



PROJET ZELDA LIKE



PROJET ZELDA LIKE : AVANT-PROPOS

Contexte

Ce projet a été réalisé dans le cadre des cours de **Level Design, Direction Artistique, Infographie 2D et Programmation** à l'ETPA de Rennes, du 8 mars au 10 avril 2023.

Le sujet portait sur la création d'un **Zelda Like**, un jeu d'action-aventure en 2D, vu de dessus. Ce document expliquera en détail les recherches et différentes étapes de création pour aboutir à un résultat, ainsi que les différentes adaptations et remises en question face aux contraintes identifiées au cours du développement.

Ressources

L'entièreté du code source et des assets est accessible sur [Github](#), et hébergée via la fonctionnalité “pages”. Le jeu est jouable à l'adresse suivante : https://alanoixdecoco.github.io/GD1B_AVENTURE_SALAUN.



ETPA Rennes



Alan SALAÜN



PROJET ZELDA LIKE : TABLE DES MATIÈRES

AVANT-PROPOS

TABLE DES MATIÈRES

CAHIER DES CHARGES

GAME DESIGN

LEVEL DESIGN



PROJET ZELDA LIKE : CAHIER DES CHARGES

Programmation

Le jeu devait être développé à l'aide du framework **Phaser 3**, un framework **javascript** qui permet la création de jeux pour navigateurs sans installation d'outils supplémentaires.

Il devait également présenter une liste de **fonctionnalités** :

- contrôles au clavier et à la manette
- ennemis avec un comportement et qui interagissent avec le joueur, peuvent faire perdre de la vie et être battus
- items collectibles créés à la suite de la mort des ennemis et directement présents sur la carte
- inventaire, indicateur de vie et compte monétaire visibles à l'écran ou dans un menu
- deux items permettant de déverrouiller une capacité
- un item et une résolution d'action permettant d'ouvrir un passage jusqu'alors bloqué
- navigation au sein d'un niveau par déplacement de la caméra sur le joueur
- navigation entre plusieurs niveaux par un changement de scène



PROJET ZELDA LIKE : CAHIER DES CHARGES

Infographie 2D

La création de plusieurs assets était obligatoire :

- concept art du personnage principal
- animations du personnage principal
- décor avec éléments d'environnement
- écran d'accueil et logo
- éléments d'interface (UI en jeu)

Direction Artistique

Aucun rendu n'était demandé, la réflexion et la cohérence des éléments demandés dans les autres matières étant le sujet de la notation.

Level Design

Un document de pré-production présentant la démarche de création était requis.



PROJET ZELDA LIKE : GAME DESIGN

3C : Character

Qui ? Le joueur contrôle **BOB**, un personnage bipède prenant la forme d'un ordinateur des années 80 avec des jambes et des bras.

Actions ? En contrôlant BOB, le joueur peut :

- entrer en collision avec l'environnement (le vide agit comme un mur invisible qui empêche sa chute)
- subir des dégâts en touchant les ennemis, se faisant toucher par une balle ou touchant des pics
- tuer des ennemis
- ramasser des armes et des items au sol qui affectent sa santé et son inventaire
- ouvrir des portes verrouillées
- détruire des murs destructibles
- se déplacer au dessus du vide grâce à des prises à l'aide du grappin
- tirer des prises interrupteurs à l'aide du grappin



Figures 1 & 2. Sprites animé de BOB



PROJET ZELDA LIKE : GAME DESIGN

3C : Camera

Type de vue ? Le jeu se déroule en vue de dessus dans un environnement en 2D.

Placement ? La caméra est placée proche du joueur avec une zone de vue de 16×9 tuiles, le joueur occupant environ 1.5 tuiles en hauteur et 1 tuile en largeur.

Plan ? Le plan est dynamique, la caméra suit le joueur en se heurtant aux limites de la carte.

La caméra tremble également lorsqu'une entité subit des dégâts pour créer un retour de ses actions au joueur et rendre lisible le jeu même dans des zones très actives.

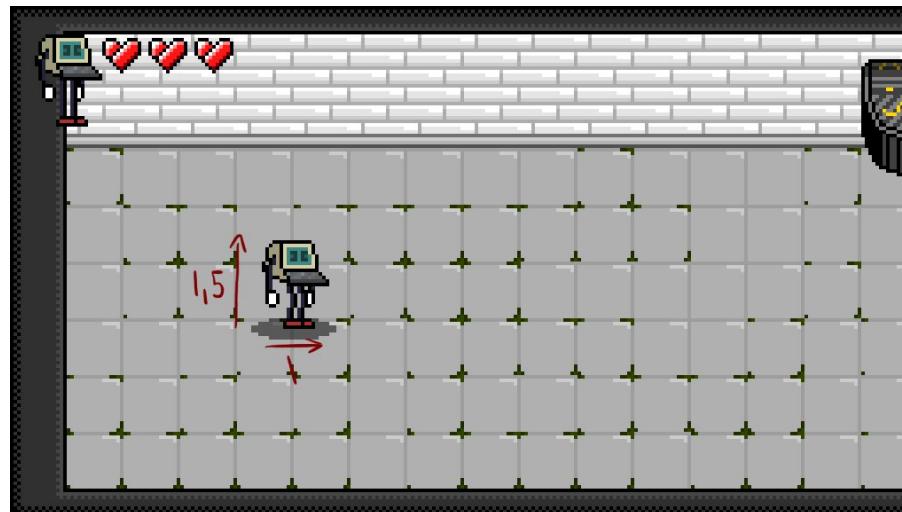


Figure 3. Zone de vue et dimensions du joueur



PROJET ZELDA LIKE : GAME DESIGN

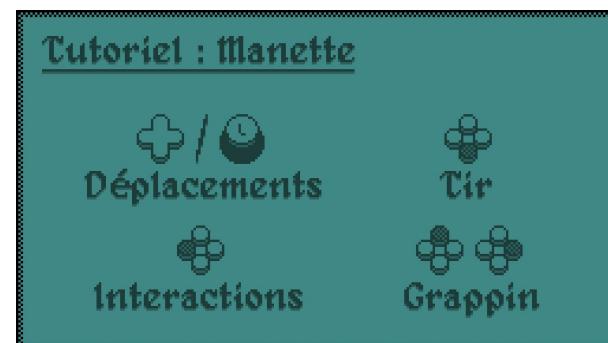
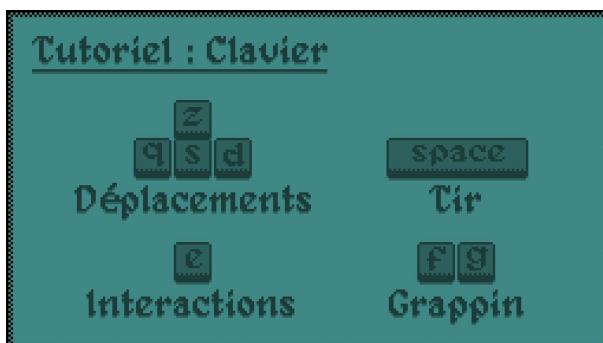
3C : Controls

Appareil ? Le jeu est jouable à l'aide d'un clavier ou d'une manette et la navigation dans les menus se fait via la souris ou un écran tactile.

Input ? Le joueur peut saisir les différentes inputs :

- Déplacements : haut/Z, bas/S, gauche/Q, droite/D et les diagonales qui peuvent en découler au clavier, D-PAD et les diagonales qui peuvent en découler ou joystick gauche pour la manette.
- Interaction : E au clavier, bouton de gauche selon le modèle de manette
- Attaque : Espace au clavier, bouton du bas selon le modèle de manette
- Capacité spéciale : F/G au clavier, bouton de droite/haut selon le modèle de manette
- Navigation dans les menus : Clic gauche/droite de souris selon le système, appui sur écran tactile

L'entrée joystick sur la manette est traitée de telle manière que les directions restent limitées à 8 différentes.



Figures 4 & 5. Écrans de présentations des touches clavier/ boutons manette



PROJET ZELDA LIKE : GAME DESIGN

3C : Controls

Déplacements ? Le joueur peut se déplacer, tirer et lancer son grappin dans les 8 directions permises par les inputs. Il évolue dans un environnement sans gravité, se déplaçant dans la largeur et la profondeur de celui-ci.

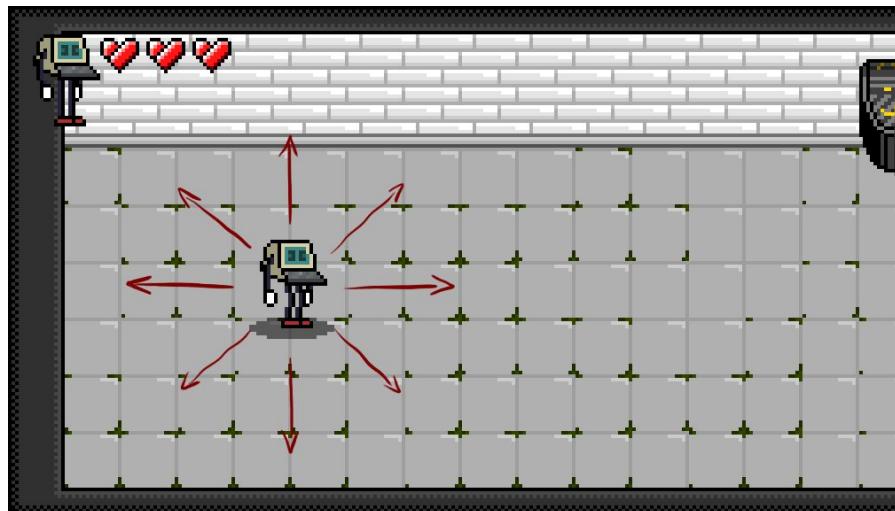


Figure 6. Les 8 directions du joueurs



PROJET ZELDA LIKE : GAME DESIGN

Mécaniques abandonnées

Glitch : Il était initialement prévu d'intégrer une mécanique de glitch, celle-ci se serait présentée comme une capacité permettant l'esquive de projectiles.

Gant de boxe : Le gant de boxe aurait permis la destruction des éléments de décor destructibles, cependant la présence d'armes rendait cette capacité trop limitée (aux environnements) et pas assez intéressante de ce fait.

Vol d'armes / récupération d'items au grappin : Le vol d'armes aux ennemis et la récupération d'items à l'aide du grappin a été placée en bas de ma liste de tâches, car elle permet un gameplay plus fluide mais entraîne énormément de modifications dans le code pour être implémentée, et beaucoup de tests et d'équilibrage du côté du level design.

Déplacement par écran : Pourtant implémentée et fonctionnelle, cette mécanique de mouvement de caméra par écran entier a été retirée car elle nuisait fortement à la fluidité de la circulation et la liberté de mouvement durant les combats qui sont pourtant le cœur du jeu.



PROJET ZELDA LIKE : LEVEL DESIGN

Croquis de la carte

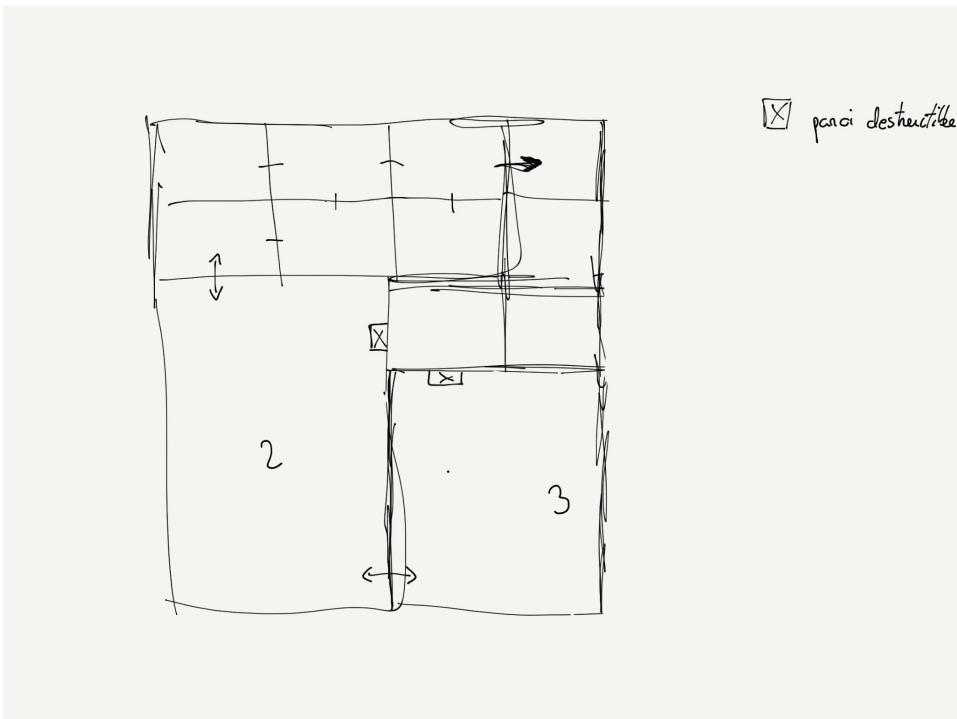


Figure 7. Premier croquis des zones de la carte

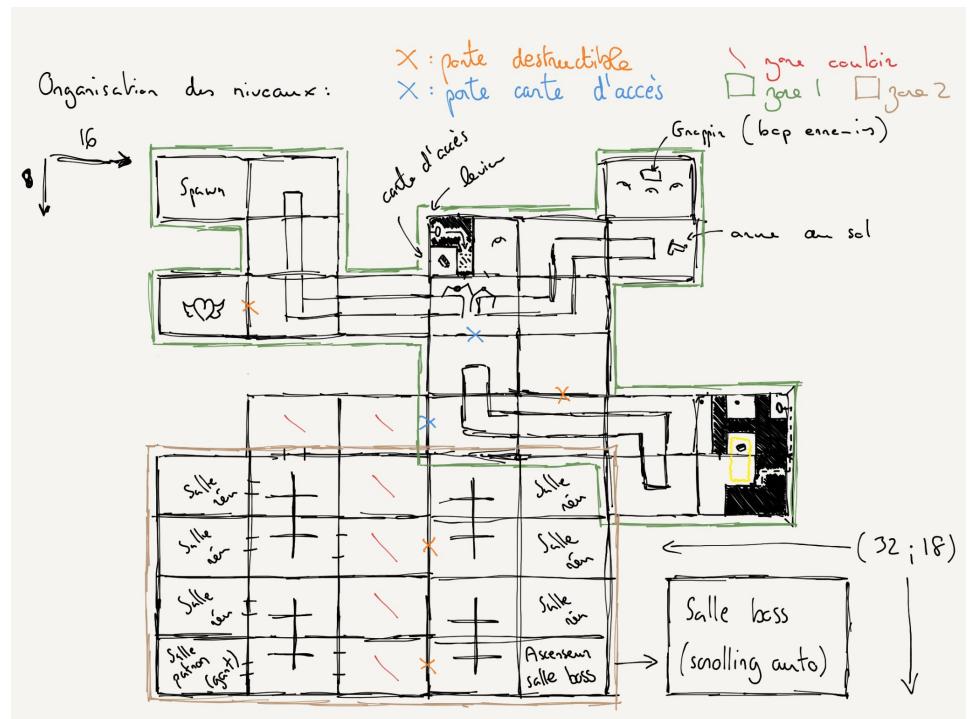


Figure 8. Croquis final avant création de la carte placeholder



PROJET ZELDA LIKE : LEVEL DESIGN

Analyse du croquis

Zones : Sur le croquis final on voit la volonté de créer 2 zones distinctes, une première zone “usine” composée d'un enchevêtrement de convoyeurs et une seconde zone “open space / bureaux”.

Une troisième partie de niveau est également présente, la salle du boss. L'objectif était de créer un niveau qui soit indépendant de celle-ci pour créer une ambiance plus sombre et plus vaste favorisant le combat sans gêner la circulation dans les autres zones.



PROJET ZELDA LIKE : LEVEL DESIGN

Analyse du croquis

Backtracking : La majorité du backtracking engendré par le parcours de la carte était lié au gant de boxe, une capacité qui n'a pas été conservée (voir [mécaniques abandonnées](#)). En effet les murs destructibles pouvant être cassés dès le début du jeu à l'aide d'une arme, il fallait remplacer certaines zones pour exploiter la capacité du grappin.

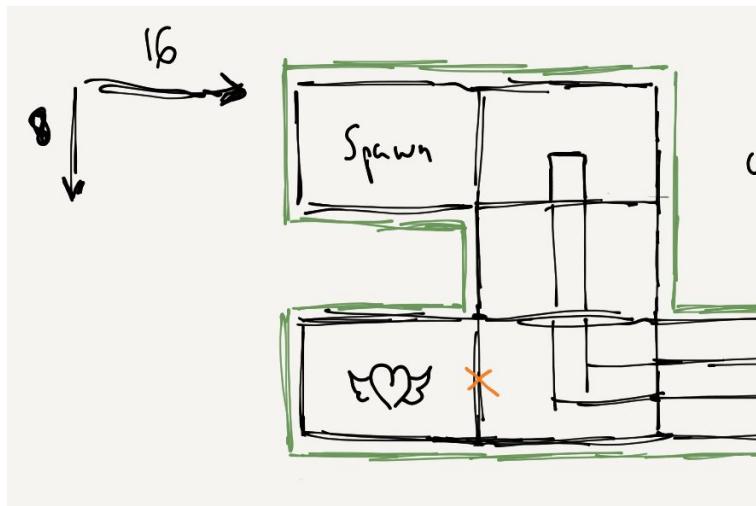


Figure 9 Passage initial favorisant le backtracking

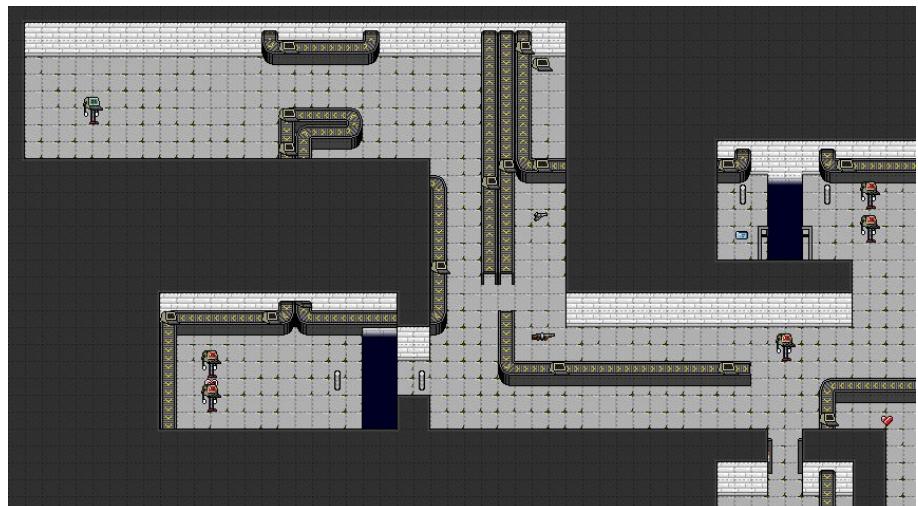


Figure 10. Même passage mais centré sur la mécanique de grappin dans la version finale



PROJET ZELDA LIKE : LEVEL DESIGN

Analyse du croquis

Intérêt : La carte initiale s'est avérée très répétitive, j'ai donc choisi de retirer de nombreuses parties de la seconde zone principalement.

Malgré la justification possible de l'architecture de cette zone par l'aspect uniforme et répétitif des open spaces, la circulation s'est avérée très difficile de part la difficulté d'identifier sa position dans le niveau, ce qui affectait également la tension du joueur, qui, après avoir tué tous les ennemis se retrouve à parcourir le vaste niveau dans l'espoir de trouver rapidement comment le terminer.

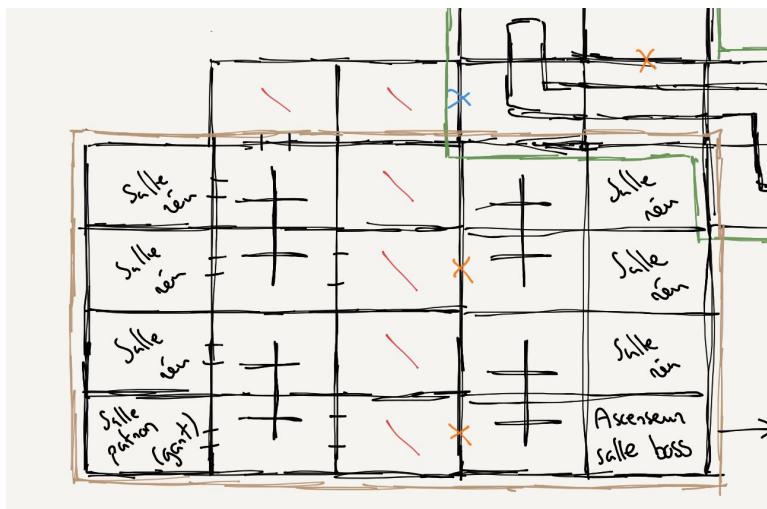


Figure 11. Zone 2 dans le croquis final



Figure 12. Zone 2 dans le jeu



PROJET ZELDA LIKE : LEVEL DESIGN

Cartes placeholder et finale

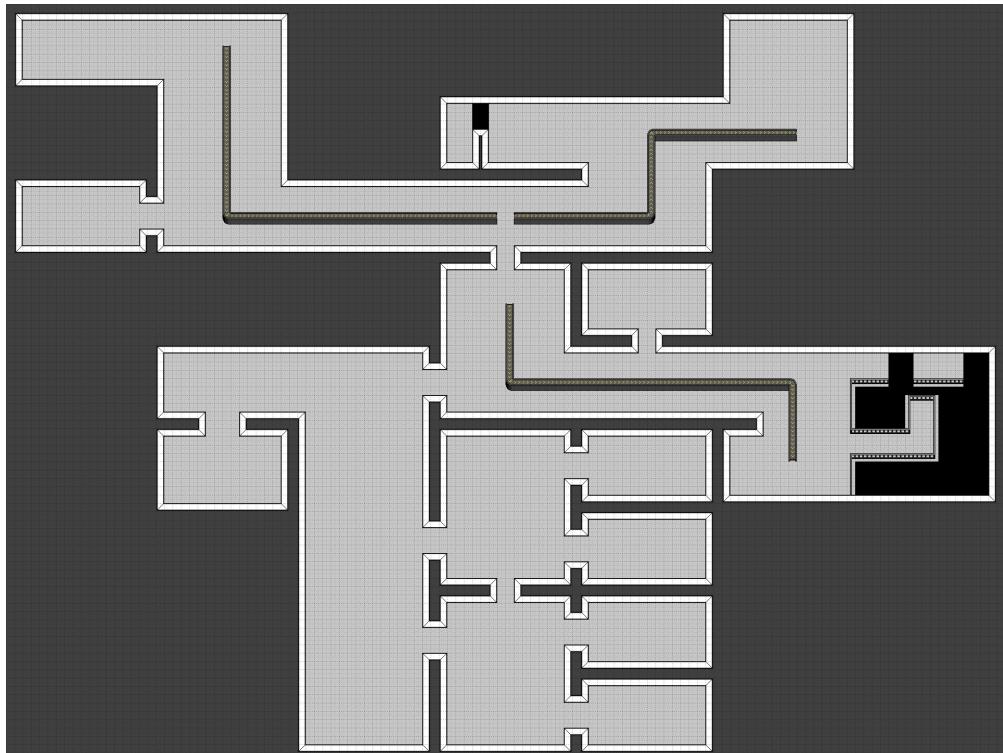


Figure 13. Carte placeholder réalisée sur Tiled



Figure 14. Carte finale réalisée sur Tiled



PROJET ZELDA LIKE : LEVEL DESIGN

Présentation des zones

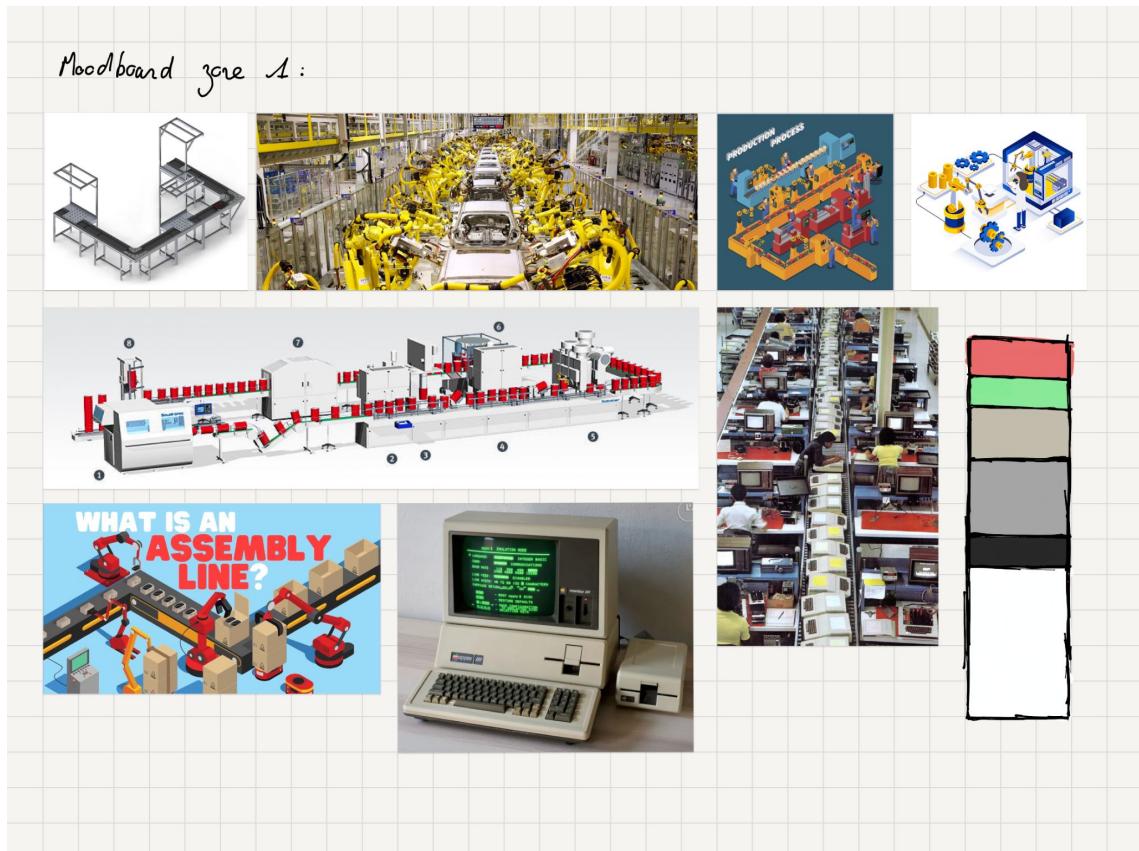


Figure 15. Moodboard de la première zone

Moodboard de la zone 1 :

La première zone est une usine d'ordinateurs, elle s'inspire des éléments mécaniques présents dans des usines à la chaîne, comme les convoyeurs, les employés et les bras robotisés.

Les deux derniers éléments n'ont pas été retenus dans le design pour la cohérence de l'époque et du contexte (l'usine est déserte).

La palette de la première zone nous plonge dans un mélange de couleurs froides avec le blanc et le gris, et de plastique vieillit avec le gris légèrement jauni.



PROJET ZELDA LIKE : LEVEL DESIGN

Présentation des zones



Figure 16. Première zone



PROJET ZELDA LIKE : LEVEL DESIGN

Présentation des zones

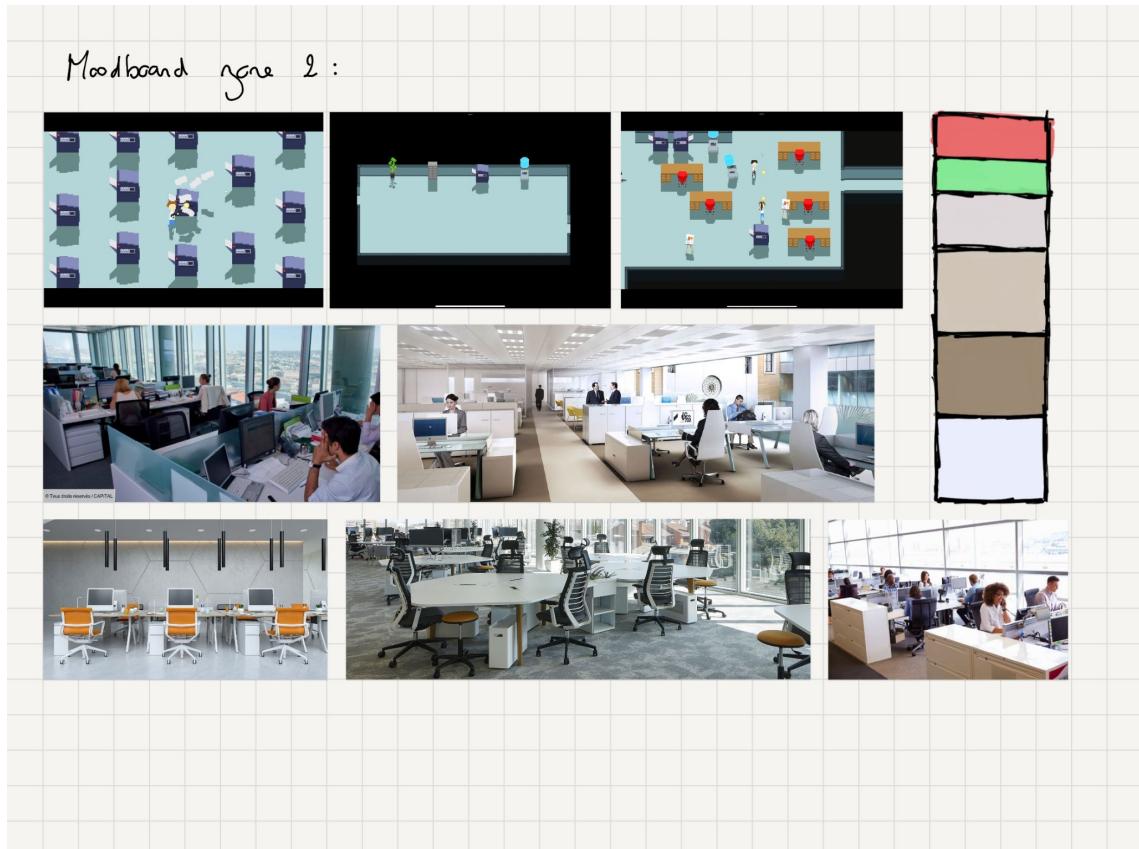


Figure 17. Moodboard de la seconde zone

Moodboard de la zone 2 :

La seconde zone est un open space, elle adopte une palette de couleur très terne dans les tons de l'époque, une organisation répétitive et très uniforme.

La présence de très peu d'objets sur les bureaux a aussi été retenue.



PROJET ZELDA LIKE : LEVEL DESIGN

Présentation des zones



Figure 18. Seconde zone



PROJET ZELDA LIKE : LEVEL DESIGN

Présentation des zones

Rappel de zone 1 : La seconde zone comporte à droite une section utilisant les assets de la première zone, ce rappel de la première zone impose au joueur de retourner dans l'environnement du début de la carte alors qu'il a déjà commencé à parcourir la seconde zone. Sans forcer le joueur à faire demi-tour, on réalise ainsi un retour en arrière qui permet d'imbriquer les deux zones l'une dans l'autre.

Étant donné la nature du projet et sa durée de développement, cette technique me permet d'éviter une trop grande redondance tout en utilisant le même nombre d'assets.



Figure 19. Rappel de la zone 1 dans la seconde zone



PROJET ZELDA LIKE : LEVEL DESIGN

Présentation des zones

Moodboard boss level:

→ Zone en grande partie vide (priorité au combat / patterns de boss)

→ Ambiance sombre, visibilité limitée

The moodboard includes four screenshots: 1) A screenshot from a game showing a large, dark, open area with a boss pattern on the floor. 2) A screenshot from a game showing a dark, narrow corridor with a boss pattern on the floor. 3) A screenshot from a game showing a large green spherical boss in a room with blue walls. 4) A screenshot from a game showing a large yellow spherical boss in a room with red walls. To the right of the screenshots is a vertical color palette consisting of five horizontal bars: light red, light green, grey, dark grey, and black.

→ Dans les couloirs de la zone 2 (filtrage assombriissant)

→ Avec des rappels de lumières verte & rouge (= code, virus,...)

Figure 20. Moodboard de la zone de boss

Moodboard de la zone de boss :

La zone de boss prend place dans le bureau du directeur de l'usine, elle reprend donc la même palette que l'open space, mais toutes les couleurs se retrouvent assombries.

Pour l'architecture du niveau, j'ai choisi de m'inspirer d'Enter The Gungeon et de certains boss dans Zelda Minish Cap, en préférant une vaste zone qui laisse toute l'importance au boss et à ses patterns et ainsi conserve l'attention du joueur sur ceux-ci.



PROJET ZELDA LIKE : LEVEL DESIGN

Présentation des zones

Création de la zone de boss : Le projet n'embarque malheureusement pas de zone de boss, celle-ci impliquant une très grande quantité de nouveaux éléments.

Cette partie du jeu devrait s'intégrer après la conclusion des deux zones, au moment du changement d'étage en empruntant l'escalier.



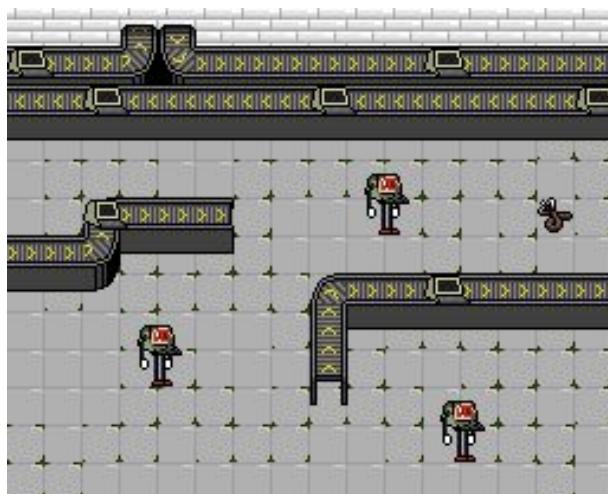
PROJET ZELDA LIKE : LEVEL DESIGN

Présentation des patterns

Convoyeurs, tables et décos : Les convoyeurs remplissent la majorité de la première zone, ils ont la particularité de créer une collision avec le joueur et les ennemis tout en laissant passer les balles qui évoluent virtuellement sur un plan plus haut que ceux-ci.

Les tables remplissent la même fonction que les convoyeurs mais dans le thème de la zone des bureaux.

Les décorations (ordinateurs éteints et livres) agissent comme des obstacles pour les balles et sont dispersés sur les convoyeurs et tables pour créer des safe zones ou les balles des ennemis ne peuvent pas atteindre le joueur, tout en conservant la tension du moment avec le point d'exclamation de l'UI de l'ennemi si il nous a détecté ainsi que son animation de marche.



Figures 21 & 22. Patterns de convoyeurs/tables et décorations



PROJET ZELDA LIKE : LEVEL DESIGN

Présentation des patterns

Open space : Il y a 2 salles de type open space dans la deuxième zone, l'objectif avec cette répétition est de créer un effet de labyrinthe pour perdre volontairement le joueur dans cet espace et ainsi le pousser à en explorer tous les recoins pour trouver les cartes d'accès et collecter les items qui faciliteront son parcours.

Vous noterez que les tables devant lesquelles des ennemis sont apparus sont dépourvues d'ordinateur, dans un souci de cohérence et un petit clin d'œil aux plus attentifs.



Figures 23 & 24. Les deux salles de type open space



PROJET ZELDA LIKE : PROGRAMMATION

La programmation de ce jeu a fait l'objet d'une attention particulière, ce projet me permettant d'expérimenter de nouvelles méthodes de travail et de les pratiquer.

Utilisation de constantes

Le code d'un jeu vidéo peut très rapidement devenir très lourd et de ce fait difficile à prendre en main après un certain temps, et très chronophage à modifier au fur et à mesure des tests. Pour pallier cette tendance j'ai choisi d'utiliser des constantes définies dans un fichier javascript unique.

Cette approche permet entre autres de faciliter la modification d'une valeur qui se retrouverait à de nombreux emplacements de différents scripts, et le réglage de divers aspects du jeu sans connaître précisément le fonctionnement de celui-ci (dans un projet en équipe par exemple).

```
Assets > Scripts > Components > Constants.js > ...
1 // Debug mode
2 const DEBUG = false;
3 const VERSION_NUMBER = "09.04.23_22.20";
4
5 // Inputs
6 const INPUT_ZERO_TOLERANCE = 0.1;
7
8 // Dimensions
9 const GAME_WIDTH = 256;
10 const GAME_HEIGHT = 144;
11
12 const TILE_SIZE = 16;
13
14 const ROOM_H_OFFSET = 16;
15 const ROOM_V_OFFSET = 16;
16
17 // Durations
18 const CAMERA_ROOM_TRANSITION_SPEED = 256; // px/s
19 const CAMERA_ROOM_TRANSITION_H_FACTOR = 2;
20 const CAMERA_ROOM_TRANSITION_V_FACTOR = 1;
21 const CAMERA_FADE_OUT_DURATION = 1000; // ms*
22 const CAMERA_FADE_IN_DURATION = 3000;
23
24 const CAMERA_UI_FADE_OUT_DURATION = 500;
25 const CAMERA_UI_FADE_IN_DURATION = 500;
26
27 const BULLET_LIFETIME = 3000; // milliseconds
28 const EMPTY_WEAPON_LIFETIME = 3000; // ms
29
30 const INVINCIBLE_DURATION_PLAYER = 1000; // ms
31 const INVINCIBLE_DURATION_ENEMY = 0; // ms
32 const INVINCIBLE_BLINK_INTERVAL = 100; // ms
33
34 const WEAPON_DEFAULT_RELOAD_DURATION = 1000; // ms
35 const WEAPON_RIFLE_RELOAD_DURATION = 2000; // ms
36 const WEAPON_REVOLVER_RELOAD_DURATION = 1000; // ms
```

Figure 0. Extrait du code des constantes



PROJET ZELDA LIKE : PROGRAMMATION

Développement itératif

Il est rare de trouver du premier coup la meilleure méthode pour arriver à un résultat, ce qui a souvent tendance à me bloquer dans le développement de nouvelles fonctionnalités. J'ai décidé après plusieurs recherches d'adopter une logique de développement par itérations : plutôt que remettre en question chacune de mes idées pour programmer un composant, je me contente d'une option "correcte", puis au fur et à mesure que j'ajoute des comportements à celui-ci, je le modifie pour l'optimiser ou le corriger.

Un exemple concret est la classe **Player**, celle-ci étant initialement une classe vierge encapsulant une instance de **Sprite**, puis une classe héritée de **Sprite**, avant de devenir une classe héritée de **Entity**, elle-même héritée de **Sprite** et permettant la création simplifiée d'ennemis sur le même modèle que le joueur.



Figure 0. Processus itératif de conception de la classe **Player**