BACK2REVENGE STUDIO

MACRO GAME DESIGN DOCUMENT

THE LAST FLOOR

STAFF :

El Mehdi Hamte – Game Designer and Developer
Otmane Aayachi – Sound Designer
Abdellah Hannache – Game Designer and Artist 2D

Présentation du jeu :

“The Last Floor: Out of Breath” is a **2D stealth/puzzle** platformer with a pixel art visual style.
The game is set inside a sealed, high-rise research tower where a toxic experimental gas known as **V-GAS** has begun to leak.
The player takes on the role of a technician partially resistant to the gas, whose objective is to climb five contaminated floors to reach the main control unit, disable the lockdown, and escape.
Core gameplay focuses on stealth, light environmental puzzle-solving, and managing gas mask resources, while avoiding mutated enemies that react dynamically to sound and light.
The project aims to deliver a short, intense survival horror experience, emphasizing tension, atmosphere, and strategic decision-making within a compact playtime.

Plateformes : PC (Windows), WebGl

Configuration Minimale (PC) :

- OS : Windows 10 (64-bit)
- CPU : Intel Core i3 / AMD Ryzen 3
- RAM : 4 GB
- GPU : Carte graphique compatible DX10 (Intel HD Graphics 4000 ou mieux)
- Espace disque : 500 Mo

Environnement de développement :

- **Moteur de jeu : Unity 2022.3 LTS.**
    - *Argumentation :* Cette version est stable (Long Term Support). Unity excelle dans la 2D et propose des outils natifs robustes (Tilemap, 2D Physics, URP 2D Lighting) qui nous permettent de tenir le délai de 6 semaines sans avoir à développer un moteur de rendu ou de physique nous-mêmes.

- **IDE :** Visual Studio 2022 ou Visual Studio Code (avec extensions C#).

- **Versionning :** Git (hébergé sur GitHub ou GitLab). Client : GitHub Desktop ou Sourcetree.

Risques techniques

Risque	Impact	Solution Technique
Pathfinding 2D complexe	Les ennemis restent coincés dans les murs ou ne trouvent pas le joueur.	Utilisation de NavMeshSurface 2D (intégré à Unity) au lieu de coder un A* personnalisé. Simplification des niveaux pour éviter les géométries trop complexes.
Performance des lumières	Trop de lumières dynamiques (2D Lights) font chuter les FPS.	Limitation du nombre de lumières actives. Utilisation du "Light Culling" et réglage de la "Render Scale" si nécessaire. Baking des lumières statiques si possible.
Gestion des états (Spaghetti Code)	Le code devient impossible à débuguer avec les interactions Gaz/Masque/Ennemis.	Application stricte du pattern State Machine pour le joueur et les ennemis. Séparation des responsabilités (Managers).
Délais courts (1.5 mois)	Fonctionnalités non finies.	Adoption d'une architecture modulaire (Feature Toggling). Si une mécanique est buguée à la fin, on peut la désactiver sans casser tout le jeu.

Pipeline de production

Graphismes (Pixel Art) :

Logiciel : Aseprite / Photoshop.

Export : .PNG ou .PSB.

- **Import Unity :**
  - **Texture Type :** Sprite (2D and UI).
  - **Sprite Mode :** Multiple (pour les sprite sheets).
  - **Pixels Per Unit (PPU) :** 16 ou 32 (fixe pour tout le projet).
  - **Filter Mode :** Point (No Filter) pour garder le look pixel art net.
  - **Compression :** None (pour éviter les artefacts sur le pixel art).

Level Design :

- **Outil :** Unity Tilemap Editor.
- **Données :** Création de "Tile Palettes" catégorisées (Sol, Murs, Arrière-plan). Utilisation de "Composite Colliders" pour optimiser la physique des sols.

Audio :

- **Formats : .WAV pour les SFX courts (décompressés en mémoire), .OGG pour les musiques (streaming).**

Les outils à développer

Compte tenu du délai court, nous minimisons les outils custom, mais ceux-ci sont essentiels :

- **Waypoints Editor : Un script simple avec des Gizmos (lignes visibles dans l'éditeur) pour permettre aux Level Designers de placer les points de patrouille des ennemis par simple "Drag & Drop".**
- **Gas Zone Manager : Un composant qui permet de définir visuellement une zone (BoxCollider2D trigger) et de régler la densité du gaz et la vitesse de réduction du filtre via l'Inspector.**

L'Architecture Logicielle

- **GameManager : Le chef d'orchestre. Il ne gère pas le gameplay direct, mais l'état de l'application.**
- **PlayerController : Gère la physique et les inputs. Il envoie des événements (ex: OnPlayerDeath) que le GameManager écoute.**
- **GasMaskSystem : Script modulaire attaché au joueur. Gère la logique "Porté/Non porté" et le timer.**

Techniques de rendu graphique & complexité

- **Pipeline : URP 2D Renderer.**
-
- **Shaders :**
  - ***Sprite-Lit-Default :* Shader standard URP pour que les sprites réagissent à la lumière.**
  - ***Custom Distortion Shader (Shader Graph) :* Un shader d'effet d'écran qui déforme l'image (UV displacement) pour simuler l'intoxication au gaz ou la chaleur.**

- **Stack Post-Process :**
  - ***Vignette :* S'intensifie quand le masque est porté (réduit le champ de vision).**
  - ***Bloom :* Pour faire briller les yeux des ennemis et le gaz toxique.**
  - ***Chromatic Aberration :* Augmente lorsque la santé est basse.**

- **Complexité :**
  - Scène 2D composée de Sprites (Quads).
  - Nombre de polygones négligeable.
  - *Draw Calls :* Optimisés via le "Sprite Atlas" d'Unity qui regroupe les sprites en une seule texture pour réduire les appels au GPU.

Performances

- **Objectif :** 60 FPS constants sur la configuration minimale.
- **Contraintes :** Les jeux de plateforme demandent une fluidité parfaite pour la précision des sauts.
- **Optimisation :**
  - Utilisation de l'Object Pooling pour les projectiles ou particules (ne pas faire Instantiate/Destroy en boucle).
  - Désactivation des scripts des ennemis (IA) lorsqu'ils sont trop loin de la caméra (Culling logic).

Mise en oeuvre technique des mécanismes de gameplay

1. Intelligence Artificielle (Ennemis)

L'IA utilise une Machine à États Finis (Finite State Machine - FSM).

- États : Idle -> Patrol -> Suspicious (Entend un bruit) -> Chase (Voit le joueur).
- Détection visuelle : Physics2D.Raycast vers le joueur. Bloqué par le layer "Ground".
- Détection sonore : Le joueur émet un "Cercle de bruit" invisible (Physics2D.OverlapCircle). Si l'ennemi est dans ce cercle, il passe en état Suspicious et va vers la source du bruit.

(Illustration recommandée : Un diagramme de flux montrant les transitions entre les états Patrol, Chase et Search)

2. Le Gaz et le Masque

- Logique : Une simple variable float filterLevel qui décroît dans Update().
- Rendu : Le gaz n'est pas une simulation de fluide (trop coûteux). C'est un système de particules couplé à des SpriteMasks et un Global Volume vert pour teinter l'écran.

Menus & HUD

- Technologie : Unity UI (uGUI).
- Implementation :
  - Canvas Scaler : Réglé sur "Scale With Screen Size" (Ref: 1920x1080) pour s'adapter à tous les écrans.
  - HUD : La barre d'oxygène et de santé sont mises à jour via des Events. Le script UI ne vérifie pas les valeurs à chaque frame, il attend que le Player envoie l'événement OnFilterChanged(float value).

Audio

- Audio Mixer : Utilisation de l'Audio Mixer d'Unity avec 3 groupes : Master, Music, SFX.
- Duck Volume : Quand le joueur porte le masque, un effet "Low Pass Filter" est activé sur le groupe Master via script pour étouffer les sons ambiants et simuler l'isolation du masque.

Convention d’écriture de code

Pour assurer la lisibilité et la collaboration rapide :

- **Langue :** Anglais (noms de variables/fonctions) et Commentaires en Français/Anglais.
- **Variables :**
  - public float MaxSpeed; (PascalCase)
  - private float _currentSpeed; (camelCase avec underscore pour les privés)
- Classes : public class PlayerController : MonoBehaviour (PascalCase).
- **En-têtes :**

// Auteur : [Nom]

// Date : 17/12/2025

// Description : Gère les déplacements et sauts du joueur.

- **Sérialisation :** Utiliser [SerializeField] private plutôt que public pour exposer des variables dans l'inspecteur tout en gardant l'encapsulation.

Equipe et organisation

- **Méthodologie :** Agile/Scrum simplifié (Sprints de 1 semaine).
- **Partage du code :** GitHub.
  - Branch main : Version stable et jouable uniquement.
  - Branch develop : Intégration des fonctionnalités.
  - Commits atomiques : "Ajout saut", "Correction bug collision" (pas de "update final v2").
- **Partage des données (Assets) :** Utilisation de Git LFS (Large File Storage) pour les fichiers binaires (PSD, WAV) afin de ne pas bloquer le repo.
- **Communication :** Discord pour le quotidien, Trello pour le suivi des tâches (To Do, Doing, Done).