Groupe : **Année :2024/2025**

Mathis WARCHE

Lucas PAVONE

Benjamin HUON

Projet : Waremon

Date de soumission du projet : **13-14/12/2024**

Cours : **Développement et applications / Professeur : Mr Tomczak**

Sommaire

Compte-Rendu Globale :

Description Générale du Projet :

Nous voulons faire un jeu en 2 dimensions s'apparentant à un ancien jeu Pokémon donc un jeu de type RPG(Role Playing Game), tout cela dans l'univers de la cybersécurité avec par exemple le fait que nous incarnerons un hacker qui devra capturer des Waremon représentant des virus plus ou moins connus du réel, pour déjouer les plans des Anonymous, groupe antagoniste jouant le rôle de la « Team Rocket ».Le jeu se lancera et ne stockera pas de sauvegarde ce qui veut dire que la partie durera jusqu'à l'arrêt du jeu.

Synopsis du jeu :

Dans le monde de Waremon, les joueurs incarnent des Hackers dont le but est de capturer, entraîner et combattre avec des créatures appelées Waremon. En explorant un monde rempli de paysages variés, les joueurs rencontrent et capturent différentes espèces de Waremon, chacune ayant ses propres capacités et caractéristiques. Ils forment une équipe pour affronter d'autres hackers, participer à des combats stratégiques. Sur leur chemin, ils devront aussi déjouer les plans des Anonymous, une organisation criminelle qui utilise les Waremon à des fins malveillantes.

1. Présentation :

1.1- Présentation du Contexte :

Nous devons faire un projet dans le cadre d'un module appelé "Développement Applications", ce projet consiste à développer un site, un jeu, une application, un service avec aucune restriction thématique, ni logicielle.

1.2- Objectifs et Finalités :

En termes d'objectifs, nous avons comme objectif premièrement de réaliser le défi de réaliser ce jeu, qu'il soit fini ou non. Ensuite bien évidemment, nous préférons que ce jeu soit terminé et fonctionnel.

Dans un but personnel, un des objectifs sera aussi de nous informer un peu plus sur les virus en tout genre, leur fonctionnement en général. Et donc pourquoi pas si le temps nous le permet, de faire un petit peu de sensibilisation durant les dialogues.

En finalité, nous espérons proposer une expérience de jeu qui sera retenue par l'utilisateur, de par l'innovation, de par l'univers dans lequel il jouera. Nous espérons aussi que le message de sensibilisation sera tout aussi bien retenu.

1.3 - Importance et justification du projet :

Tout d'abord, ce projet est d'une importance pour notre matière Développement Applications, et il est aussi d'une importance par rapport à l'évolution du risque d'infection informatique de nos jours.

1.4 - Définition précise de ce qui inclus et exclu du projet :

Inclus : Système de capture de virus, gestion des virus, système de combats, exploration d'un monde virtuel, une histoire prenante, une interface utilisateur soignée.

Exclus : Système de sauvegarde, type de créatures, cosmétiques.

1.5 - Limites et contraintes du projet :

Limites : Les limites sont fixées par nos capacités et le temps.

Nous nous limiterons à une application dans une fenêtre d'exécution python.

Donc aussi à une seule et unique plateforme : PC.

Contraintes : Les contraintes sont matérielles, nous disposons de notre matériel personnel uniquement, nous avons aussi une contrainte temporelle, qui est d'environ 2 mois. Et la contrainte budgétaire est de zéro car aucun budget pour une aide extérieure ne sera alloué.

2. Analyse des besoins, exigences fonctionnelles /Document de conception de jeu :

2.1 - Système de capture de virus

• Interactivité et stratégie : Le processus de capture doit être engageant, nécessitant des stratégies pour affaiblir le virus avant de tenter de le capturer. Différents types de firewalls peuvent être utilisés, chacun ayant un taux de succès variées.

• Équilibrage : Les chances de capture doivent varier selon le type de virus et le niveau de compétence du joueur, ajoutant un aspect de gestion des ressources (types de firewalls utilisés).

2.2 - Gestion des virus capturés

• Équipe de virus : Les joueurs peuvent constituer une équipe de virus actifs (par exemple, jusqu'à 3) qu'ils peuvent utiliser en combat. Les autres virus sont stockés dans un "Data Vault".

• Amélioration et évolution : Les virus peuvent gagner de l'expérience en combat et monter en niveau, débloquant de nouvelles attaques.

• Personnalisation : Les joueurs peuvent choisir leurs capacités à mesure qu'ils montent en niveau pour optimiser leur équipe.

2.3 - Combats stratégiques

• Mécanique de combat au tour par tour : Les combats sont basés sur un système au tour par tour où les joueurs et leurs adversaires sélectionnent des actions (attaquer, utiliser un objet, changer de virus).

• Attaques et types : Chaque virus possède des attaques spécifiques et est associé à un ou plusieurs types (comme "Trojan", "Spyware", "Ransomware"). Les types interagissent les uns avec les autres selon un système de forces et faiblesses.

2.4 - Exploration du monde virtuel

- Systèmes et environnements : Le jeu doit permettre aux joueurs de naviguer dans divers environnements virtuels.
- Événements aléatoires : L'apparition de virus sauvages doit se faire de manière aléatoire lorsque les joueurs explorent certaines parties de la carte.

2.5 - Gameplay et Mécanique de Jeu

-Objectif du joueur : Le but du joueur est d'attraper et collectionner le maximum de Waremon et d'y arriver en combattant les différents Boss du jeu et en découvrant la carte.

-Contrôle et interface : On pourra jouer au jeu avec un clavier pour les mouvements et la souris qui pourra nous servir pour tout ce qui est sélectionner quelque chose (exemple : sélectionner un Waremon, sélectionner nos Waremon pour notre équipe etc…).

Pour les éléments d'interfaces nous aurons un menu par exemple pour notre équipe quant aux barres de vie nous n'en aurons pas pour notre personnage mais nous aurons des barres de vie pour nos Waremon. Nous pourrons interagir avec les différents Boss pour justement lancer le combat.

-Mécanique Principales : Donc dans notre aventure nous pourrons attraper des Waremon dans le monde assez aléatoirement en les croisant mais il y aura aussi un système de combat avec un duel en tour par tour mettant en place notre équipe de Waremon face aux Waremon de notre adversaire.

Nous lancerons chacun à notre tour des attaques étant plus ou moins efficaces et la victoire du combat s'obtient en éliminant l'équipe de notre adversaire.

2.6 -Système de Récompenses et Progression

-Récompenses et progression : Les points d'expériences sont acquis lors de l'attrapage de Waremon ou lors de combat.

-Progression et niveaux : Les points d'expériences acquis nous serviront à passer nos niveaux avec une base de 100 points d'expériences requis pour passer un niveau.

2.7 - Histoire et progression

- Quêtes principales et secondaires : L'histoire principale suit le joueur qui affronte le groupe Anonymous avec à leur tête le grand frère de notre personnage Joaquim qui menace de rendre le cyberespace inutilisable et pour ça ils ont besoin du personnage que l'on va incarner (Maluben) qui a avec lui une clé USB que lui a confier ses parents qui a une importance Cornélienne qui peut soit sécurisé le cyberespace et nous sauvés où anéantir le cyberespace.

Notre rôle dans cette histoire va être de pouvoir battre le groupe Anonymous afin de sauver le cyberespace auquel cas le groupe Anonymous arrivera à ces fins.

-Relations Personnages :

Notre personnage s'appelle Maluben et à toujours rêver de devenir un grand expert en cybersécurité grâce au génie de son grand frère qui est devenu un grand agent en cybersécurité. Mais lorsque son frère à tourner le dos au monde du bien en rejoignant le groupe Anonymous pour mettre fin aux cyberespaces Maluben c'est juré de réussir à remettre son frère sur le droit chemin.

-Environnement :

L’univers de ce jeu ne diffère pas complétement du notre mais ça se passe dans 75 ans avec un monde ou l’informatique est partout avec une entité globale qui s’appelle le Cyberespace où toute chose informatique transite.

Différents lieux clés sont présent dans cet univers comme la maison familiale de notre personnage et le repère des Anonymous.

Mais en lieux plus général il y a un village, une plage, une forêt et un massif montagneux.

• Boss et défis : Les joueurs rencontrent des boss représentant des hackers connus, qui constituent des défis plus complexes.

2.8 - Esthétique et Style Visuel

Direction artistique générale : Le jeu est dans le style des anciens pokémons c’est-à-dire pixélisé en vue du dessus.

Références visuelles :

-Design des personnages et environnement :

Le personnage principal va ressembler à quelqu’un dans la vingtaine, brun avec des habits plutôt clairs de couleur unie et simple.

Le village natal de notre personnage se trouvera au Nord-Ouest de la carte, il sera sous la neige avec juste quelques maisons de couleur (bleue, verte ou rouge) il y aura aussi un lac glacé au sud des maisons.

En contrebas de ce village il y aura une petite forêt qui fera le lien avec le côté Est de la carte et qui fera la délimitation avec la mer qui se trouve tout au Sud-Ouest de la carte.

Suite à cela on pourra soit aller vers le Sud-Est ou une plage assez tranquille sous quelques palmiers se trouve ou bien aller au Nord-Est pour aller à la découverte d’une montagne accompagné de verdure.

Un pont mystérieux se trouvera au Sud de la carte qui traversera la mer du Sud-Ouest qui pourra mener à un endroit tout nouveau avec plein d’autres environnement à découvrir.

-Effets visuels et animations :

On pourrait avoir une animation lorsqu'un PNJ (personnage non joueur) décide de nous prendre en combat.

Il pourrait aussi y avoir un effet visuel lorsque les attaques sont lancées lors de combat, lors de la capture d'un Waremon.

Un effet de transition entre des interfaces que ça soit entre la carte initial avec l'interface d'un combat ou même lors de chargement d'instance comme des maisons.

2.9 -Musique et Son :

Ambiance sonore :

Le type de sons d'ambiance sera globalement de la musique avec de l'aspect techno pour rappeler le côté informatique/futuriste avec des glitchs dans les sons.

Dans la ville ainsi que dans les combats il y aura donc ce type de musique tandis que dans la forêt et à la plage on aura une musique plus posée plus tranquille.

Dans la montagne on aura plus un aspect mystérieux.

Effets sonores :

Il y aura les bruits de pas de notre personnage qui seront assez bas mais on les entendra quand même, on pourrait même avoir un système de résonnance dans une grotte ou autre…

Lors de combat il y aura un effet/bruit qui accompagnera l'attaque.

Il serait aussi sympathique d'avoir le son de vagues lorsque l'on s'approche de la mer.

Musique :

-Thèmes musicaux : Il y aura une selon la zone exemple : une musique assez techno dans la ville et les combats, des chants d'oiseau dans la forêt avec une musique assez calme comme hors de toute cette technologie, et dans la montagne il y aura moins de musique ce sera assez calme ce qui alimentera ce côté mystérieux…

-**Style musical :**

Le style musical sera plus penché vers de l'électro pour rappeler ce côté futuriste et très informatique.

2.10 -Spécifications Techniques :

-Moteur de jeu :

Nous utilisons PyGame pour coder le jeu, et donc en Python.

-Configurations requises :

Un PC au minimum. Le jeu ne sera pas assez gros et gourmand pour nécessiter une configuration particulière.

-Défis techniques et solutions :

-Délimitation de la carte et des zones accessibles pour que notre personnage ne passe pas à travers de structure.

Fonctions auto de Pygame, délimitation par pixels.

-Système de combat : on va faire une carte différente pour montrer que l'on est plus en exploration mais dans un combat, système de vie et d'attaque donc avec des PV (point de vie) et des attaques qui un font un certain nombre de dégât, on évite un système de coup critique, 2 attaques pour chaque Waremon nous trouvons que ce n'est ni peu ni trop.

-Système de sauvegarde : peut-être une belle avancée dans notre jeu mais demande beaucoup de ressource pour le faire à voir si l'on a assez d'avance pour envisager de faire ce système.

2.11 - Interface utilisateur et expérience utilisateur

• Menus intuitifs : Les menus doivent permettre une navigation facile pour gérer l'équipe de virus, accéder à l'inventaire, suivre la progression de l'histoire, et consulter les statistiques.

• Indications visuelles et retours : Des indicateurs visuels doivent signaler les interactions disponibles, les effets de statut des virus, et les objets utilisables en combat.

• Tutoriel intégré : Un didacticiel au début du jeu doit expliquer aux nouveaux joueurs les concepts clés, les contrôles, et les mécaniques de jeu.

2.12 -Cahier des charges fonctionnelles

• Exploration : Les joueurs peuvent explorer un monde numérique.

• Capture : Les joueurs capturent des virus en utilisant des firewalls. Chaque type de virus a ses propres forces et faiblesses.

• Entraînement : Les joueurs peuvent entraîner leurs virus, améliorer leurs compétences.

• Combat : Le jeu propose des combats stratégiques où les joueurs utilisent leurs virus pour affronter d'autres virus ou des boss représentant des hackers.

• Centres de soin (DataCenters) : Les joueurs peuvent soigner et améliorer leurs virus capturés dans des centres dédiés.

• Progression de l'histoire : Une histoire linéaire où le joueur doit contrecarrer les plans d'Anonymous, une organisation de hackers malveillants.

2.13 - Spécifications fonctionnelles

• Description des interactions utilisateur :

• Capture de virus : Le joueur doit interagir avec un virus pour lancer une séquence de capture utilisant des firewalls.

• Combats : Interface de combat avec options pour attaquer, utiliser des objets, changer de virus ou fuir.

• Navigation : Interface intuitive pour se déplacer à travers le monde numérique, accéder aux menus, et gérer son équipe de virus.

• Centres de soin : Interface permettant de sélectionner les virus à soigner ou à améliorer.

• Scénarios d'utilisation ou cas d'utilisation :

• Cas d'utilisation 1 : Capture d'un virus :

1. Le joueur rencontre un virus sauvage.
2. Il choisit d'utiliser un firewall pour tenter de le capturer.
3. Si la tentative est réussie, le virus est ajouté à son équipe.

• Cas d'utilisation 2 : Combat contre un boss (hacker) :

1. Le joueur accède à la zone du boss et initie un combat.
2. Il doit utiliser ses virus pour vaincre le hacker en exploitant les faiblesses des virus adverses.
3. Le joueur remporte des récompenses en cas de victoire.

2.14 -Spécifications non fonctionnelles

• Exigences liées à la performance :

Le jeu doit fonctionner sans latence perceptible sur les plateformes visées.

Les temps de chargement doivent être minimaux pour éviter de nuire à l'expérience de jeu.

• Exigences d'ergonomie :

Interface utilisateur claire, intuitive et adaptée à différents écrans.

Contrôles faciles à utiliser et cohérents avec les standards de jeu.

Contraintes techniques

• Plateformes et technologies à utiliser ou à éviter :

• Technologies utilisée :

Python avec Pygame pour une approche en 2D.

3. Cahier des charges : Sécurité – Cybersécurité

3.1-Analyse des Risques de Sécurité

-Evaluer les menaces spécifiques au type de projet (logiciel ou site web) :

Il peut y avoir un risque dans les fichiers utilisés s'ils sont corrompus ou autre.

Pareil si dans notre code on utilise des bibliothèques externes il faut vérifier si elles sont fiables et sûre avant de les implémenter dans notre code.

Risque de Reverse engineering si notre code est facile d'accès.

Identifier les données sensibles qui seront manipulées et les conséquences potentielles de leur compromission :

Fichiers du jeu : la configuration et les différentes ressources utilisés.

3.2-Conformité et Réglementations

-Examiner les lois et normes de sécurité applicables :

-RGPD pour les données personnelles :

Obligatoire dans le cas où l'on collecte des données or dans notre cas notre jeu ne collectera aucunes données.

-La norme OWASP pour les applications web :

Protection contre les différentes injections (Command injection, code injection).

3.3-Définition des Exigences de Sécurité :

Contrôles d'accès spécifiques :

Restreindre les privilèges pour empêcher les utilisateurs ou programmes non autorisés d'accéder aux fichiers internes du jeu.

3.4-Identification des Interactions Externes :

Contrôles de sécurité recommandés :

Limiter les permissions et surveiller l'utilisation des services.

3.5-Solutions Envisagés :

Pour les fichiers du jeu :

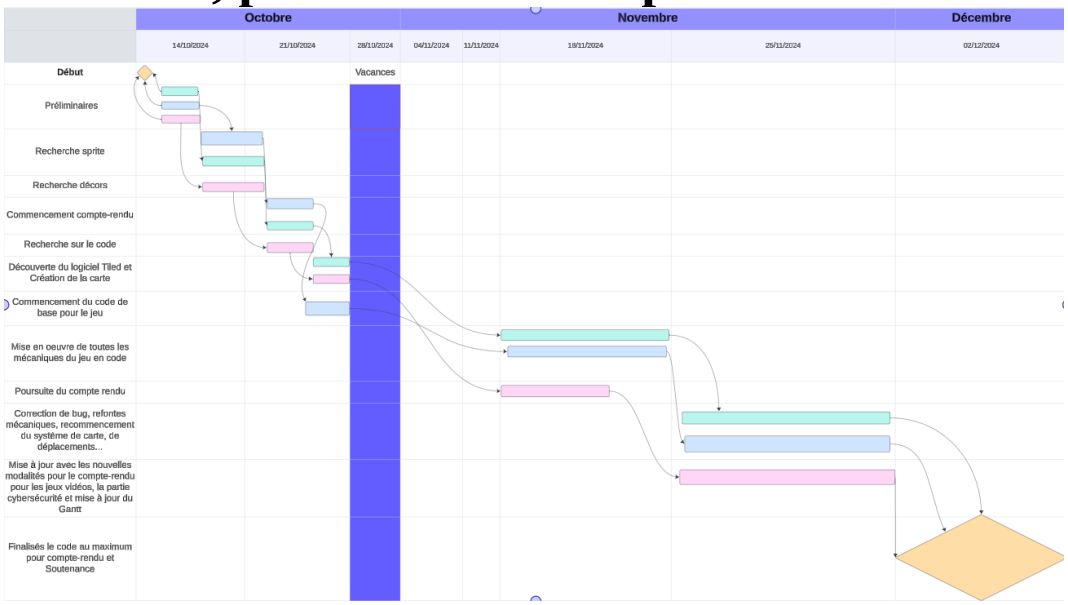
Il faut vérifier les fichiers utilisés dans le risque ou ils seraient corrompus.

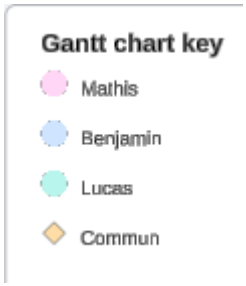
Dans le même thème il faut limiter d'utiliser des bibliothèques inconnues dans notre code.

Empêcher l'accès non autorisé aux fichiers en utilisant des permissions système restreintes.

Possibilités de scanner le projet avec différent outils comme Safety pour détecter des vulnérabilité et PyArmor pour contrer le reverse engineering.

4. Phases, planification et répartition des tâches





5. Budget et ressources

Estimation du coût total du projet : entre 45 et 60 heures

Détail des ressources nécessaires (humaines, matérielles, financières) :

Humaines : 3 personnes.

Matérielles : Pc portable avec logiciel (Pygame, Pycharm, Aseprite, Tiled)

Financières : pas de financement

6. Critères d'acceptation :

Le projet sera considéré terminé si toutes les fonctionnalités de base (capture de virus, combat, exploration, scénario) sont implémentées et fonctionnent sans bugs majeurs. Les graphismes et animations doivent être cohérents et de bonne qualité, et le jeu doit être stable.

La validation inclut des tests de fonctionnalité, d'utilisabilité et de performance. Enfin, le jeu doit être publié et accessible.

7. Risques et plan de gestion des risques :

Les risques potentiels incluent des bugs majeurs, des retards dans le développement, des problèmes de performance et un mauvais téléchargement par les utilisateurs. Pour les atténuer, des stratégies telles que des tests réguliers, une bonne gestion du projet, des optimisations de code et des tests de performance.

Partie Personnelle :

Partie Personnelle Mathis WARCHE :

1. Introduction / Positionnement :

Tâches à réaliser :

-Recherche Sprite : Benjamin / Lucas

-Recherche décors : Mathis

-Commencement compte-rendu : Benjamin / Lucas

-Recherche sur le code : Mathis

-Création de la carte : Mathis / Lucas

-Commencement du code : Benjamin

-Codage du jeu et des mécaniques : Lucas / Benjamin

-Poursuite compte-rendu : Mathis

-Correction de bug, refontes mécaniques, recommencement de la carte… : Lucas / Benjamin

-Mise à jour avec les nouvelles modalités pour le compte-rendu pour les jeux-vidéos, la partie cybersécurité et mise à jour du Gantt : Mathis

-Finalisation du code pour Compte-Rendu et passage soutenance : Mathis Lucas Benjamin

2. Fin de l'Analyse :

2.1- Capture de virus (scénario typique)

Acteurs :

- Joueur
- Virus sauvage

Étapes du scénario :

1. Le joueur explore une zone.
2. Un virus sauvage apparaît aléatoirement.
3. Une séquence de capture est déclenchée :
   - Le joueur peut affaiblir le virus en utilisant ses Waremon.
   - Il choisit un type de firewall pour capturer le virus.
4. En fonction des probabilités (et de l'état du virus), la capture réussit ou échoue.
5. Si la capture réussit :
   -Le virus est ajouté à l'équipe ou au "Data Vault".
   -Une récompense (expérience, objet) est donnée.
6. Si la capture échoue, le joueur peut :
   -Réessayer.
   -Fuir ou continuer le combat.

2.2- Combat contre un boss (scénario critique)

Acteurs :

- Joueur
- Boss (hacker avec ses Waremon)

Étapes du scénario :

1. Le joueur arrive dans une zone clé où un boss attend.
2. Une interaction avec le boss est initiée (dialogue narratif).
3. Le combat commence :

   -Les deux parties jouent à tour de rôle.

   -Le joueur peut choisir une action : attaquer, changer de Waremon.
4. La stratégie est essentielle :

   -Exploitation des faiblesses des types de virus.

   -Gestion des points de vie et d'attaque.
5. Si le joueur gagne :

   -Le boss est vaincu, un objet clé ou une progression dans l'histoire est débloqué.
6. Si le joueur perd :

   -Fin du jeu le jeu se ferme.

2.3- Exploration d'une nouvelle zone (scénario typique)

Acteurs :

- Joueur
- Environnement (PNJ, objets interactifs, zones d'exploration)

Étapes du scénario :

1. Le joueur entre dans une nouvelle zone.
2. L'exploration est libre, avec des événements aléatoires comme :

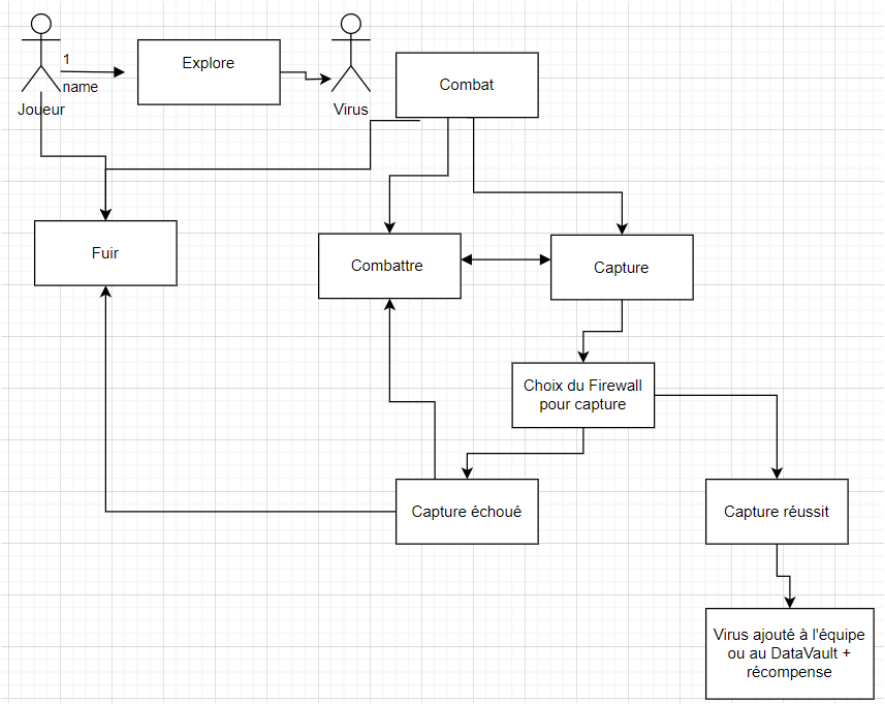
   -Apparition de virus sauvages.

   -Découverte d'objets ou indices sur l'histoire.

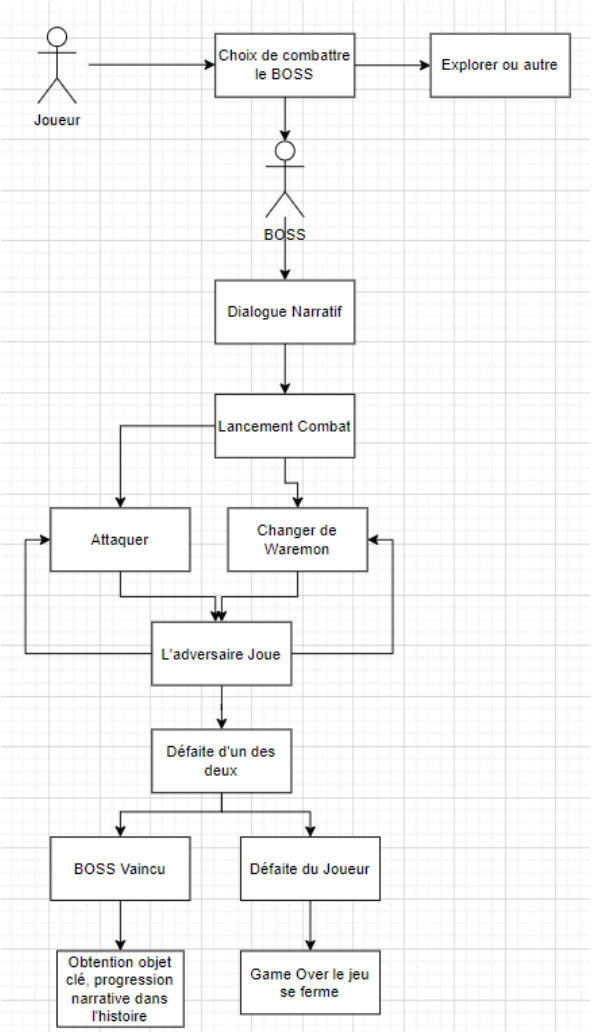
   -Dialogue avec des PNJ offrant des conseils, et des informations.
3. Le joueur peut interagir avec l'environnement pour avancer dans le jeu ou entraîner ses Waremon.
4. Il peut aussi rencontrer des Hackers (PNJ) qui le prendront en duel.

2.4- Création des Diagrammes :

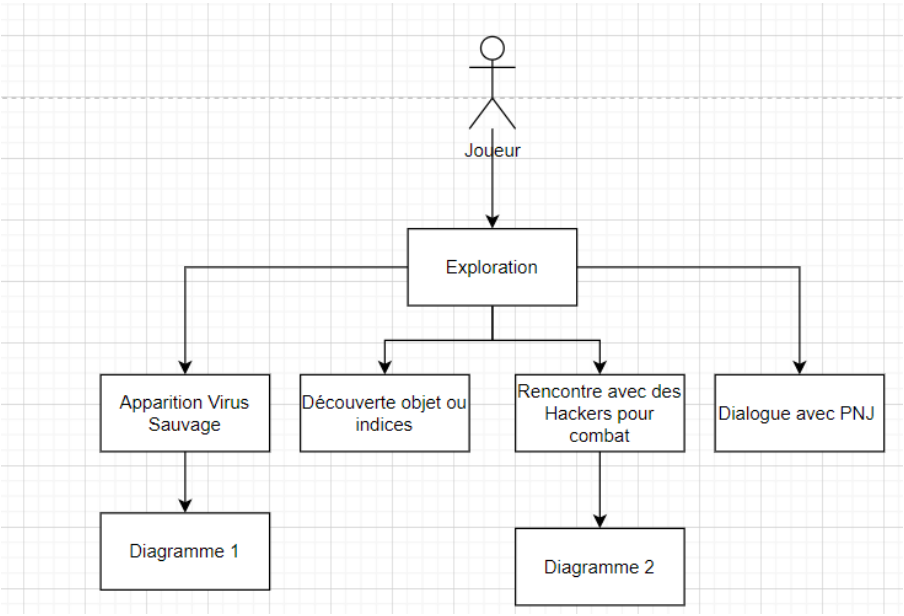
Capture d'un virus :



Combat contre BOSS :



Exploration :



3.Conception :

3.1-Conception Générale :

Le jeu sera codé en Python sur Pygame en prenant comme exemple les anciens jeu Pokémon.

Grâce à de nombreuses ressources disponibles un peu partout telle que des vidéos YouTube, documents sur des blogs nous avons pu nous renseigner sur comment coder un jeu et quelle était la meilleure voie pour nous pour créer notre jeu.

Où l'on va pouvoir combattre, collectionner les Waremons et explorer.

-Il y aura donc un système de combat au tour par tour avec un système de Point de Vie pour nos Waremons et des attaques plus ou moins efficaces.

-Nous pourrons explorer la carte pour découvrir sa diversité.

-Et suivre l'histoire à travers les différents dialogues des PNJ.

Interface Utilisateur :

Il y aura une interface assez simple pas poussée, elle sera donc facile à prendre en main et même quelqu'un qui n'a jamais jouée à des jeux pourra comprendre.

Questions sonores nous ajouterons une ambiance sonore d'un aspect futuriste/électro pour suivre avec le thème informatique dans des dizaines d'années.

Difficultés rencontrées et Solutions apportées :

Pour ce qui est des difficultés rencontrées je dirais que la plus grande des difficultés a était en sois de créer un jeu vidéo en n'ayant en tout cas pour ma part aucune expérience dans la création de jeu vidéo.

Donc il a fallu une grande partie de recherches un peu partout pour établir un plan de ce que l'on allait faire et comment ont allaient le faire.

Il a fallu faire des erreurs aussi pour comprendre encore plus notre code, le corriger et avancés dans le Développement.

3.2-Conception Détaillée :

Tout d'abord le choix de notre langage de programmation a été assez vite dirigé sur Python pour PyGame et je pense que pour notre jeu ça a était un bon choix, non pas que d'autres moteur de jeu vidéo n'auraient pas bien fonctionner mais pour nous, nous ne sommes pas déçus d'avoir utilisé PyGame.

Pour ce qui est de l'affichage nous avons utilisé Tiled pour la création de la map, les différentes couches notamment qui serviront au bon fonctionnement des collisions et de l'immersion dans le jeu. Les différentes implémentations comme les maisons, verdures, personnages ont était empruntés dans des bibliothèque libre services.

Pour suivre dans l'affichage du jeu et du gameplay les rencontres du personnages joueur avec les personnages non joueur (PNJ) seront prévu à différents endroits de la map chaque (PNJ) aura soit pour rôle d'engager un combat avec nous ou alors il pourra juste offrir un dialogue ou même rien du tout.

Toutes les rencontres et l'exploration auront pour but de nous mener au Boss de Fin pour finir le jeu.

4-Implémentations :

Pour ce qui est des implémentations, nous n'avons pas fait plusieurs fonctions que l'on a ajoutées au fur et à mesure, mais nous avons développé le code dans l'ensemble directement, ce qui à ses atouts et ses faiblesses, car cela nous permet de modifier directement le code et voir les résultats, mais si ça ne fonctionne pas, c'est tout le code qui ne fonctionne plus, ce qui n'est peut-être pas la meilleure des manières, cependant ça nous a obligés à être beaucoup plus assidus sur notre façon de coder car chaque ligne était importante dans le bon fonctionnement du code en général.

Hors le code, il y a aussi les implémentations graphiques telles que la Map, les Sprite qui ont du être ajoutés à leur tour.

5- Tests unitaires et validation / Test D'intégration :

Pour ce qui est des Tests nous y sommes allés petit à petit c'est-à-dire que pour chaque fonctionnalité nous la testions à son niveau le plus primaire et si elle fonctionnait ont poursuivaient le code pour rendre la fonctionnalité opérationnelle et comme nous la voulions.

Exemple pour les déplacements nous avons commencé avec un petit carré rouge qui se déplacé sur la Map puis nous avons centré la caméra sur lui puis changé ce carré en un personnage qui n'avait qu'une face et au final nous avons un personnage qui peut se mouvoir dans toutes les directions avec des animations adaptées.

6- Discussion :

Pour ce qui est des choix je pense qu'une approche du code différente aurait permis une meilleure réalisation du code en faisant plus de fonctions indépendantes des autres cependant, certaines fonctions était dans tous les cas dépendants d'autres mais nous aurions peut-être pu faire plus de fonctions indépendantes.

Ce qui aurait permis une encore meilleure réalisation collective du projet.

Au niveau du choix du moteur de jeu vidéo Pygame je pense que ça a était un bon choix il existe certes d'autres moteurs peut-être plus adaptés ou autres mais la prise en main de PyGame et le résultat qu'on a réussi à en tirer me satisfait du choix.

Partie Personnelle Lucas PAVONE :

1. Introduction

Dans le cadre du projet Waremon, mon rôle principal a été de développer les aspects fondamentaux du jeu, c’est-à-dire :

- l’affichage de la carte,

- l’affichage du personnage jouable, avec suivi de caméra et déplacement,

- l’affichage des autres personnages non jouables (PNJ du type Dresseur),

- les ombres pour les personnages (jouables ou non),

- l’animation de l‘eau et des côtes,

- les collisions,

Dans mes responsabilités spécifiques, il est inclus :

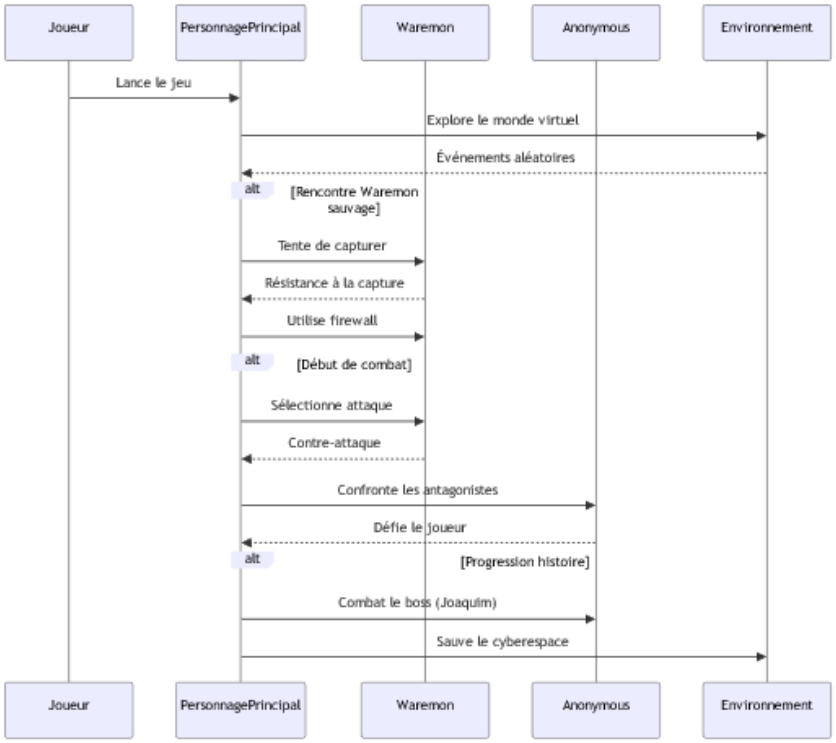
- la gestion du Github,

- la création de la carte avec le fond sonore,

- la création des visuels pour les Waremon, ainsi que du fichier contenant les informations de chaque waremon (en .py) contenu dans le projet

2. Fin de l’analyse

Voici un diagramme de séquence (simplifié) pour illustrer les différents scénarios et cas d’utilisation possibles :



3. Conception :

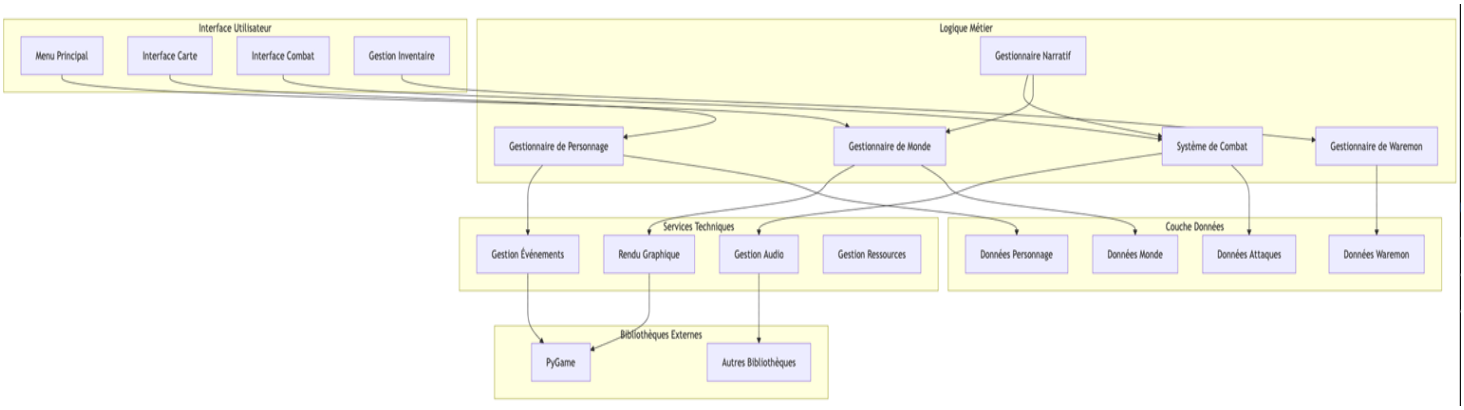
L’objectif principal du projet est de développer un jeu RPG en 2D (du même style que Pokemon) où l’on capture des virus informatiques.

Au niveau de la conception du projet, il est conçu de façon modulaire pour permettre à chacun de développer différentes parties du code en même temps, et aussi permettre d’y voir clair sur quels « modules » du code permet de faire quelles actions en jeu.

Au niveau des modules, nous en avons plusieurs, certains pour des réglages généraux, d’autres pour créer l’affichage de chaque personnages(jouables ou non), etc.

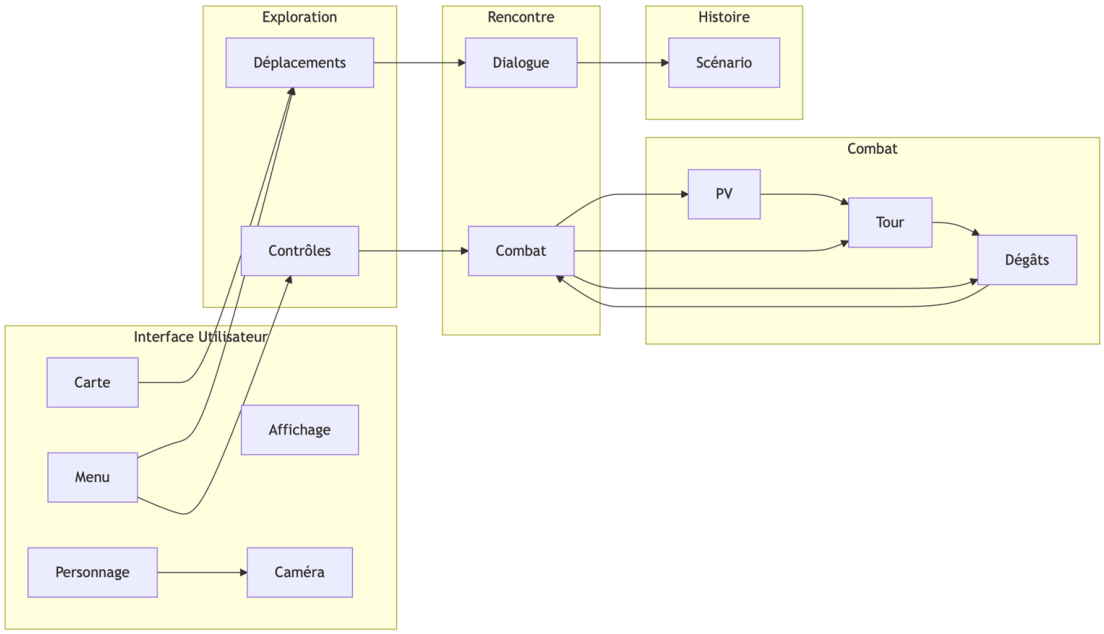
Au niveau des considérations techniques, nous avons choisi d’utiliser python et plus particulièrement les librairies pygame-ce ainsi que pytmx (pour charger la carte qui est dans ce format) car c’est ce qui est couramment utilisé pour développer ce genre de petits jeux. Il n’y a pas de plateformes prédéfinies pour jouer au jeu mis à part l’installation de python et des librairies, il peut donc être joué sur Windows, MacOS, Linux.

Voici un diagramme d'architecture logicielle permettant de comprendre la structure du système :



Au niveau des difficultés que nous avons pu rencontrer il y a tout d'abord, l'affichage de la carte car il y avait des problèmes aux niveaux des chemins, ce qui nous a pris un bon moment pour corriger ce problème. Nous avons également eu des problèmes au niveau de la création de la carte car Tiled est un logiciel plutôt complexe de prime abord, ayant réalisé la carte j'ai dû recommencer celle-ci plusieurs fois ainsi qu'en refaire d'autres versions plusieurs fois afin qu'elle corresponde d'une part à nos envies et d'autre part pour qu'elle puisse être utilisable facilement dans le projet.

Pour mieux expliquer le système de modularité je joins un petit diagramme :



Interface Utilisateur

• Affichage : Ce module gère tout ce que le joueur voit à l'écran, notamment la carte, le personnage.

• Caméra : Suivant les déplacements du personnage, la caméra ajuste le point de vue pour maintenir le focus sur les zones importantes.

• Menu : Permet d'accéder aux options du jeu, sauvegarder, charger ou quitter le jeu. (Optionnelle à voir si le temps de développement nous permet de le réaliser)

Exploration :

• Déplacements : Le joueur peut déplacer son personnage dans l'univers du jeu. Les déplacements sont influencés par les contrôles qui interprètent les actions du joueur.

• Contrôles : Ce module interprète les entrées utilisateur pour contrôler les actions du personnage et interagir avec les autres fonctionnalités.

• Carte : Une représentation visuelle du monde du jeu, où le personnage évolue. La carte inclus des couches spécifiques (collisions, mouvement de vague, etc.) et est associée à la caméra.

Rencontre

• Dialogue : Lorsqu'une rencontre survient (avec un PNJ, par exemple), un dialogue peut s'engager pour enrichir l'histoire ou déclencher un combat.

• Combat : Les rencontres peuvent inclure un passage au mode de combat. Ce module se connecte au système de contrôle, d'affichage et d'histoire.

Histoire

• Scénario : Représente la trame narrative du jeu. Chaque interaction, dialogue ou combat fait progresser le joueur vers la conclusion.

Combat

• PV (Points de Vie) : Les points de vie représentent la santé des personnages. Ils diminuent en fonction des dégâts subis.

• Tour : Le combat est basé sur un système de tour par tour, où chaque personnage effectue une action à tour de rôle.

• Dégâts : Calculés en fonction des attaques choisies par le joueur ou l'ennemi. Ces dégâts modifient les PV.

Connexion entre les modules

1. Les contrôles alimentent directement les déplacements, la caméra et les actions en combat.

2. Les rencontres sont déclenchées dans le cadre de l'exploration, à des points spécifiques (par exemple, coordonnées précises sur la carte).

3. Le combat, une fois initié, interagit avec les modules de PV, tour, et dégâts pour gérer la résolution des affrontements.

4. L'interface utilisateur permet au joueur de visualiser son environnement (carte, personnage) et d'accéder aux menus pour configurer son expérience.

4. Implémentation :

Au niveau de l'implémentation, nous n'avons pas implémenté des fonctions de façon classique, en effet, nous avons ajouté des fonctionnalités en modifiant plusieurs « modules » en même temps.

Au niveau des problèmes rencontrés, il y a les collisions, en effet, bien qu'au niveau du code cela n'a pas été complexe, en revanche, mettre en place les « blocs » de collisions sur le logiciel Tiled a été complexe car parfois de par la taille de ces blocs les collisions ne se faisaient pas ou mal. Nous avons donc dû jongler en quelques sortes entre la carte et les tests après chaque modification pour vérifier.

5. Tests unitaires et validation :

Pour tous les tests unitaires, nous avons procédé par fonctionnalités, par exemple, lors du développement des déplacements nous sommes passés par un carré rouge, puis au sprite du personnage, pour nous assurer que tout fonctionnait de façon séparée sans l'affichage du personnage.

De même avec les collisions, nous avons d'abord tester les collisions des objets puis celle des surélévations de la carte.

Pour chacune de ces fonctionnalités, nous avons vérifié si elles fonctionnaient avant de passer à la suite, si ce n'était pas le cas nous résolvions le problème avant de continuer.

6. Test d'intégration (collectif) :

Comme nous avons fait les tests unitaires ensemble la plupart du temps, ces tests ont été similaires.

7. Discussion :

Ce projet a été pour moi très intéressant, car je suis plutôt attiré par le développement depuis très longtemps, cela m'a permis de me rendre compte de ce qu'était un projet et de comment le mettre à bien de A à Z. Cela m'a également permis de recoder en python car je n'en ai que très peu l'occasion, et cela m'a donné envie de réaliser plus de projets personnels de ce type. Au-delà de recoder en python cela m'a permis de découvrir la programmation en python avec des librairies tels que pygame(-ce) ou encore pytmx. Même si ce projet n'arrive pas totalement à son terme (ou en tout cas qu'il n'est pas dans nos attentes), j'aimerais le poursuivre pour voir jusqu'où on pourrait arriver. Je pense que nos attentes pour le projet ont été trop élevées dès le départ par rapport au temps que nous avons eu.

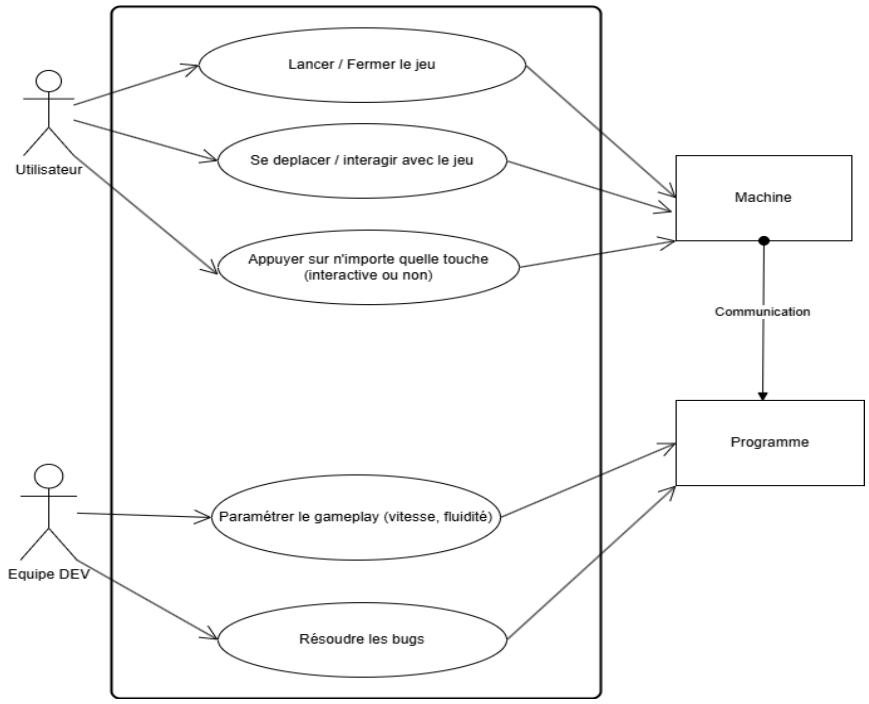
Partie personnelle : HUON BENJAMIN

1.Introduction

Dans ce projet, mon rôle a été de commencer le code de base sur la première inspiration que nous avons eu, de faire les commentaires du code pour une meilleure compréhension d'un œil extérieur au projet, ainsi que d'aider, comme mes camarades, à la recherche des ressources nécessaires pour les visuels.

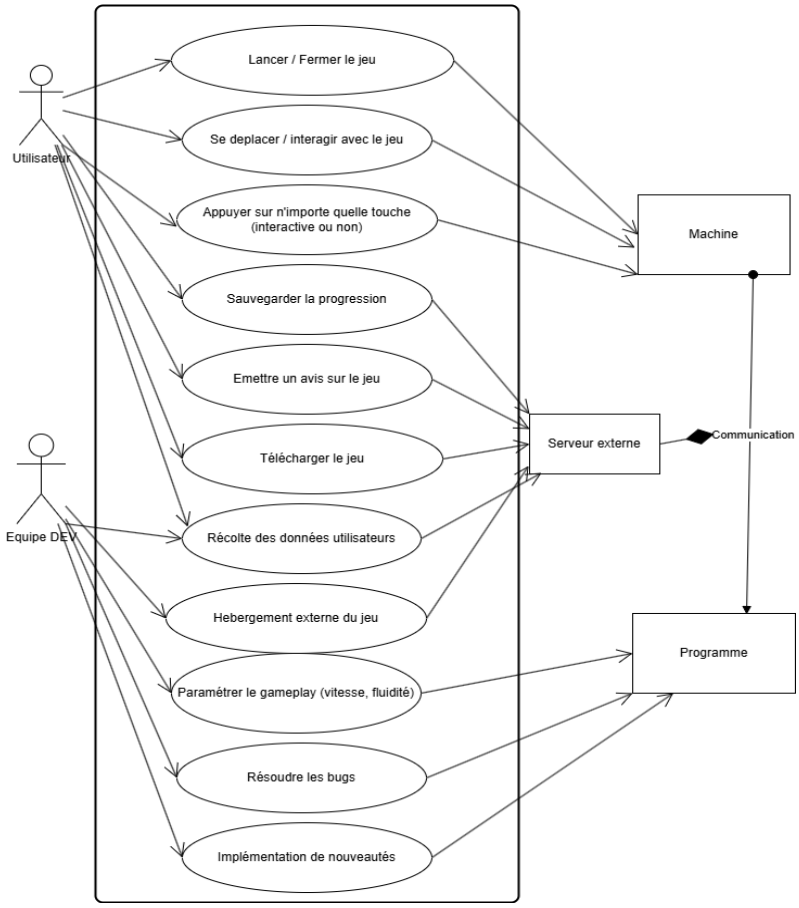
2. Fin de l'Analyse

Pour ce qui est de l'analyse, nous avons recherché les besoins utilisateurs, trouver les solutions pour y répondre, mais surtout nous avons exclus tous les possibles scénarios qui nous auraient retardés. Voici en représentation de diagramme UML le résultat final des besoins utilisateurs qui ont été retenus



Ici, si tous les besoins possibles avaient été retenus, nous y trouverions une option de sauvegarde, une option d'interface de lancement, une option de personnalisation de visuel du personnage principal et bien d'autres.

Voici un exemple (pas représentatif à 100%) de ce qu'aurait pu être le diagramme UML si toutes les fonctionnalités possibles avaient été retenues :



Certaines questions, comme les critères concernant le matériel nécessaire pour exécuter l'application, n'ont pas particulièrement retenu notre attention, étant donné les faibles ressources sollicitées par notre code. Cependant, l'exploration des outils logiciels pour la programmation de ce jeu a tout de même été abordée dès le début du projet, notamment en raison du choix d'Unity.

3. Conception

3.1 Générale :

Pour ce qui est de la conception générale, le jeu sera conçu et est conçus de façon schématique / modulaire pour permettre une meilleure gestion du code, des bugs et des implémentations.

Langage : Le choix qui nous a semblé évident est de coder en python, le langage est universel pour des petites applications comme nous voulons faire, la bibliothèque Pygame (Pygame-ce pour nous) semble très complète, et de nombreuses vidéos sont à notre disposition.

Plateforme : Nous n’avons pas spécialement choisis de portage sur une plateforme extérieure à Windows car nous exécutons notre programme dans une fenêtre Pygame.

Système de programmation modulaire :

1. Affichage : Système d’affichage de la carte, avec un personnage dont la caméra est centrée sur celui-ci.

2. Exploration : Aussi appelé système de déplacement, grâce aux touches directionnelles le personnage (et la caméra) se déplacent n’importe où sur la carte proposée.

3. Rencontre : Selon l’endroit où le personnage se situe, il peut rencontrer un autre personnage, ce personnage ouvre donc un dialogue puis par la suite un…

4. Combat : Mécanisme de combat au tour par tour, sans avantages selon des types, avec des choix d’attaques, des PV, des dégâts.

5. Histoire : Un « progrès » narratif est observable, pour arriver à une fin, l’affrontement FINAL.

L’interface

Nous ne voulons pas offrir une interface des plus complètes, du fait de la difficulté. Donc pas de « Menu », pas de différentes touches interactives. Les flèches directionnelles et la souris répondront essentiellement aux interactions possibles.

Mais pour une expérience un peu plus plaisante, nous voulons avoir une ambiance sonore. Qui semble un peu arcade retro lors de l’exploration et épique lors des combats.

Difficultés et Solutions

La difficulté principale est simplement l’expérience. Construire de A à Z une application de cette envergure est un défi. Pour y remédier, c’est uniquement de la motivation et de la réussite qui est nécessaire.

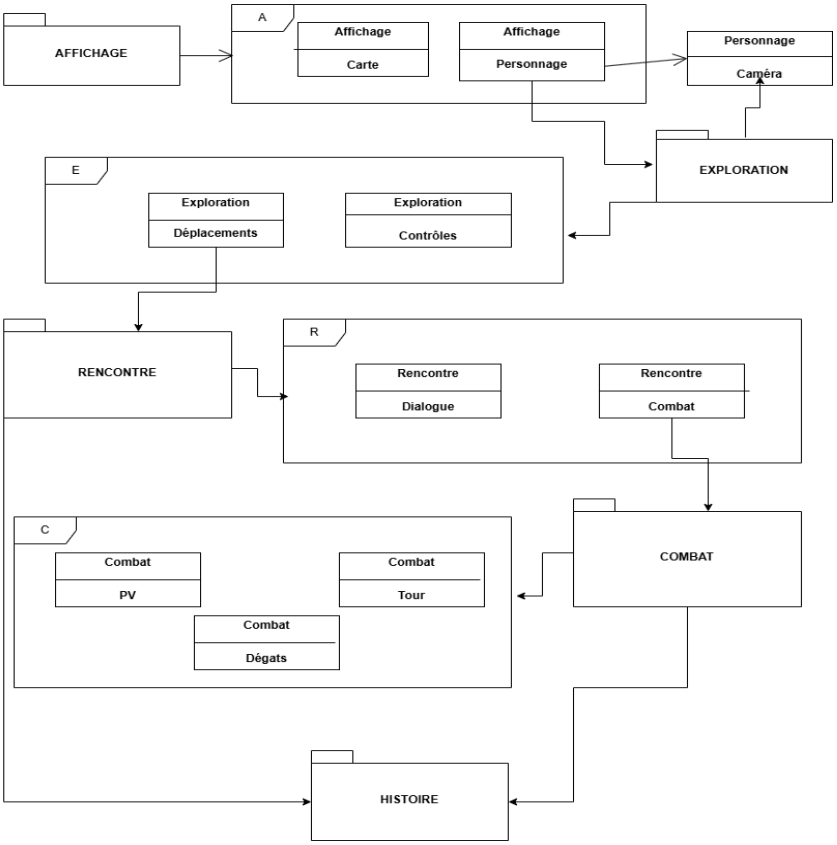
D’autres problèmes se sont présentés comme des erreurs de codes que nous avons du résoudre avec l’aide de forums, d’intelligence artificielle, d’aide extérieure (professeurs, camarades). La production des ressources externes aussi peut être considéré comme un problème, en sachant que nous pouvions emprunter des cartes déjà existantes, nous avons choisis de faire parler notre créativité, pour en quelque sorte y voir notre ADN.

3.2 Conception Détaillée :

Notre choix de faire un programme de façon modulaire est pour moi le meilleur choix, de ce fait nous pouvons avancer « étapes » par « étapes », la programmation orientée objet en est l’inspiration mais au lieu de programmer une fonction et de l’implémenter dans le code. On modifie l’ensemble du code, qui à la fin ajoute la fonctionnalité souhaitée. Cela est un frein au travail simultané sur des machines différentes mais un point fort pour le programme final car la progression dans le code suit l’ajout des fonctionnalités.

Un choix du langage nous a été demandé au début du projet. On a évoqué de faire ce jeu sur Unity ou Python (pygame) par exemple, avec un peu plus de recherche lors du projet, j'ai découvert que Ruby aurait pu être un bon choix aussi. Notre choix s'est finalement porté sur Python car cela semblait plus simple, car nous avons les bases, de nombreuses aides (vidéos YouTube, forums, …).

Le système de modularité ne pouvant être mieux expliqué, je propose un visuel pour mieux le comprendre :



L'affiche comprends donc la carte, faite sur Tiled, avec des couches pour gérer par la suite les collisions, bords de maps.

Le personnage principal ainsi que les personnages annexes qui correspondent aux rencontres sont des images empruntées donc déjà existantes. Les animations de déplacement sont faites par successions de 4 images. Les contrôles sont faits grâce aux fonctions pré-existantes de la bibliothèque Pygame.

Les rencontres ont lieus lorsque le personnage principal se trouve à proximité d'un endroit aux coordonnées précises, comme l'emplacement d'un personnage annexe.

Les combats sont gérés au tour par tour, avec une barre de vie, 2 attaques (sauf modifications) qui ont leur dégâts prédéfinis.

Toutes ces successions de fonctionnalités font une progression de l'histoire pour arriver jusqu'à l'affrontement final et finir le jeu.

4. Implémentation

Pour ce qui est de l'implémentation, nous ne faisons pas des ajouts de fonctions mais plutôt des ajouts de fonctionnalités ce qui apportait des modifications à plusieurs fichiers à la fois.

On ne peut donc pas vraiment parler d'implémentation au sens propre. Sauf pour ce qui est des ressources, assets (cartes, personnages (qui n'est pas de mon sort)).

Les seuls exemples d'implémentations directe sont des résolutions de bugs, lorsque nous étions bloqués. Et aussi l'ajout de tous les commentaires, décrivant les boucles, fonctions, classes, pour une meilleure compréhension du code.

5. Test unitaires et validation

Les ajouts de fonctionnalités se faisaient en « parties », par exemple pour ajouter le personnage ainsi que les déplacements de celui-ci, on a premièrement ajouté un simple carré de couleur puis nous avons testé s'il n'y avait pas de soucis.

Si aucun souci n'était détecté, on ajoute donc le centrage caméra puis on test, ensuite les déplacements, le personnage final, les animations et ainsi de suite.

A chaque petit ajout, nous vérifions si le programme s'exécutait correctement, si ce n'était pas le cas **on** résolvait le bug avant de passer à la prochaine étape.

6. Tests d'intégration

Les tests unitaires et d'intégrations suivaient le même fonctionnement à la lettre.

7. Discussion

Ce projet est pour moi un choix très intéressant car il lie plusieurs thèmes que j'apprécie fortement, en particulier la cybersécurité. Donc ce choix thématique fait partie intégrante de la motivation à réussir ce projet. L'exploration de plusieurs langages de programmation aurait pu être très intéressant mais malheureusement le temps était et est pour moi notre plus gros désavantage. Un jeu de ce type étant produit dans son entièreté en environ 1 an, ce n'est pas en 2 mois qu'on aurait obtenu un jeu à la hauteur des attentes de tout le monde, mais malgré ça, le résultat reste entièrement satisfaisant.

Pour la suite je serai intéressé de savoir ce qu'aurait donné ce projet dans un autre langage de programmation et jusqu'où l'aboutissement de ce projet aurait pu aller.