FIVE NIGHTS AT CHÂTELET

- PROMO 2030 -

NEEV CHANDIRAMANI - GAUTHIER ROLAND - MAXIME YE - MAXIME GONTRAN

HTTPS://FIVENIGHTSATCHATELET.NEEVCHANDIRAMANI.COM/

RAPPORT DE SOUTENANCE 2

EPITA
ÉCOLE D'INGÉNIEURS EN INFORMATIQUE

TABLE DES MATIÈRES :

I. Introduction

Five Nights At Châtelet est un jeu d'horreur et de survie inspiré de la célèbre franchise Five Nights at Freddy's.
Ce projet s'inscrit dans un cadre strictement scolaire et académique. Il a pour objectif la conception et la réalisation d'un jeu vidéo en équipe, permettant de mobiliser des compétences en conception fonctionnelle, organisation de projet et développement logiciel.
Le jeu Five Nights at Châtelet est un jeu d'horreur et de survie multijoueur en vue à la première personne, destiné à des joueurs occasionnels appréciant les ambiances angoissantes et les mécaniques simples mais immersives.

Ce rapport présente l'avancement du projet depuis la première soutenance de janvier 2026. Il détaille les fonctionnalités implémentées, les choix techniques réalisés, les difficultés rencontrées et les perspectives pour la soutenance finale.

Depuis la première soutenance, notre équipe a réalisé des progrès majeurs sur tous les fronts : le multijoueur en réseau est désormais pleinement fonctionnel sur un serveur dédié, les mécaniques de gameplay ont été profondément enrichies, et le jeu est disponible sous forme d'exécutables compilés pour Windows et Linux.

Note : notre équipe est passée de 5 à 4 membres en cours d'année à la suite au départ de Maximilien Cochet. Ses tâches ont été redistribuées entre les membres restants.

II. Concept et déroulement du jeu

Five Nights at Châtelet repose sur un gameplay **multijoueur asymétrique** en vue à la première personne. Les joueurs sont automatiquement répartis en deux camps aux objectifs opposés : les **Explorateurs**, qui doivent coopérer pour résoudre une énigme et s'échapper, et les **Infectés**, dont le but est de traquer les survivants.

Une partie s'articule autour des étapes suivantes :

- **Initialisation :** Création d'une session en ligne et répartition automatique des rôles.

- **Phase active :** Exploration et résolution d'énigmes dans une atmosphère oppressante sous contrainte de temps.

- **Dénouement** : La partie prend fin par l'évasion des explorateurs, l'infection totale du groupe, ou l'écoulement du temps imparti.

III. Organisation du groupe

Afin d'assurer une collaboration fluide et une progression constante du jeu ***Five Nights At Châtelet***, notre studio ***BambouX*** a mis en place une organisation basée sur la communication constante et la répartition des tâches.

Gestion de la communication : La quasi intégralité des messages échangés sont échangés sur un canal de discussion sur la plateforme ***Discord***.
Cette méthode nous permet de centraliser les informations afin que tous les membres du groupe soient à jour. Ce choix nous permet également de réaliser des sessions de travail régulières.
Cet outil est indispensable pour notre studio, il nous permet de résoudre rapidement les blocages techniques et de maintenir une cohésion d'équipe indispensable pour un projet de cette envergure.

Gestion du code source et versionnage : Pour le développement logiciel, nous utilisons GitHub comme plateforme de forge logicielle. L'usage de Git est essentiel pour centraliser notre code source Python, gérer les versions et permettre un travail collaboratif en parallèle. Un système de CI/CD via GitHub Actions génère automatiquement des exécutables à chaque nouvelle version taguée, conformément aux exigences du projet SAE J3D.

IV. Répartition des tâches

La conception de *Five Nights At Châtelet* a été découpée en plusieurs pôles techniques afin de paralléliser le travail et de garantir que chaque membre apporte sa pierre à l'édifice. Suite au départ de Maximilien Cochet en cours d'année, nous avons redéfini pour chaque tâche un responsable principal (R) et un secondant (S) pour assurer la continuité du développement.

Pôle Graphisme et Environnement :

La cohérence visuelle est portée par **Gauthier Roland**, responsable du *Map design* et du *Character design*, soutenu par **Neev Chandiramani**. Ce pôle s'occupe de créer l'atmosphère oppressive du jeu à travers l'architecture des niveaux et l'apparence des entités.

Pôle Technique et Gameplay :

Ce pôle gère le moteur interne du jeu sous Ursina :

- **Déplacements, physique et combat : Maxime Gontran** dirige la gestion des mouvements, de la physique, du système de HP et du système d'attaque, avec l'appui de **Maxime Ye**.
- **Menu et interactions : Maxime Ye** est responsable du menu Options, du rebinding des touches et de la gestion des équipes, secondé par **Maxime Gontran**.

Pôle Multijoueur et Communication :

- **Réseau :** Élément obligatoire du projet, le multijoueur est piloté par **Neev Chandiramani** avec l'aide de **Maxime Gontran**.
- **Site Web :** Le site internet, vitrine de notre projet, est géré par **Neev Chandiramani** et **Gauthier Roland**.
- **Build / Déploiement :** La compilation et la distribution des exécutables sont assurées par **Neev Chandiramani**, secondé par **Gauthier Roland**.

Pôle Audio :

L'immersion sonore, cruciale pour l'horreur, est confiée à **Maxime Ye**, responsable des sons, avec le soutien de **Maxime Gontran**.

Tâches	Login1	Login2
Map design / Character design	(R) gauthier.roland@epita.fr	(S) neev.chandiramani@epita.fr
Multijoueur / Réseau	(R) neev.chandiramani@epita.fr	(S) maxime.gontran@epita.fr
Sons / Audio	(R) maxime.ye@epita.fr	(S) maxime.gontran@epita.fr
Déplacement / Physique	(R) maxime.gontran@epita.fr	(S) maxime.ye@epita.fr
Combat / HP	(R) maxime.gontran@epita.fr	(S) maxime.ye@epita.fr
Menu / Options	(R) maxime.ye@epita.fr	(S) maxime.gontran@epita.fr
Site Web	(R) neev.chandiramani@epita.fr	(S) gauthier.roland@epita.fr
UI / HUD	(R) maxime.gontran@epita.fr	(S) maxime.ye@epita.fr
Build / Déploiement	(R) neev.chandiramani@epita.fr	(S) gauthier.roland@epita.fr

nouvelle répartition des tâches à ce jour

V. Avancement général

Le tableau suivant résume l'état actuel de chaque fonctionnalité principale du jeu :

Fonctionnalité	État
Menu principal (effets CRT, glitch, volume)	Terminé
Menu Options avec rebinding des touches	Terminé
Correction AZERTY/QWERTY	Terminé
Déplacement joueur (WASD + souris)	Terminé
Physique : gravité, saut, collisions multi-points	Terminé
Stamina et sprint	Terminé
Système HP, dégâts, invincibilité, respawn	Terminé
Attaque directionnelle (hitbox vectorielle)	Terminé
Modèles 3D personnages (Blender)	Terminé
Map 3D centre commercial (Blender)	Terminé
Screamers vidéo (cv2 + réseau)	Terminé
Multijoueur réseau TCP sur VPS dédié	Terminé
Synchronisation position + rotation	Terminé
Communication de combat réseau	Terminé
Exécutables Windows / Linux / macOS	Terminé
Site web avec téléchargement	Terminé
Sons d'ambiance in-game	En cours
Intelligence artificielle (infecté)	À faire
Système d'énigmes	À faire
Attribution automatique des rôles	À faire

VI. Site web

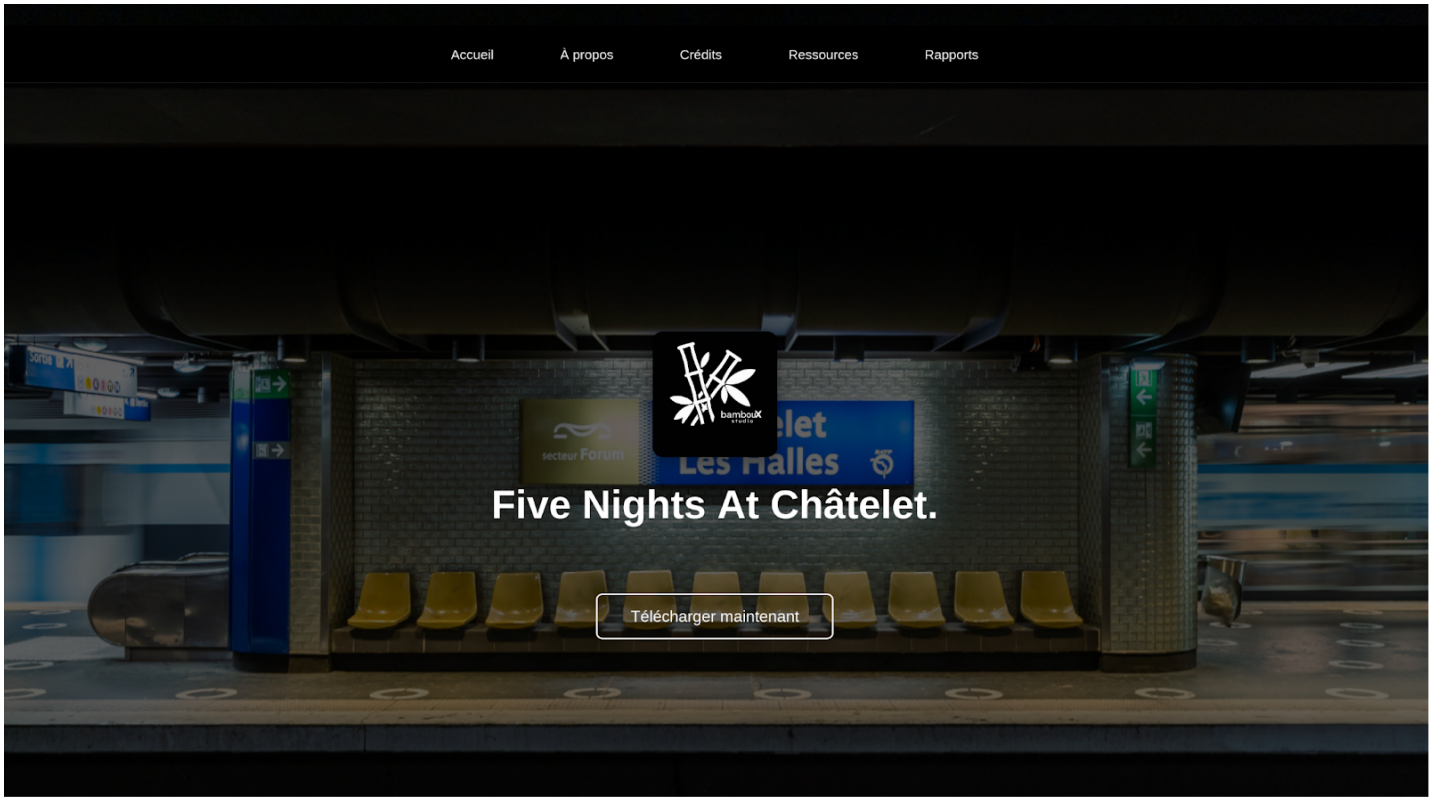
Conformément aux exigences du cahier des charges, nous avons mis à jour notre site internet qui fait office de vitrine et de centre de ressources pour Five Nights At Châtelet. Ce support de communication, mis à jour à chaque étape du projet, remplit plusieurs fonctions essentielles :

- **Présentation :** Le site propose une description détaillée du concept de jeu ainsi qu'une présentation de notre équipe et de ses membres, avec leurs rôles mis à jour suite au départ de Maximilien Cochet.

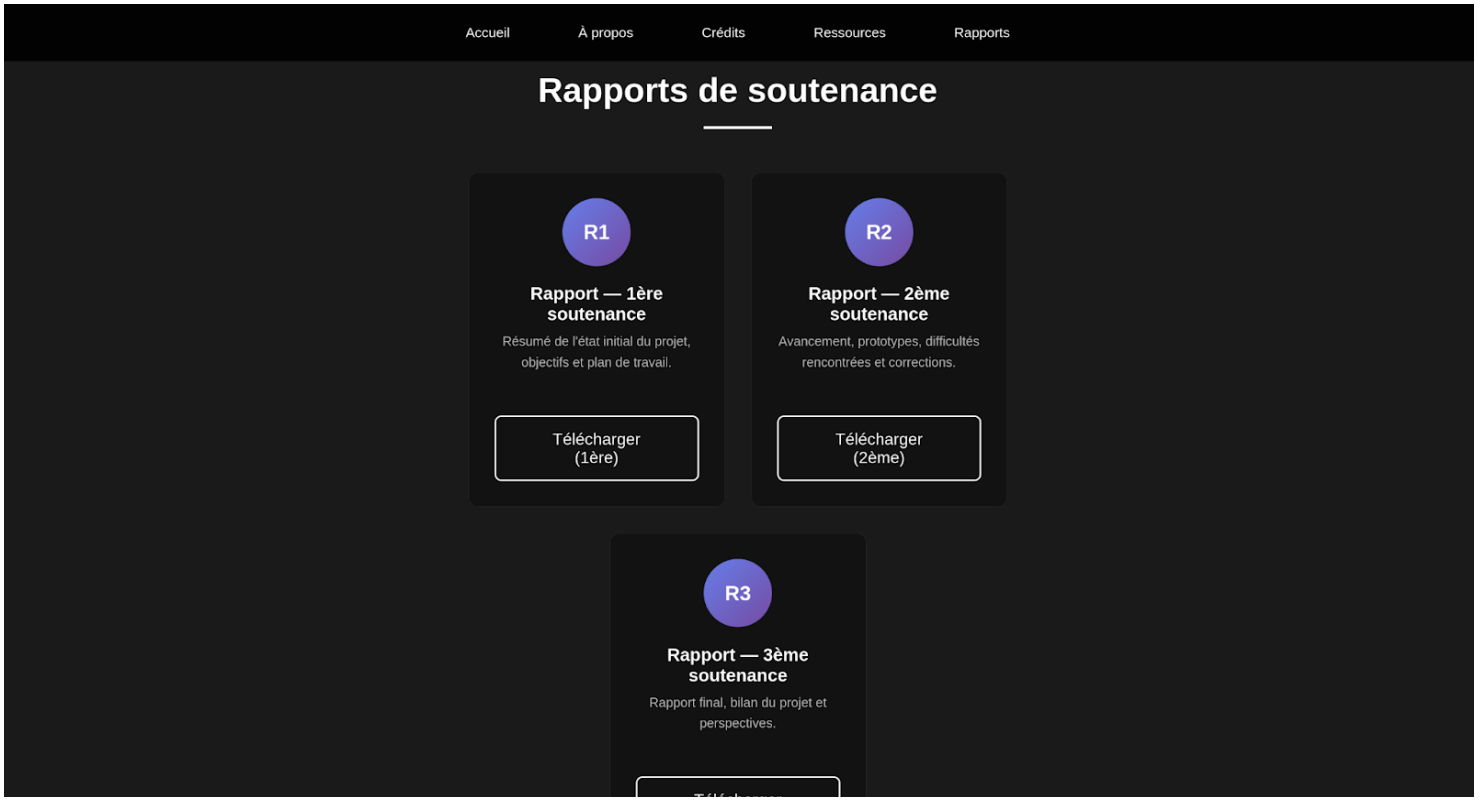
- **Centre de documentation :** Une section dédiée regroupe l'ensemble des ressources externes mobilisées, incluant les librairies logicielles utilisées (Ursina Engine, PyGame, Blender, GitHub Actions) ainsi que les banques d'assets sonores (Freesound.org).

- **Plateforme de téléchargement :** Le site offre un accès direct aux exécutables compilés pour Windows, Linux et macOS, ainsi qu'aux rapports de soutenance.

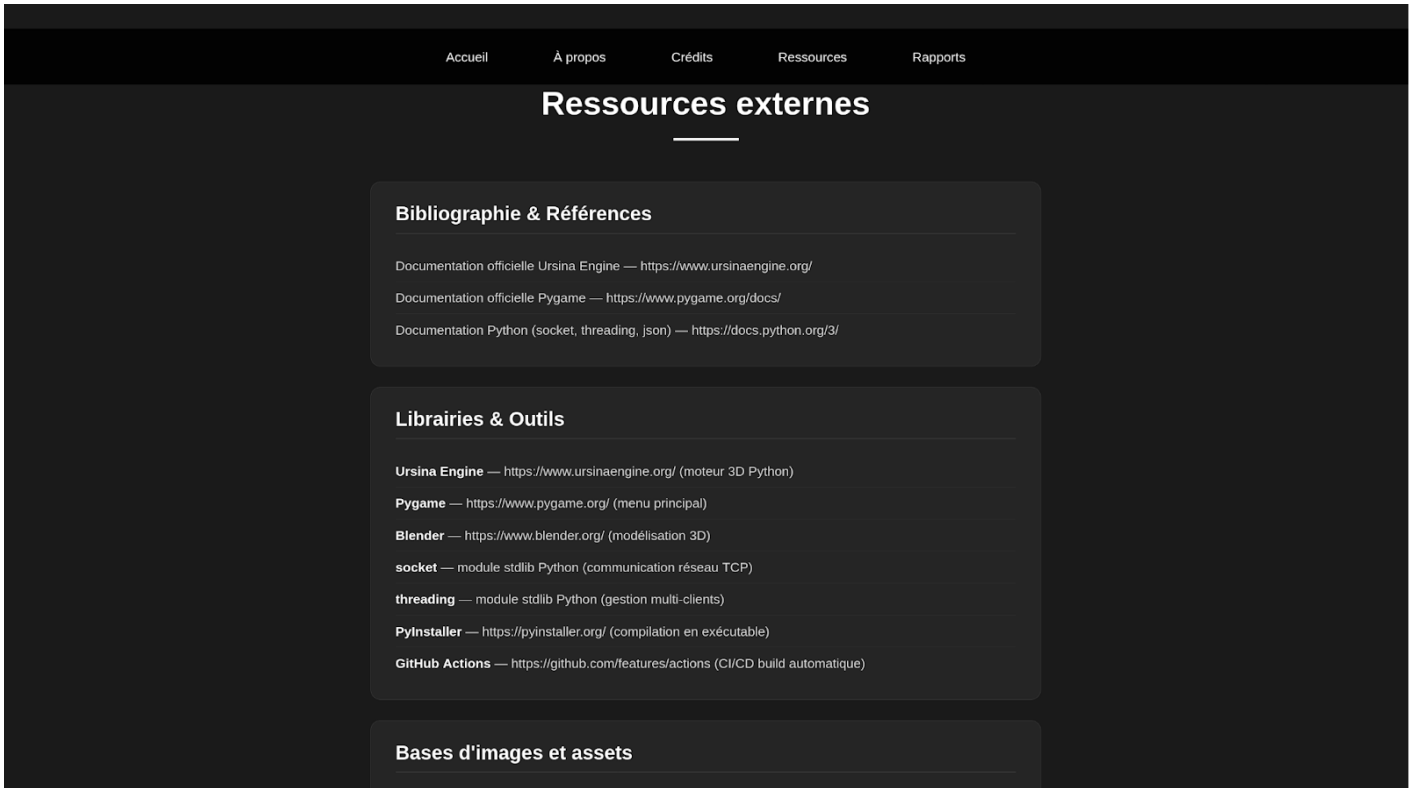
Le site web est accessible à l'adresse : https://fivenightsatchatelet.neevchandiramani.com/



page d'accueil du site web



page de téléchargement des rapports du site web



page des ressources du jeu

VII. Modèles 3D

Dans le projet, nous nous occupons de tout l'**aspect graphique** du jeu : la création de la map, des personnages, des animations, mais aussi de leur **implémentation directement en Python**.

Pour l'instant, nous avons principalement travaillé sur les **personnages**. Nous en avons créé deux, qui représentent les deux équipes du jeu : un **explorateur** et un **infecté**. Le jeu a une ambiance volontairement horrifique, qui se déroule dans un centre commercial. Pourtant, nous avons fait le choix d'un **style graphique cartoon**, avec des grosses têtes et de grands yeux un peu amusants. L'objectif était de créer un **contraste** entre le visuel des personnages et l'ambiance oppressante du jeu, pour rendre l'ensemble plus marquant et original.

modèle 3D d'un "explorateur"

modèle 3D d'un "infecté"

Ces modèles ne sont pas seulement statiques : ils s'inscrivent dans une logique d'animation. Un travail a été réalisé en amont dans Blender pour créer des animations, ce qui implique l'utilisation d'une armature et d'un rig permettant de déformer le mesh. Cela montre que le personnage a été préparé pour être animé, avec une structure adaptée aux mouvements (marche, course, etc.). Le fait d'avoir déjà conçu ces animations permet d'avoir un personnage plus réaliste et dynamique. Même si elles ne sont pas encore exploitées directement dans le moteur. Le personnage est donc prêt à évoluer d'un simple modèle affiché à une entité animée capable de réagir aux actions du joueur.

Concernant la map :

À ce stade du projet, la base du jeu est déjà solidement mise en place. Une map complète a été intégrée dans le moteur à partir d'un modèle 3D importé depuis Blender, ce qui permet de disposer d'un environnement jouable et cohérent.

Cependant il faut que la taille et la complexité de la map restent raisonnables, pour permettre une exécution fluide et adaptée à un jeu. C'est l'une des raisons pour lesquelles elle reste assez simple. La map a été conçue et importée depuis Blender sous forme d'un modèle complet, permettant de structurer un véritable environnement de jeu.

Ce modèle, représentant un espace de type centre commercial. Le fait d'avoir un mesh relativement optimisé, avec un nombre de faces raisonnable, montre que le modèle a été pensé pour une utilisation en temps réel, ce qui est essentiel dans un contexte de jeu. Au-delà de la simple importation, la map n'est pas seulement décorative : elle sert directement de support à l'expérience de jeu. Les éléments qui la composent — murs, sols, structures — définissent l'espace, les déplacements possibles et la perception du joueur. Le modèle est donc déjà utilisé comme une véritable scène jouable, et pas simplement comme un prototype visuel.

modèle 3D de la map

Perspectives et objectifs

Pour la suite, nous prévoyons d'affiner la map en ajoutant des **détails visuels** et des **éléments interactifs**, de continuer les **animations des personnages**, et de créer d'autres modèles si besoin. L'objectif est d'avoir un environnement et des personnages **cohérents visuellement et techniquement**.

VIII. Multijoueur en réseau

Le multijoueur constitue l'évolution majeure depuis la première soutenance. À cette époque, l'aspect réseau n'était pas fonctionnel et nous testions uniquement sur des prototypes simples. Aujourd'hui, il est pleinement opérationnel avec un serveur dédié hébergé à domicile.

Architecture client-serveur

Le projet repose sur une architecture **client-serveur TCP**. Un serveur dédié, hébergé sur un serveur Debian Linux à domicile, est chargé de la gestion des connexions et de la synchronisation de l'état du jeu entre les différents clients. Il fonctionne de manière indépendante du moteur graphique.

Le serveur est accessible via une adresse DynDNS sur le port 5555, ouvert via une règle NAT. Il utilise des sockets TCP pour recevoir les données envoyées par les clients et leur renvoyer l'état global du jeu.

Protocole d'échange

Chaque message échangé est un objet JSON suivi d'un saut de ligne. Deux types de messages circulent sur le réseau :

- **Position :** {"x": 15.2, "y": 3.0, "z": 0.5, "ry": 90} — envoyé par chaque client toutes les 50ms.
- **Événements :** dégâts de combat et déclenchement de screamers — envoyés ponctuellement lors des interactions.

Client réseau

Côté client, un thread de réception tourne en arrière-plan pour ne pas bloquer le rendu du jeu. Un timeout de connexion de 3 secondes permet au jeu de démarrer en mode solo si le serveur n'est pas joignable, garantissant une expérience utilisateur satisfaisante même en l'absence de réseau.

Chaque client transmet au serveur les informations relatives à son joueur (position et rotation). Le serveur centralise ces données, les maintient à jour pour l'ensemble des joueurs connectés, puis les redistribue afin d'assurer la synchronisation des entités visibles dans chaque instance du jeu.

Cette séparation stricte entre logique réseau et rendu graphique améliore la modularité du projet, limite les dépendances et facilite les évolutions futures. En effet, dès que nous changeons quelque chose dans le jeu, nous n’avons pas besoin de toucher au serveur, le multijoueur est donc tout le temps fonctionnel. Attention cependant, tous les joueurs doivent être sur la même version du jeu afin de garantir la meilleure expérience.

IX. Menu et sons

1. Amélioration du menu : onglet Options

Pour cette deuxième étape, nous avons transformé le menu pour qu'il ne soit plus seulement esthétique, mais vraiment fonctionnel.

- Système de navigation : nous avons modifié le code pour permettre de passer de l'écran d'accueil aux options sans aucun bug de transition.
- Changement des touches (Rebinding) : nous avons ajouté une fonctionnalité qui permet au joueur de choisir ses propres touches pour se déplacer, sauter et interagir.
- Sécurité anti-doublons : Le menu empêche désormais d'utiliser la même touche pour deux actions différentes, évitant de bloquer le personnage en jeu.
- Lien avec le jeu (Fichier JSON) : Quand le joueur clique sur Start, ses préférences de touches sont sauvegardées dans un fichier config_touches.json chargé au démarrage du jeu 3D.
- Correction AZERTY / QWERTY : Une table de correspondance assure la compatibilité entre les noms de touches PyGame et Ursina pour les claviers français.

image du menu option

2. Travail sur le son

Le son est l'élément le plus important pour créer l'immersion dans un jeu d'horreur.

- Sons d'ambiance : Recherche et intégration de musiques de fond et sons ambiants (échos, grincements) qui renforcent l'atmosphère oppressante du jeu.
- Sons d'action : Préparation des sons de bruits de pas, cris des infectés, ouverture des portes, et sons interactifs comme une annonce radio sur une invasion de zombies.
- Son d'ambiance gare : Intégration d'un son se déclenchant aléatoirement toutes les 120 à 240 secondes in-game pour renforcer l'immersion.

X. Combat, physique et interface utilisateur

1. Système de combat et de santé

- **Gestion des HP :** Implémentation d'une jauge de santé (MAX_HP = 100) avec affichage dynamique changeant de couleur (vert, orange, rouge) selon le niveau.

- **Mécanique de dégâts :** La fonction receive_damage inclut une période d'invincibilité post-coup de 0.5 seconde pour éviter les morts instantanées, et un flash rouge à l'écran pour signaler le coup reçu.
- **Mort et respawn :** À zéro HP, le joueur passe en état is_dead (mouvements bloqués). Un timer de 3 secondes précède le respawn automatique aux coordonnées de départ, avec une brève invincibilité post-respawn.
- **Attaque directionnelle :** La fonction do_attack utilise le produit scalaire (dot product) pour détecter si un adversaire est bien situé dans le cône d'attaque (portée 3 unités, largeur 2, hauteur 3).

2. Physique et mobilité

- **Refonte du saut :** Passage à une physique basée sur la vélocité verticale et la gravité (-40). Un raycast assure une détection du sol ultra-précise avec un snap fluide pour éviter les rebonds.
- **Collisions multi-points :** Le joueur utilise désormais 6 raycasts simultanés (centre, gauche, droite à deux hauteurs) pour tester les collisions. Cela permet de glisser contre les murs sans rester bloqué dans les angles.
- **Colliders invisibles :** Ajout de murs et rambardes invisibles pour fluidifier la navigation dans les zones complexes de la map.

3. Réseau et synchronisation

- **Synchronisation position et rotation :** Les autres joueurs affichent désormais leur rotation Y en plus de leur position, rendant leurs déplacements plus naturels.
- **Communication de combat :** Chaque coup porté est envoyé au serveur via network.send_damage(). Le client écoute les événements de dégâts pour appliquer les baisses de HP sur la cible concernée.

- **Feedback visuel réseau :** Lorsqu'un joueur est touché, son modèle 3D clignote en blanc pour tous les utilisateurs connectés, confirmant visuellement le hit.

L’interface utilisateur :

Le HUD (Head-Up Display) affiche en temps réel les informations essentielles pour le joueur :

- Stamina : Indicateur textuel du niveau d'endurance restant.
- HP : Compteur de points de vie coloré (vert / orange / rouge).
- Indicateur d'attaque : Feedback visuel du cooldown de l'attaque au coin bas-droit de l'écran.
- État réseau : Indicateur de connexion au serveur avec le nombre de joueurs connectés (vert = connecté, rouge = déconnecté).
- Flash de dégâts : Overlay rouge semi-transparent lors de la réception d'un coup.

XI. Build et déploiement

Une infrastructure de build automatisée a été mise en place via GitHub Actions pour générer des exécutables compilés à chaque nouvelle version taguée du projet. Le workflow déclenche trois jobs en parallèle (Windows, Linux, macOS) qui installent les dépendances, compilent le jeu avec PyInstaller, et publient les exécutables dans une GitHub Release téléchargeable.

Exécutables générés :

Fichier	Plateforme
FiveNightsAtChatelet-Windows.exe	Windows 10/11
FiveNightsAtChatelet-Linux	Linux (x86_64)
FiveNightsAtChatelet-macOS	macOS (x86_64)

Problème résolu : assets PyInstaller

PyInstaller extrait les fichiers dans un dossier temporaire (_MEIPASS) au lancement. Ursina cherchait ses assets depuis le répertoire de lancement de l'exécutable, et non depuis _MEIPASS. La solution finale combine os.chdir(BASE_DIR) et application.asset_folder = Path(BASE_DIR) pour Ursina, et des chemins absolus via une fonction res() pour PyGame et

OpenCV, qui ne respectent pas le changement de répertoire courant.

XII. Difficultés rencontrées

Départ d'un membre

Le départ de Maximilien Cochet en cours d'année a représenté un défi organisationnel. Ses tâches ont été redistribuées entre les membres restants, notamment Maxime Gontran qui a absorbé une grande partie du travail de gameplay.

Compatibilité Python 3.14 / Ursina

La version Python 3.14, installée par défaut sur Fedora 43, cause des segmentation faults avec Panda3D, le moteur sous-jacent d'Ursina. La solution a été de créer un environnement virtuel Python 3.11 pour le développement et les tests.

Build macOS : incompatibilité d'architecture

Les runners GitHub Actions macOS utilisent des machines Apple Silicon (arm64), incompatibles avec certaines librairies Panda3D compilées en x86_64. La solution a été d'exclure le module panda3d.rocket du build via l'option --exclude-module, ce module n'étant pas utilisé dans le jeu.

Chemins d'assets dans les exécutables

Le chargement des assets dans les exécutables PyInstaller a nécessité plusieurs itérations avant d'aboutir à une solution stable, car Ursina, PyGame et OpenCV gèrent les chemins de fichiers de manières différentes.

XIII. Conclusion

Le projet *Five Nights at Châtelet* représente une aventure collaborative ambitieuse menée par notre équipe BambouX, composée de Neev Chandiramani, Gauthier Roland, Maxime Ye et Maxime Gontran. Chacun a apporté ses compétences techniques, sa créativité et son investissement personnel afin de poser les bases d'un jeu d'horreur multijoueur immersif, encore en cours de développement mais déjà fonctionnel sur de nombreux aspects.

Les bases posées au cours de cette deuxième phase nous permettent d'envisager la suite avec une vision claire : le multijoueur est fonctionnel, les mécaniques de gameplay sont riches, et le jeu est distribuable sur trois plateformes.

Five Nights at Châtelet n'en est encore qu'à une étape intermédiaire de son développement. Pour la soutenance finale, nos objectifs sont les suivants :

- Intelligence artificielle : Implémentation d'un infecté capable de détecter et poursuivre les joueurs de manière autonome.
- Système d'énigmes : Conception et intégration des puzzles pour les explorateurs.
- Attribution automatique des rôles : Système de lobby permettant la répartition automatique en début de partie.
- Polish graphique et sonore : Textures, animations de personnages et effets sonores complets.
- Rapport final : Rédaction du rapport de projet (50 pages minimum) et du dossier d'exploitation.

Five Nights at Châtelet entre désormais dans une phase de finalisation. Les fondations solides posées au cours de cette deuxième étape multijoueur fonctionnel, mécaniques de gameplay complètes, exécutables distribuables, nous permettent d'aborder la soutenance finale avec confiance, en concentrant nos efforts sur ce qui reste à implémenter : l'intelligence artificielle, les énigmes et le polish global du jeu.
