

# DYING LIGHT

## ANALYSE THÉORIQUE

### 1. De quel type de jeu s'agit-il ?

Dying Light est un jeu de survie en monde ouvert avec des éléments de parkour et de combat à la première personne. Le jeu se déroule dans un environnement post-apocalyptique infesté de zombies.

### 2. Après une heure de jeu, nommez les différentes mécaniques de gameplay rencontrées.

Après une heure de jeu, les différentes mécaniques de gameplay rencontrées passent par le parkour, le combat au corps à corps, la gestion des ressources, la fabrication d'objets, le cycle jour/nuit avec des variations de difficulté, ainsi que des missions et quêtes secondaires.

### 3. Donnez en détail l'originalité et/ou l'importance d'une/des mécanique(s) de gameplay.

Une des mécaniques importantes de Dying Light est son système d'arbre de compétences, l'un des éléments centraux du gameplay. L'arbre de compétences offre aux joueurs la possibilité de personnaliser leur progression en choisissant parmi diverses compétences, allant du combat au parkour en passant par des améliorations de survie. Cette mécanique est cruciale car elle permet aux joueurs de définir leur style de jeu et de s'adapter aux défis rencontrés.

#### 4. L'aspect musical / graphique / game design / animation / algorithmique est/sont-il(s) primordial/aux dans ce jeu ? Pourquoi ? On détaillera au maximum son/leurs importance(s).

- Musicalement, la bande-son accompagne les moments tendus et dramatiques, elle arrive souvent au bon moment/timing. Cela permet de renforcer l'expérience émotionnelle ressentie par le joueur.
- Graphiquement, l'environnement détaillé et le design des zombies et des armes contribuent à l'immersion et à l'atmosphère post-apocalyptique/terrifiante.
- Au niveau du Game design, la variété de missions/quête, d'armes, de compétences, et la bonne qualité du level design (tours, zone Safe, forts, verticalité), permettent d'éviter un détachement du joueur envers le jeu tout en limitant une répétition trop contraignante du jeu.
- L'animation, fluide et réaliste du parkour contribue à l'authenticité des mouvements du personnage, offrant une expérience de déplacements agréable et satisfaisante.
- Sur le plan algorithmique, la gestion des cycles jour/nuit, des niveaux de compétences et de l'IA des zombies sont cruciales pour maintenir un niveau de défi approprié. De plus, le code semble être assez optimisé pour ne pas avoir de lag sur une config peu puissante.

## 5. Ce jeu vous a plu. Pourquoi ?

J'ai pu jouer à **Dying Light** en **coop** avec un ami, le jeu m'a vraiment plu. Le **mode coop** est vraiment très drôle, voir son pote se faire **attaquer** par les **zombies** ou le **tuer** en tirant sur un **bidon explosif** est vraiment cool. Les **mécaniques de compétitions coop** sont **intéressantes** à faire, ne serait-ce que par les **récompenses** gagnées que par le **côté fun** des épreuves (tuer le plus de zombies, course, fouille de trésors).

La **trame scénaristique** est bien **pensée** et **peu répétitive** avec une bonne durée de vie, un **dlc** peut venir en **rajouter**.

Les **compétences** sont vraiment **intéressantes** à développer, que ce soit en **agilité** (grappin, sauter sur les zombies, wall jump) ou en **combat** (coup lourd, plaquer un zombie, les objets de stun).

Globalement, Dying Light (en coopération ou non) offre une expérience vraiment **divertissante, captivante** et **fun**, sans trop de **bugs** après les mises à jours.

Le jeu a vraiment été super agréable à jouer !

# PRACTIQUE :

Définissez, selon vous, la mécanique de gameplay principale du jeu, et reproduisez là sous le moteur de jeu de votre choix (vous pouvez faire cette mécanique sur un projet en 2D ou 3D).

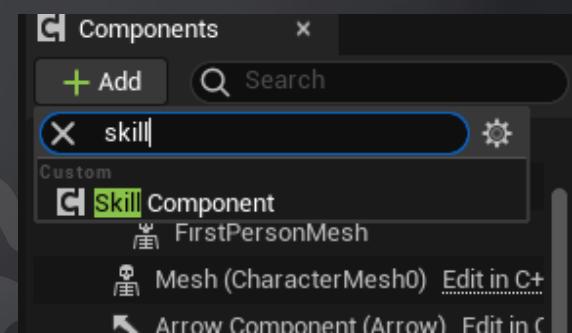
Ce jeu possède plusieurs mécaniques principales :

- Parkour fluide
- Cycle jour/nuit
- Combat à la première personne
- Système de compétences
- Mode coopératif

J'ai décidé de refaire le **Système de compétences** du jeu sur le moteur Unreal Engine

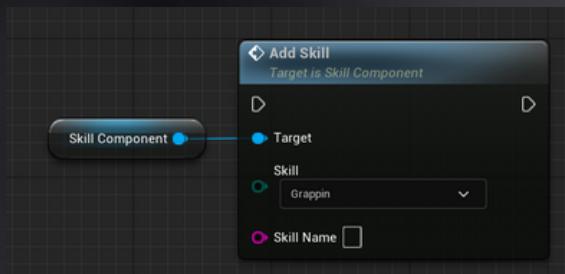
Pour utiliser le système de skill, le component "SkillComponent" doit être ajouté.

Le component possède une map contenant le nom de la compétence (string) et son niveau (integer).



Deux fonctions sont nécessaires pour utiliser le component.

"AddSkill" pour ajouter une compétence à la map  
(override du string sur l'enum).



"QuerySkill" requête pour savoir si le joueur possède la compétence et le niveau de la compétence.

