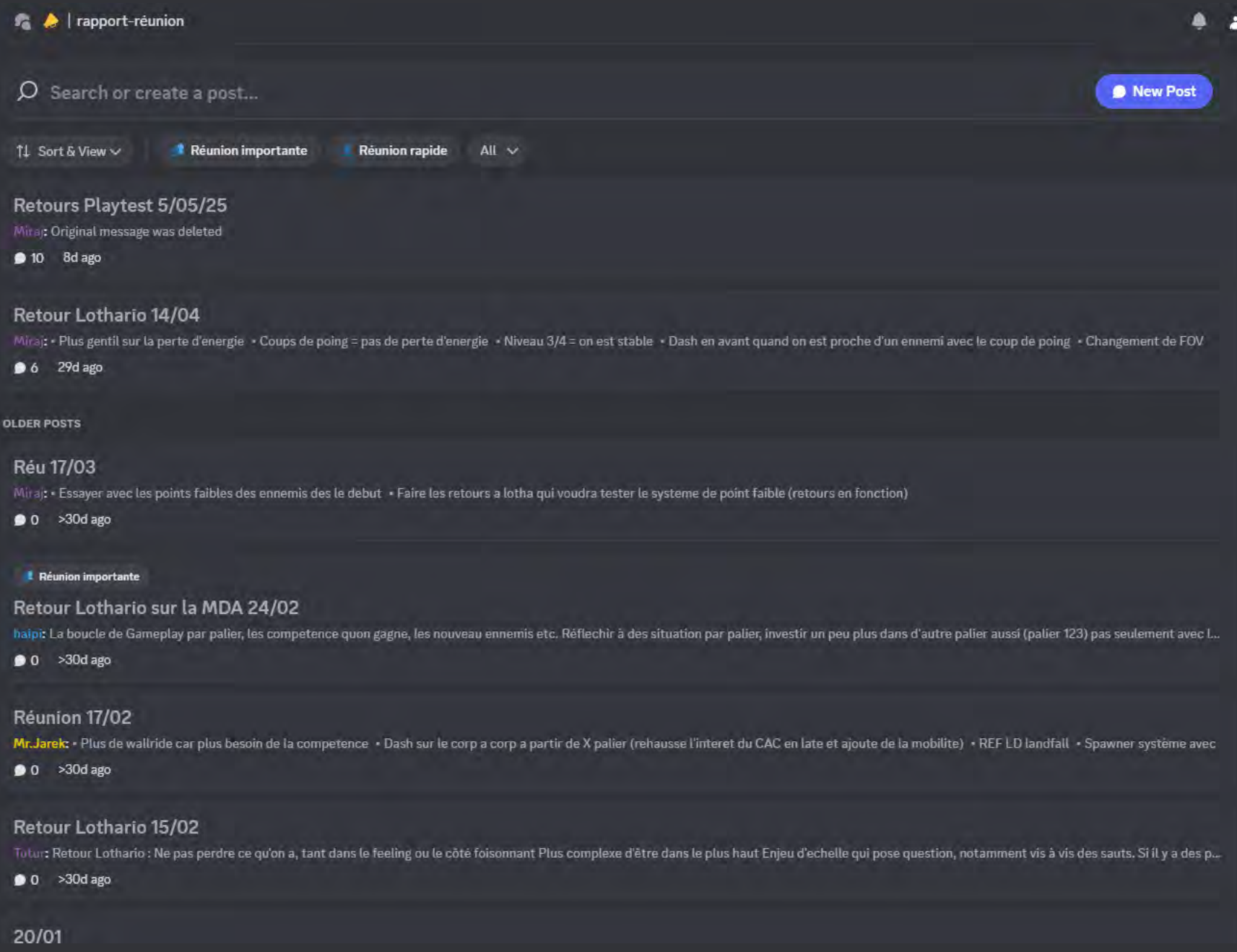
Conclusion

Grâce à notre organigramme des pôles et des tâches, chaque membre peut identifier les responsabilités par domaine (Game Design, Level Design, Game Feel, Game Art, UI/UX, Programmation). Cela facilite la répartition du travail, la lecture des dépendances et permet à chacun de situer l’avancement de son propre pôle dans le cadre global du projet.

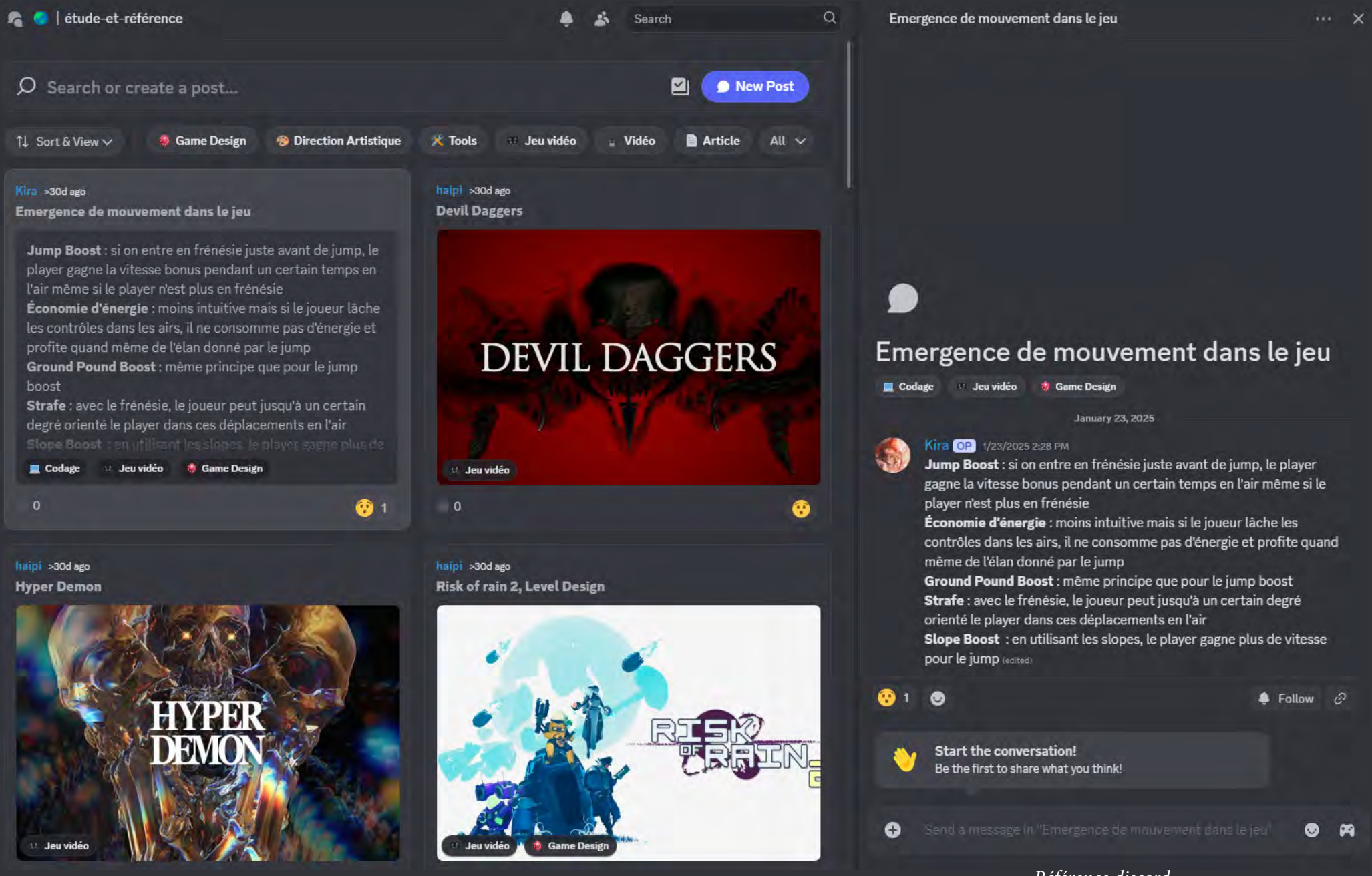


Nomenclature Channel Discord

GitHup Bot

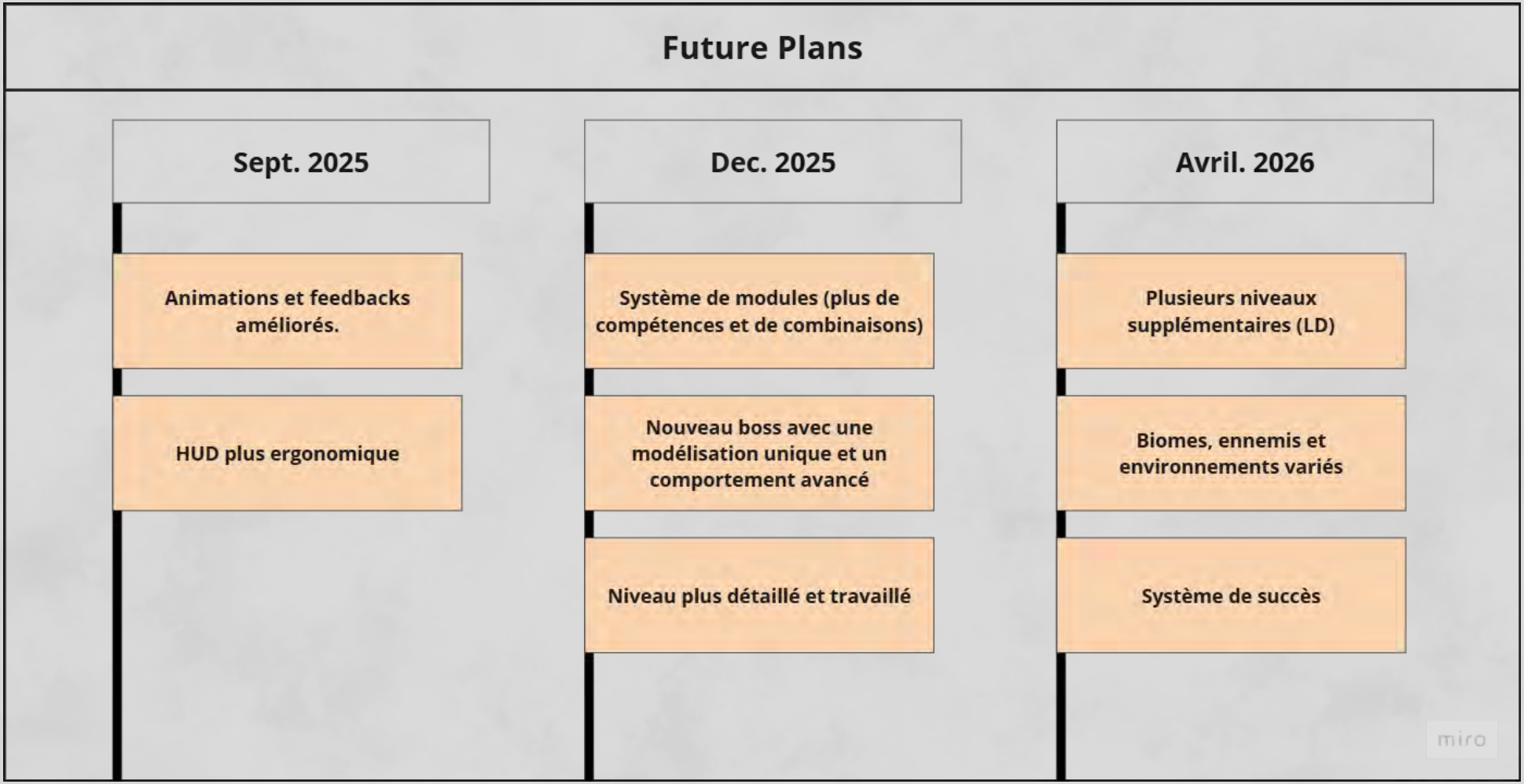


Rapport de réunion discord



Référence discord

Projections


Future Plans
Sept. 2025
Animations et feedbacks améliorés.
HUD plus ergonomique
Dec. 2025
Système de modules (plus de compétences et de combinaisons)
Nouveau boss avec une modélisation unique et un comportement avancé
Niveau plus détaillé et travaillé
Avril. 2026
Plusieurs niveaux supplémentaires (LD)
Biomes, ennemis et environnements variés
Système de succès
miro

Projection Septembre :

La phase de projection à court terme regroupe les éléments déjà présents dans le jeu, mais qui nécessitent des améliorations ciblées pour assurer la réussite de notre démo jouable. Ces ajustements portent à la fois sur le ressenti en jeu (animations et feedbacks) et sur la compréhension de l'interface (HUD). Ils sont cruciaux pour garantir une expérience cohérente, lisible et percutante.

1. Animations et feedbacks à renforcer

Nous avons identifié que certaines animations manquaient de puissance visuelle. Notamment lors des impacts ou des actions clés, le joueur ne ressent pas encore assez la violence ou l'intensité attendue dans un fast FPS.
Cela s'explique en partie par le fait que notre équipe ne comptait pas d'animateur expérimenté, ce qui a limité la finesse et le dynamisme de certaines animations.
Nous souhaitons donc reprendre une partie des animations critiques, avec l'objectif de renforcer leur lisibilité, leur poids, et leur réactivité.

En parallèle, les feedbacks visuels et sonores seront retravaillés. C'est une composante centrale de notre gameplay : les feedbacks doivent être clairs, immédiats et satisfaisants. Nous pensons qu'il est possible de pousser encore plus loin leur impact, notamment lors des kills, des changements de palier, ou de l'utilisation de compétences.

2. Un HUD plus ergonomique et intuitif

Le HUD actuel fonctionne sur le plan technique, mais nous avons constaté plusieurs incompréhensions de la part des joueurs lors des sessions de test. Les éléments clés comme la jauge d'énergie, l'état des compétences, ou l'affichage du score, ne sont pas toujours perçus de manière intuitive.

Notre objectif est de simplifier la lecture de l'interface, en renforçant la hiérarchisation visuelle et en clarifiant les éléments de feedback liés à l'état du joueur.

Nous retravaillerons notamment :
- La jauge d'énergie, pour la rendre plus lisible dans son fonctionnement par paliers ;
- La représentation des compétences, avec des visuels plus clairs sur leur disponibilité et leur puissance ;
- Le système de score, pour qu'il soit compris instantanément et associé à une valeur de performance.

Projection Octobre à Décembre :

Cette phase correspond à la finalisation de la préproduction du projet. À ce stade, notre objectif est de rendre le prototype entièrement fonctionnel et représentatif de notre vision complète du jeu. Avec ces ajouts, Kill Dem'On gagnerait en profondeur, en rejouabilité, et en cohérence systémique. Le joueur aurait alors toutes les raisons de relancer des parties pour optimiser ses choix, s'adapter aux ennemis, et atteindre des scores toujours plus élevés. Le combat final contre le boss serait repensé pour représenter un véritable climax, et le level design serait construit pour soutenir pleinement les compétences du joueur.

1. Système de modules : vers un gameplay personnalisable

Le système de modules faisait partie de notre design initial, mais avait été mis de côté faute de temps et de ressources. L'idée est simple : chaque compétence du jeu (sprint, saut, pilonnage, etc.) serait représentée par un module, et au début de chaque partie, le joueur pourrait composer son propre build.

Les modules se diviseraient en plusieurs types :

- Modules actifs (liés à des compétences déclenchables)
- Modules passifs (buffs : vitesse, énergie, dégâts...)
- Modules négatifs (des debuffs qui ajoutent du challenge en échange de bonus ailleurs)

Ce système apporterait une véritable profondeur stratégique, en obligeant le joueur à faire des choix avant de commencer une partie. Il encouragerait aussi la rejouabilité, avec des combinaisons de builds variées à tester pour obtenir le meilleur score possible.

2. Nouveau boss : une vraie rencontre marquante

Le boss actuel, un gros dashoot, fonctionne sur le plan technique, mais son comportement reste basique et son design peu mémorable.
Pour finaliser la préproduction, nous voulons concevoir un nouveau boss entièrement original, aussi bien dans son apparence visuelle que dans son système de comportement.

Ce boss devra proposer une vraie rupture de rythme, des patterns clairs mais menaçants, et obliger le joueur à mobiliser toutes ses compétences acquises pendant la partie. Il marquera la fin de la session comme un moment fort, un véritable test de maîtrise, et non un simple obstacle final.

3. Level Design adapté aux compétences

Enfin, un travail de fond devra être mené sur le level design, pour qu'il corresponde parfaitement :

- Aux capacités du joueur (dash, saut, vitesse)
- Aux types d'ennemis rencontrés
- Aux rythmes de jeu attendus (zones d'intensité, respirations, verticalité)

Un bon level design ne se contente pas d'être agréable à explorer : il doit mettre en valeur les mécaniques du jeu, encourager les bons réflexes, et proposer des situations variées pour renouveler le challenge tout au long de la partie.

Projection Janvier à Avril :

Cette dernière phase correspond à la production finale de notre projet. Le cœur du gameplay ayant été solidement établi durant la préproduction, nous n'avons aucune volonté de bouleverser les mécaniques principales. Notre objectif est désormais de renforcer la rejouabilité et d'apporter de la diversité pour prolonger l'intérêt du jeu et offrir aux joueurs de nouvelles expériences sur une base déjà maîtrisée.

1. Nouveaux niveaux, nouveaux biomes

Nous prévoyons la création de plusieurs nouveaux niveaux, chacun apportant :

- Un biome visuel distinct (ambiance, décors, atmosphère)
- De nouveaux ennemis, conçus pour surprendre ou casser les habitudes du joueur
- Un nouveau boss, adapté à la logique du biome et aux mécaniques du joueur
- Un level design repensé, avec des topographies différentes, exploitant différemment les dashs, les sauts ou les lignes de tir

Cette diversité permettra aux joueurs de rejouer avec plaisir, en redécouvrant les mécaniques sous de nouveaux angles, sans jamais perdre la sensation de fluidité et de puissance centrale à notre expérience.

2. Système de succès et défis

Pour encourager les joueurs à repousser leurs limites et à explorer toutes les possibilités du jeu, nous intégrerons un système de succès.
Ce système sera accessibles via un menu dédié et contiendra des objectifs secondaires de difficulté variable, pensés pour :

- Récompenser la créativité (ex. : finir le jeu sans tirer une seule balle)
- Mettre à l'épreuve la maîtrise (ex. : terminer un niveau en moins de 2 minutes)
- Inciter à tester des builds ou des stratégies spécifiques (ex. : n'utiliser que le dash)

En plus d'augmenter la durée de vie, ces succès permettront de valoriser différents styles de jeu et d'offrir un nouvel objectif à celles et ceux qui ont déjà terminé le jeu une première fois.

Remerciements :

Nous tenons à remercier toutes les personnes qui ont contribué, de près ou de loin, à la réalisation de Kill Dem' On.
Que ce soit à travers des avis pertinents, des conseils techniques, ou simplement en prenant le temps de playtester nos différentes versions, votre aide nous a été précieuse à chaque étape du projet.
C'est aussi grâce à vous que le jeu a pu évoluer, gagner en clarté, en profondeur et en efficacité.

Un merci tout particulier à Alexis Lambert, qui a composé et produit l'ensemble des musiques du jeu.
Son travail sonore apporte une identité forte et cohérente à l'univers de Kill Dem' On, renforçant à la fois l'intensité des combats et l'immersion du joueur.
Sa contribution a été essentielle pour faire de notre prototype une expérience complète et mémorable.