Le Mans Université

Licence Informatique *2ème année*

Module 174UP02 Compte rendu Projet

Tales Of Depression

Asensi Ridde Juan Miguel Enrique
Juan.Asensi_Ridde.etu@univ-lemans.fr

April 11, 2025

https://github.com/aMiguel25/tales-of-depression

Contents

1 Introduction

Le projet Tales Of Depression est un jeu vidéo développé dans le cadre du module de projet en deuxième année de Licence Informatique à l'Université du Mans, durant la période de janvier à avril 2024. Ce jeu, de plateforme en 2D, a été conçu en utilisant le langage de programmation C, en s'appuyant sur la bibliothèque SDL (Simple DirectMedia Layer) pour la gestion des graphismes, des sons et des interactions avec l'utilisateur.

Le but principal du jeu est de guider un personnage à travers divers niveaux, en surmontant des obstacles, en résolvant des énigmes et en interagissant avec des objets. Le jeu plonge le joueur dans une atmosphère sombre et émotionnelle, explorant des thèmes de la dépression, de la solitude et des luttes intérieures, d'où le titre "Tales Of Depression". Ce jeu est un sidescroller, ce qui signifie que l'action se déroule de gauche à droite, avec une vue de profil du personnage et de son environnement.

Objectifs du projet L'objectif principal de ce projet était de développer un jeu complet qui inclut non seulement des mécaniques de jeu solides, mais aussi une dimension artistique et émotionnelle forte. À travers la création de personnages, d'environnements et de niveaux variés, le projet vise à offrir une expérience captivante et immersive au joueur tout en permettant aux membres de l'équipe de renforcer leurs compétences en programmation, en gestion de projet et en conception de jeux vidéo.

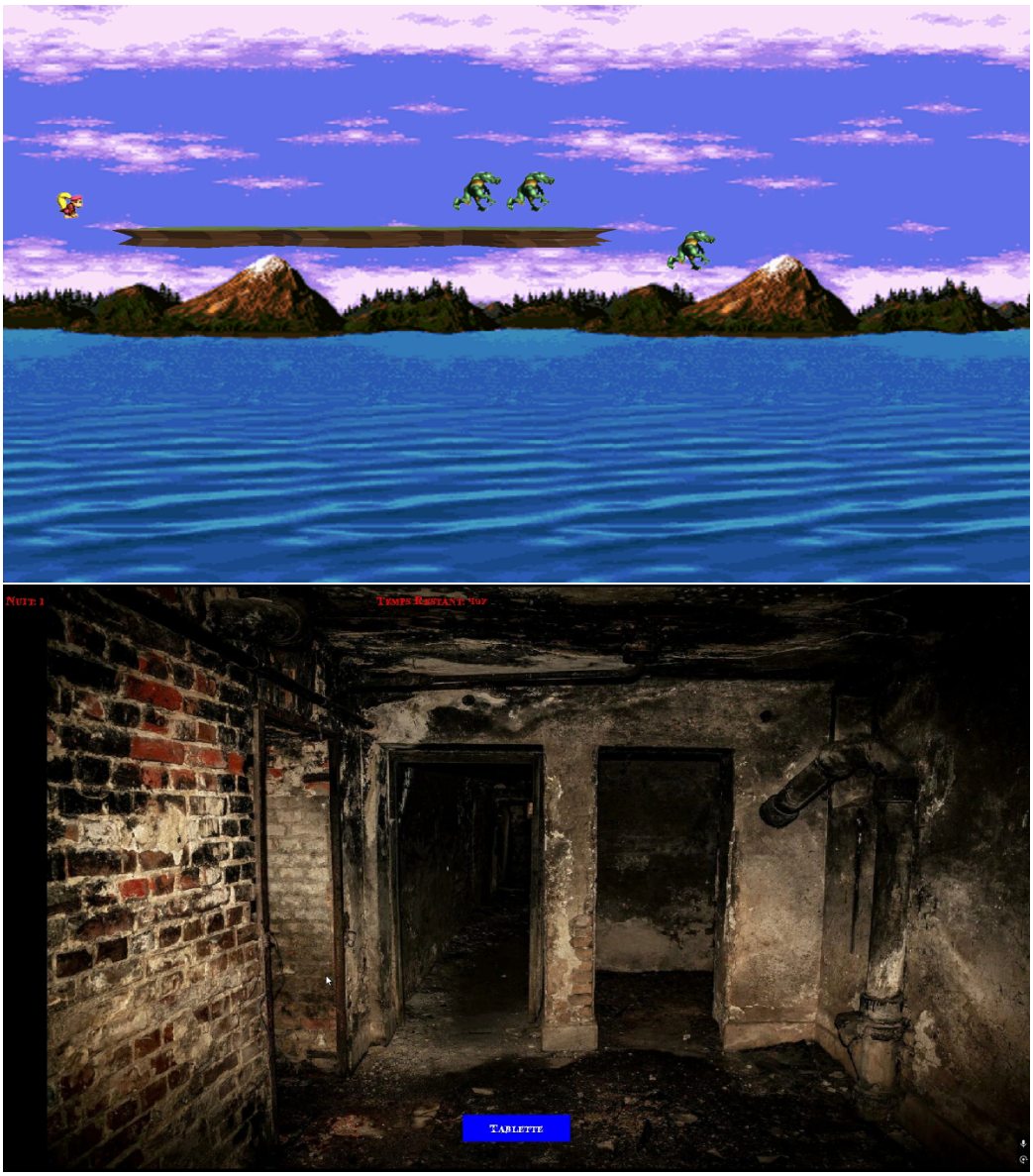
Caractéristiques du jeu Le jeu propose plusieurs fonctionnalités intéressantes qui enrichissent l'expérience du joueur :

- **Mécaniques de plateforme** : Le joueur contrôle un personnage qui doit naviguer à travers des niveaux variés, en sautant par-dessus des obstacles et en interagissant avec l'environnement.
- **Animations et gestion des ressources** : Les animations du personnage principal ont été réalisées à partir de fichiers GIF, découpés et convertis en images individuelles pour être utilisées comme textures dans le jeu.
- **Musique et effets sonores** : La bande sonore et les effets sonores ont été réalisés à partir de fichiers extraits de YouTube et modifiés à

l'aide du logiciel Audacity, créant une ambiance sonore qui renforce l'atmosphère du jeu.

- **Menu interactif et personnalisation** : Le jeu dispose d'un menu principal interactif où le joueur peut ajuster les paramètres (son, affichage, etc.) et démarrer la partie.
- **Easter egg** : Un jeu caché est accessible via un code secret dans le menu principal, offrant ainsi une expérience supplémentaire à ceux qui parviennent à le débloquer.

Image du premier niveau et cellier du jeu secret ci-dessous


Nuit 1
Temps Restant 107
Tablette

2 Conception

2.1 Cahier des charges

Il devait être prévu d'avoir plus de contenu dans ce projet, cependant les contraintes de santé dans l'équipe on beaucoup entravé la progression du projet.

L'objectif principal du projet était de créer un jeu complet avec une boucle de jeu fonctionnelle, une interface utilisateur, une gestion de ressources graphiques et audio, ainsi qu'un système de progression à travers différents niveaux. Le tout devait être implémenté avec un code structuré, modulable et documenté.

Fonctionnalités principales prévues :

- Déplacement du personnage principal dans un monde en 2D avec gestion des collisions.
- Interaction avec différents objets : plateformes, ennemis, éléments interactifs.
- Système de menu (accueil, pause, paramètres).
- Affichage d'un HUD (points de vie, score, etc.).
- Gestion des événements clavier pour le gameplay.
- Ambiance sonore (bruitages, musiques de fond).
- Éventuels effets visuels simples (animations de personnage, transitions).

Contraintes techniques :

- Utilisation exclusive du langage C.
- Utilisation de bibliothèques libres et compatibles multiplateformes.
- L'application doit pouvoir se compiler et s'exécuter sous Linux et Windows.
- Organisation du code source en modules séparés (par exemple : `main.c`, `player.c`, `map.c`, `menu.c`, etc.).
- Aucune fuite mémoire (usage systématique de `valgrind` pour valider).

3 Organisation du projet

3.1 Gantt prévisionnel

Le diagramme de Gantt prévisionnel a été utilisé pour planifier les différentes étapes du projet. Il a permis de définir un calendrier de travail, en indiquant les tâches à réaliser ainsi que les délais associés. Voici une représentation graphique de ce plan initial :

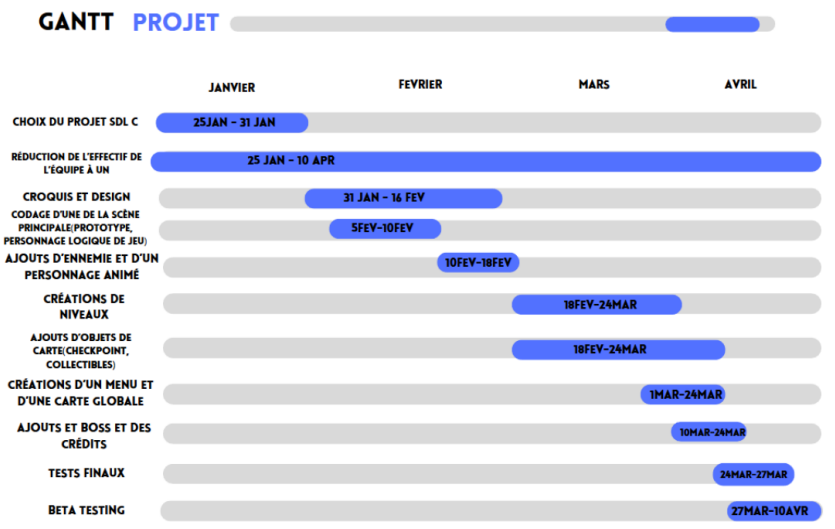


Figure 1: Diagramme de Gantt prévisionnel du projet

3.2 Outils de développement

Pour le développement du projet Tales Of Depression, plusieurs outils et technologies ont été utilisés afin de faciliter la gestion du projet, le développement, le test et la documentation. Voici une liste détaillée de ces outils :

- **Langage de programmation** : C
  - Le choix du langage C a permis une gestion fine des ressources et un contrôle total sur la mémoire et la gestion des textures. Cepen-

dant, cela a également engendré certains défis, notamment en ce qui concerne la gestion de la mémoire et les bugs de débordement.

- **Bibliothèque SDL (Simple DirectMedia Layer)** :
  - SDL a été utilisé pour gérer l'affichage graphique, les entrées (clavier, souris) et les sons. Cette bibliothèque est largement utilisée pour le développement de jeux 2D, ce qui nous a permis d'implémenter des fonctionnalités comme les animations de personnages, les déplacements, et la gestion des événements.
- **Éditeur de code** : Visual Studio Code
  - Visual Studio Code a été utilisé pour la plupart du développement, grâce à sa légèreté, sa prise en charge des langages C/C++ et ses nombreuses extensions.
- **Gestion de version** : Git et GitHub
  - Git a été utilisé pour la gestion des versions et la collaboration sur le code. GitHub a permis de stocker le code source et de suivre l'évolution du projet de manière collaborative.
- **Outils de création de ressources** :
  - **ezgif.com** : Outil en ligne utilisé pour découper des fichiers GIF animés et générer des images individuelles que nous avons ensuite utilisées comme textures dans le jeu.
  - **Audacity** : Logiciel de traitement audio utilisé pour éditer et créer les effets sonores et les musiques utilisées dans le jeu.
  - **YouTube** : Des ressources audio et musicales ont été extraites de YouTube à l'aide d'outils de conversion, pour enrichir l'expérience sonore du jeu.
- **Outils de gestion de projet** :
  - **Trello** : Utilisé pour la gestion des tâches et la planification du projet sous forme de tableaux et de cartes. Cela a permis de suivre l'avancement et d'affecter les différentes tâches aux membres de l'équipe.

  - **Gantt Project** : Utilisé pour créer le diagramme de Gantt prévisionnel et suivre l'évolution du projet à travers des mises à jour régulières.

- **Système d'exploitation** : Windows

  - Le développement s'est fait principalement sous Windows, avec une configuration qui supporte tous les outils mentionnés ci-dessus. Le jeu a été testé et exécuté sous ce système d'exploitation avant d'être potentiellement transféré vers d'autres plateformes.

4 Développement

4.1 Création des ressources

Sprites et animations Les animations des personnages, ennemis et objets interactifs ont été réalisées à partir de fichiers **GIF animés**. Ces GIF ont été découpés en plusieurs images individuelles (frames) à l'aide de l'outil en ligne ezgif.com, qui permet de transformer facilement une animation en une séquence d'images PNG.

Une fois extraites, ces images ont été intégrées dans le projet sous forme de tableau de textures SDL :

- Chaque frame est chargée dans une `SDL_Texture` à l'aide de `SDL_image`.
- Un tableau dynamique (ou une structure personnalisée) permet d'indexer ces frames et d'itérer dessus à chaque boucle d'animation.
- Le système de timing a été calibré pour afficher une nouvelle frame à intervalle fixe afin d'assurer la fluidité des animations.

Ce système est utilisé aussi bien pour l'animation du personnage principal que pour les ennemis, objets et effets visuels (comme les particules, flammes ou effets de hit).

Sons et musiques Les effets sonores et les musiques du jeu ont été obtenus à partir de contenus libres disponibles sur YouTube. Pour cela :

- Les pistes audio ont été téléchargées depuis YouTube à l'aide d'outils en ligne respectant les droits d'utilisation.

- Le logiciel **Audacity** a été utilisé pour :
  - Nettoyer les pistes (suppression de bruit, découpage précis).
  - Réduire la taille des fichiers en les exportant au format `.wav` ou `.ogg`.
  - Régler les volumes pour éviter les saturations pendant l'exécution du jeu.
- L'intégration s'est faite via la bibliothèque `SDL_mixer`, permettant de jouer plusieurs pistes en parallèle (musique de fond + bruitages).

Les sons sont joués lors d'événements spécifiques (saut, collision, victoire, etc.), tandis que la musique varie en fonction du niveau ou de l'état du jeu (menu, jeu, pause...).

4.2 Rendu du jeu

Le rendu du jeu est géré grâce à la bibliothèque **SDL** qui permet d'afficher à l'écran les éléments graphiques (sprites, tuiles, textes, etc.) de manière fluide et en temps réel. L'objectif est d'assurer une expérience visuelle cohérente, avec une bonne réactivité et une animation fluide.

Fenêtre et contexte de rendu Lors du lancement du jeu, une fenêtre principale est créée via `SDL_CreateWindow`, et un contexte de rendu matériel est initialisé via `SDL_CreateRenderer`. Ce renderer est utilisé tout au long du jeu pour dessiner les textures sur l'écran.

Gestion des couches graphiques L'affichage repose sur un système de rendu en plusieurs couches, dessinées dans un ordre précis :

1. **Arrière-plan** : ciel, montagnes, décors fixes (chargés une seule fois pour économiser des ressources).
2. **Carte du niveau** : tuiles représentant le sol, les murs, les plateformes, etc.
3. **Objets** : pièges, bonus, portes, plateformes mobiles, etc.
4. **Personnage principal et ennemis** : avec animations gérées par frame.

5. **HUD (Head-Up Display)** : barre de vie, score, messages temporaires, etc.

Chaque élément est dessiné à l'aide de `SDL_RenderCopy` (ou `SDL_RenderCopyEx` pour appliquer des effets comme la rotation ou le flip horizontal).

Animation et synchronisation Les animations sont gérées à partir d'un système de frames :

- Un compteur de temps (timer) permet de savoir quand passer à la frame suivante.
- Chaque sprite animé (personnage, ennemi, objets) possède son propre tableau de textures.
- Le nombre de frames par seconde est limité volontairement (ex. : 60 FPS) à l'aide de `SDL_Delay` ou d'un gestionnaire de temps précis.

Effets visuels simples Des effets basiques comme les clignotements (invincibilité temporaire), le fondu (transition entre menus et niveaux), ou les mouvements de caméra sont également gérés par le moteur. La caméra est simulée en décalant dynamiquement les coordonnées d'affichage selon la position du joueur, afin de suivre son mouvement dans le niveau.

Optimisation Le rendu a été optimisé en :

- Évitant les rechargements d'images pendant la boucle principale.
- Stockant les textures en mémoire dès le lancement du niveau.
- N'affichant que les objets visibles dans la fenêtre (culling).

Le rendu final est cohérent et fluide, même avec plusieurs animations simultanées, ce qui garantit une bonne jouabilité et une immersion satisfaisante.

4.3 Menu

Le menu principal constitue le point d'entrée du jeu. Il est affiché dès le lancement de l'application et permet à l'utilisateur de naviguer vers les différentes fonctionnalités du jeu : démarrer une partie, accéder aux paramètres, ou quitter le programme.

Structure du menu Le menu principal est composé des éléments suivants :

- Un titre graphique ou textuel animé.
- Plusieurs boutons : **Jouer**, **Paramètres**, **Quitter**.
- Éventuellement une musique de fond spécifique et une animation d'arrière-plan ou un fond statique.

Navigation La navigation dans le menu se fait via les touches clavier (flèches haut/bas ou Z/S) et la touche Entrée pour valider. Un système de surbrillance visuelle indique quel bouton est actuellement sélectionné. Le joueur peut ainsi naviguer intuitivement entre les différentes options.

Implémentation Chaque bouton du menu est représenté par une structure contenant :

- Le texte du bouton ou son image.
- Sa position à l'écran.
- Son état (sélectionné / non sélectionné).

Les boutons sont dessinés à l'aide de textures SDL ou simplement via le rendu de texte avec `SDL_ttf`. Un effet visuel (changement de couleur ou d'image) est appliqué au bouton sélectionné pour le différencier visuellement.

Boucle de menu Le menu principal fonctionne dans une boucle dédiée, distincte de celle du jeu :

1. Affichage des éléments du menu.
2. Gestion des événements clavier.
3. Mise à jour de l'état sélectionné.
4. Affichage du rendu avec `SDL_RenderPresent`.
5. Si l'utilisateur sélectionne "Jouer", la boucle se termine et le jeu commence.

Ambiance et ergonomie Une attention particulière a été portée à l'ambiance du menu : musique calme, fond visuel immersif et animations douces permettent au joueur d'entrer dans l'univers du jeu dès les premières secondes.

4.4 Paramètres

Le menu des paramètres permet au joueur de personnaliser certains aspects du jeu selon ses préférences. Ce menu est accessible depuis le menu principal et a été pensé pour rester simple et intuitif.

Options disponibles Les paramètres actuellement disponibles sont les suivants :

- **Volume de la musique** : permet d'augmenter ou de diminuer le volume de la bande-son.
- **Volume des effets sonores** : réglage indépendant des bruitages (saut, collision, victoire...).
- **Plein écran / Fenêtré** : permet de passer d'un mode fenêtré à un affichage plein écran (option selon compatibilité).
- **Touches de contrôle** : visualisation (et éventuellement modification) des touches utilisées pour jouer.

Implémentation technique Le menu des paramètres repose sur une interface similaire à celle du menu principal. Chaque option est représentée par un élément sélectionnable à l'aide des touches haut/bas, avec validation par Entrée.

Pour les réglages de volume :

- Les valeurs sont des entiers compris entre 0 et 128 (valeurs utilisées par `SDL_mixer`).
- Le volume est modifié dynamiquement à l'aide de la fonction `Mix_Volume` pour les effets sonores, et `Mix_VolumeMusic` pour la musique.
- Des indicateurs visuels (barres de progression) permettent de voir le niveau actuel.

Pour le changement d'affichage, la fonction `SDL_SetWindowFullscreen` est utilisée pour basculer en plein écran ou revenir en mode fenêtré. Cette action est immédiate et ne nécessite pas de redémarrer le jeu.

Sauvegarde des paramètres Les paramètres modifiés par l'utilisateur sont sauvegardés dans un fichier de configuration simple (fichier texte ou fichier binaire). Ce fichier est lu au lancement du jeu pour restaurer les préférences précédentes.

Ergonomie Une attention particulière a été apportée à l'ergonomie : les modifications sont visibles en temps réel, et un bouton **Retour** permet de quitter le menu à tout moment sans risque de perte de données.

4.5 Objets de carte

Les objets de carte désignent l'ensemble des éléments interactifs présents dans les niveaux du jeu. Il s'agit notamment des plateformes mobiles, des pièges, des bonus, des checkpoints ou encore des portes de fin de niveau.

Chargement depuis une carte Chaque niveau est représenté par un fichier de carte (souvent un fichier texte) où chaque caractère ou chiffre correspond à un type d'objet ou de tuile. Lors du chargement d'un niveau :

- Le fichier est lu ligne par ligne.
- Un tableau 2D est rempli avec les types d'objets à chaque position.
- Les coordonnées x/y de chaque objet sont calculées à partir de leur position dans la grille.

Types d'objets gérés Les principaux objets de carte incluent :

- **Bonus** : cœurs de vie, pièces, items spéciaux.
- **Pièges** : pics, projectiles, zones de dégâts.
- **Checkpoints** : permettent de réapparaître en cas de mort.

- **Plateformes mobiles** : se déplacent sur un axe défini, à vitesse constante.
- **Portes de fin de niveau** : déclenchent la transition vers le niveau suivant.

Chaque objet possède une structure spécifique contenant :

- Sa position (x, y)
- Son type
- Sa texture (image à afficher)
- Son état (actif, collecté, mortel, etc.)

Affichage Lors du rendu du jeu, les objets sont dessinés après les tuiles du décor mais avant le personnage. Cela permet de garantir qu'ils soient visibles et correctement positionnés dans la scène.

Détection de collisions Les objets interagissent avec le joueur via un système de détection de collisions :

- Si le joueur entre en collision avec un bonus : l'objet est collecté et retiré de la carte.
- En cas de collision avec un piège : des dégâts sont infligés, ou une réapparition est déclenchée.
- Les plateformes mobiles sont intégrées dans la logique de gravité pour permettre au joueur de se déplacer dessus.

Extension possible La structure utilisée permet d'ajouter facilement de nouveaux objets en définissant leur comportement dans une fonction dédiée. Ainsi, le moteur de jeu reste évolutif sans nécessiter une refonte complète.

4.6 Personnage

Le personnage principal est l'élément central du jeu. Il incarne le joueur dans l'univers du jeu et interagit avec l'environnement, les objets et les ennemis.

Apparence et animation Le personnage est représenté par un ensemble d'images extraites d'un GIF animé. Chaque animation (marche, saut, repos, mort) est composée de plusieurs frames stockées sous forme de textures SDL dans un tableau. Ces textures sont chargées au lancement du jeu, puis utilisées dynamiquement selon l'état du personnage.

Déplacement Le personnage peut se déplacer horizontalement (gauche/droite), sauter et retomber. Les mouvements sont gérés via les événements clavier (SDL_Event) :

- Flèche gauche / droite : déplacement horizontal
- Barre espace : saut

La position du personnage est mise à jour à chaque frame en tenant compte de la vitesse et de la gravité. Un système de collision empêche le joueur de traverser les murs, le sol ou les obstacles.

Système de gravité Une constante de gravité est appliquée au personnage pour simuler une chute naturelle. Le moteur de collision permet de détecter le sol ou les plateformes afin de stopper la chute et permettre un nouveau saut.

Animations dynamiques Le rendu du personnage change automatiquement selon son état :

- En mouvement : animation de marche
- En saut : animation de saut
- À l'arrêt : animation idle (repos)
- Lors de la mort : animation spécifique déclenchée une seule fois

Un compteur d'images et un timer permettent de gérer le passage fluide d'une frame à l'autre dans les animations.

Gestion des points de vie Le personnage dispose d'un nombre limité de points de vie. Lorsqu'il est touché par un piège ou un ennemi :

- Un son de blessure est joué
- Un effet visuel (clignotement, recul) est déclenché
- La barre de vie est mise à jour

En cas de perte de tous les points de vie, le joueur réapparaît au dernier checkpoint atteint, ou au début du niveau.

Évolutivité La structure du personnage est conçue pour être facilement étendue. Il serait possible d'ajouter des capacités spéciales (double saut, dash, attaque, etc.) ou même différents skins selon l'évolution du jeu.

5 Crédits de fin

Mes remerciments aux contributeurs : Madame Ben Zid Maha, Madame Piau Toffolon Claudine et Madame Py Domique pour ce projet.

6 Easter egg

Présentation de l'easter egg caché dans le jeuDans le but de surprendre les joueurs les plus curieux, un *easter egg* a été intégré dans le jeu. Il s'agit d'un contenu caché qui ne fait pas partie du gameplay principal, mais qui récompense l'exploration et la découverte.

Déclenchement L'easter egg peut être activé de deux façons :

- En trouvant un code secret dissimulé dans l'un des niveaux du jeu (par exemple sur un mur ou dans une pièce cachée).
- En saisissant ce code dans le menu principal via le clavier. Aucune indication visuelle n'est donnée à l'écran, il faut donc connaître le code ou tomber dessus par hasard.

Une fois le bon code entré, une transition spéciale s'active, et le jeu bascule vers un mini-jeu caché, inspiré de l'ambiance et du gameplay du jeu *Five Nights at Freddy's*.

Le mini-jeu caché Ce mini-jeu se déroule dans une pièce sombre avec plusieurs caméras de surveillance. Le joueur doit surveiller différentes zones tout en gérant des ressources (comme la batterie) et en réagissant rapidement à des événements inattendus.

Les principales mécaniques sont :

- Changement de caméra via des touches dédiées.
- Apparition soudaine d'un personnage hostile à certaines heures (générées aléatoirement).
- Bruitages spécifiques et ambiance sonore oppressante.

Le but est de survivre un certain temps sans se faire attraper. En cas d'échec, un*jumpscare*(une attaque terrifiante on l'ennemienous saute dessus) animée est affichée avant de revenir au menu principal.

Intérêt Cet easter egg permet d'enrichir l'expérience globale du jeu tout en montrant une autre facette du développement. Il offre aussi une touche de lore et de nostalgie pour les joueurs adeptes des jeux d'horreur ou des secrets bien cachés.

Ce type de contenu bonus met en avant la créativité de l'équipe de développement, tout en récompensant les joueurs attentifs.

7 Bilan et résultats

Le développement du projet Tales Of Depression a permis de mettre en avant de nombreuses compétences techniques et organisationnelles. Cependant, plusieurs aspects du développement auraient pu être améliorés, notamment en ce qui concerne les choix technologiques et les outils utilisés.

Choix technologiques et amélioration avec C# L'utilisation du langage C et de la bibliothèque SDL, bien que robuste et largement utilisée, a engendré certains défis, notamment en termes de gestion de la mémoire, de complexité du code et de débogage. L'une des difficultés majeures résidait dans la gestion des textures et des animations, qui est plus verbeuse en C qu'en d'autres langages modernes.

Une solution pour améliorer l'efficacité du développement aurait été d'opter pour un langage de programmation plus moderne et abstrait, comme **C#**. Ce langage, en plus de sa simplicité relative par rapport au C, permet une approche orientée objet qui facilite la gestion des objets du jeu (comme le personnage, les ennemis, les niveaux). L'utilisation de **MonoGame**, un framework open-source compatible multiplateforme, aurait également permis de réduire la complexité du code en offrant des outils de gestion graphique, de son et d'entrées plus faciles à manipuler.

Avantages de C# et MonoGame MonoGame et C# offrent plusieurs avantages pour un projet comme celui-ci :

- **Facilité de gestion des ressources** : MonoGame dispose d'une gestion intégrée des textures, des sons et des animations, permettant une gestion plus rapide et plus fiable des ressources.
- **Abstraction et Object-Oriented Programming (OOP)** : C# étant un langage orienté objet, il est plus facile de structurer le code en utilisant des classes et des objets, ce qui rend le code plus modulaire et réutilisable.
- **Interface graphique (Windows Forms)** : Pour des applications nécessitant une interface utilisateur, Windows Forms, avec son environnement de développement intégré (Visual Studio), aurait permis de créer des menus et des fenêtres de manière beaucoup plus rapide et intuitive.
- **Support de la communauté et des outils** : La communauté MonoGame est très active et propose une vaste documentation, ainsi que des outils pour le débogage et la gestion des ressources multimédia.

Autres améliorations envisageables En plus du passage à C# et MonoGame, plusieurs autres aspects auraient pu être améliorés :

- **Tests et débogage** : Le développement d'un système de tests automatisés, comme les tests unitaires, aurait permis d'identifier les bugs plus tôt et de stabiliser l'ensemble du projet.

- **Gestion des versions et travail en équipe** : L'utilisation plus systématique d'outils de gestion de version comme Git aurait facilité la gestion des différentes branches et la coordination au sein de l'équipe.
- **Amélioration de la documentation** : La création d'une documentation plus complète et détaillée pour le code et la gestion du projet aurait facilité le travail de développement et permis une meilleure compréhension du projet.

Conclusion En conclusion, bien que le choix du langage et des outils ait imposé certaines contraintes, le projet a permis de développer une expérience intéressante, riche en enseignements. Envisager l'utilisation de technologies plus modernes, comme C# avec MonoGame, aurait pu accélérer le développement et permettre une meilleure gestion de la complexité du projet, notamment pour la création de ressources et l'intégration de nouvelles fonctionnalités.

8 Annexes