|  |
| --- |
| The Survivor  *Rapport final*  HM40-2022 |
|  |
| 23 juin  HM40 – The Survivor  Créé par : Dujarrier-Lesueur Raphaël  & Jiang Yiwen |



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sommaire   * 1. **Cahier des charges**      1. **Questionnaire et retour utilisateur**      2. **Etude de l’utilisateur et PersonA**   2. **Conception et Architecture de l’appli**      1. **SADT**      2. **Objectif et but**   3. **Backend**   4. **Front-end**      1. **Visuel**      2. **Non visuel**   **Préface**  The Survivor est un platformer 2D dans lequel le joueur incarne le personnage d’Eden, seul survivant, ou presque, de la crise environnementale mondiale ayant décimé les humains il y a de ça une génération.  Le joueur évolue au travers du jeu grâce à certaines touches du clavier et de la souris.  L’enjeu est de sauver, au travers du personnage d’Eden, les humains peuplant encore la Terre en atteignant la fabrique de capsules, denrées indispensables à l’approvisionnement alimentaire des survivants. En effet, cette fabrique c’est mystérieusement arrêté, et risque bien de décimer le peu survivant restant. On découvrira que les marcheurs noirs, les ennemis du jeu, sont responsables de ce désastre.  Ce projet visait à mettre en œuvre les principes d'analyse, de conception et de développement d'une interface homme-machine et à produire un jeu sur la base d’une conception centrée utilisateur, exclusif à l'UV HM40.  Durée de conception : 2 mois et demi à pleine charge   1. **Cahier des charges** 2. **Questionnaire et retour utilisateur**   Sur la base des réponses au questionnaire que nous avons recueillies, nous avons décidé d'assurer les fonctions suivantes :  => Le temps de jeu des répondants étant compris entre 1H et 3H, il faut réussir à développer un jeu qui sache ne pas être trop répétitif et surtout avec une durée de vie à minima correcte (en jouant sur l’aléatoire, la replay valu et le niveau de difficulté par exemple)  => Réglage précis du volume, monter/baisser le niveau de la musique et des bruitages depuis le jeu, pour pouvoir écouter d’autres choses en parallèle. En effet, nos futurs utilisateurs ont exprimé le besoin d'écouter d’autres choses (musique, chat vocal…) en parallèle du jeu.  =>Les utilisateurs ont également exprimé leur préférence vers des logiciels de communication externe (Discords, réseaux sociaux…). Nul besoin donc de créer un chat propre au jeu, mais on peut envisager de faciliter l’accès aux chat externe depuis notre interface (Bouton d’ouverture…etc)  => Permettre la personnalisation des commandes de jeu (choix possible entre les flèches et ZQSD)  => Permettre le choix de la langue (Le besoin se fait ressentir en Français/Anglais)  => Nous pensions permettre le choix homme/femme du personnage :mais nous nous sommes rendu compte que cela était la dernières des priorité pour nos répondant, concernant la personnalisation de l’expérience de jeu.  => Apporter un soin particulier aux couleurs, à la forme des boutons, leur intégration dans les menus et sur l'écran ou encore le format des titres  => Créer et soigner le lore du jeu et sa présentation, son intégration, la manière de le raconter en introduction au joueur (penser aussi à un bouton skip)  => Envisager un système de score, bien que nous demandé en priorité par les joueurs, car pouvant favoriser la replay valu  => Faire en sorte que l'app ne prenne pas tout l'écran, puisque les gens font aussi d'autres activités, qui nécessitent de l'espace sur leur écran. De plus, notre jeu est simple et adapté à la pratique d'autres activités en parallèle. Une fenêtre de taille restreinte est donc idéale.  => L’utilisation du jeu à été recensé comme étant pour se détendre, en vacances majoritairement, ou bien pour déstresser pendant les cours. =>Objectif global, penser à autre chose et se détendre  => Les gens jouent beaucoup le soir : envisager un mode sombre, pour éviter la fatigue oculaire  => Les joueurs portent une attention particulière à l'interface de jeu et à l'écran d'accueil. Nous soignerons donc particulièrement leur implémentation et affichage, pour favoriser l’immersion du joueur dans ces zones du programme.  => Améliorer jonction fond d'écran in game, améliorer texture sol, avoir plusieurs sorte/couleurs de platform  => Les gens jouent chez eux : c'est idéal d'un point de vue conception !   1. **PersonA et base de travail**   Primo-Buts :  A travers ce jeu, notre but était de nous familiariser avec l’univers du platformer 2D. Nous nous sommes pour cela inspirés de certains concepts des jeux *Inside*, pour l’ambiance de jeu et l’aspect « histoire » de celui-ci, et *Mario Bros*, pour l’idée de vie limité/temps limité et d’objet à récupérer.  L’idée principale était de réussir à faire évoluer un personnage à travers une fenêtre sur les modèles des jeux précédemment cités*.* Nous voulions que celui-ci soit essentiellement dirigé grâce aux touches du clavier durant les sessions de jeu et que la souris puisse intervenir dans certains choix de personnalisation.  De plus, avant de se concentrer sur l’aspect IHM, il était nécessaire d’y insérer les bases de n’importe quel jeu de plates-formes, à savoir : un menu principal (avec ses futures options), les crédits (principalement liés ici aux droits d’auteurs éventuelles), évidemment la boucle du jeu avec un système de gravité, de collision et de vie, un affichage game-over et, à l’inverse, un affichage en cas de victoire, ainsi qu’un système de menu pause dans un second temps.  Sur cette base fertile, est ensuite envisageable l’apport de fonctionnalités supplémentaires demandées par les utilisateurs ou jugées utiles à notre conception IHM de l’application.  De part le but visé, nous avons choisi, à partir de Python, de travailler sur la bibliothèque Pygame. En effet, ce module permet, à partir du langage Python, de faciliter le développement de jeux vidéo d’un point de vue front-end et donc de programmer la partie multimédia (graphismes, son, interactions clavier et souris…) indispensable aux interactions avec l’utilisateur.  Outils :  Pour réaliser ce jeu, nous avons donc dû apprivoiser les particularités du module Pygame (draw, mixer, sprites…) en les combinant avec les commandes de Python (liste, random, time, fonction, class…).  En ce sens, nous avons donc commencé par suivre le tutoriel YouTube d’Alexandre Ghelli sur le sujet (<https://www.youtube.com/watch?v=IGSmLvfEz18&feature=youtu.be>), afin d’avoir une première idée des commandes utiles sur Pygame, pour l’affichage et l’animation d’un objet.  Nous nous sommes ensuite beaucoup informés pendant l’implémentation du projet via :   * Des forums: Open Classroom, Hardware… * Sur des sites : Open Classroom, Pygame.org, Isn-marchal.com, pythonprogramming.net * Grâce à des chaines YouTube : TechWithTim, KidsCanCode, sentdex   Internet a donc été une source d’information importante, mais nous avons pu également interagir avec professeures et élèves pour parfois nous sortir de problèmes inextricables.  Afin de progresser dans l’élaboration de notre jeu, il a fallu se répartir certaines tâches. Cependant, beaucoup d’entres elles ont été réalisées à la fois conjointement et séparément.  Nous devions donc chacun pouvoir communiquer avec l’autre, que ce soit pour échanger des idées ou les programmes.  Nous avons principalement communiqué via Discord pour les échanges collaboratifs et via Git pour l’envoi de fichiers et d’images, mais nous nous sommes rencontrés dès que cela était possible, en cours (TP) et hors cours.  Persona :  Sexe : Homme-Femme (issu du questionnaire : 70% des hommes y jouent et 30% des filles y jouent. A l’aide d’article et d’étude trouvés sur internet, ainsi que la non représentativité de notre questionnaire, on peut en déduire que ce type de jeu attire plus facilement les filles, que d’autres types de jeux)  Niveau : Joueur habitué, parfois expert (fréquence journalière ou hebdomadaire, avec des sessions de 1H à 3H)  Préférence de jeux : Plutôt orienté nostalgie, jeu simple d’enfance  Méthode de jeu : Plutôt hors ligne et en solo (logique)  Age :(La nostalgie apparaît une fois l’âge dépassé, donc :) 20-30 ans  Etude : Biaisé, étant nous même dans un environnement étudiant  Activité pro/perso : idem  Moyen de communication préféré : Discord (plutôt que chat intégré)  Période de jeu: Vacances (50%), week-end, hiver, fin de journée (et tout ce qui leur laisse du temps libre)  Lieu de jeu : Chez eux  Support de prédilection : PC   1. **Conception et Architecture** 2. **SADT**   Niveau A-0  Jeu - The Survivor  Joueur  Ordinateur  Absence de score  Paramètres par défauts  Record établi  Paramètres personnalisés  Joueur disponible ET désireux  Programme téléchargé ET exécuté  Expérience acquise  Personnalisation Options  Niveau A-1  Démarrage jeu  *Score initial*  Paramètres par défauts  Consultation Score et crédits  Paramètres par défauts  *Score*  Menu  Nouveaux paramètres  Fin de session  *New Score*  Raffinement de niveau 1  Nouveaux paramètres  Paramètres par défauts  Personnalisation Options  Choix difficulté  Choix bruitages/son  Choix langues  Choix touches  Niveau A-2  *Facile*  *Français*  *Flèches*  *On: 100%*  *Facile ou moyen ou difficile*  *Français ou anglais…*  *ZQSD ou flèches…*  *Off ou ajusté*  *SI clique*  *SI clique*  *SI clique*  *SI clique*  Raffinement de niveau 2 des options  *Position perso > 19000*  Récolter collectibles  Avancer sur la carte  Esquive d’ennemies  Niveau A-2  Démarrage jeu  *T = 0s*  *Vie à 100%*  *Position carte*  *Saisie clavier*  *Position perso*  *Position ennemie*  *New position*  *Mort ou non*  *Modif vie*  *Défilement décors*  *Inertie perso/ennemie*  *Position perso*  *Position objet*  *Modif vie*  *Disparition collectible*  *Arrivé ? Fin ?*  Victoire  *Score = T*  Raffinement de niveau 2 d’une partie   1. **Objectifs et Buts**   *-Une implémentation répondant à des besoins-*  Sur la base de l’analyse des besoins, nous en avons déterminés les objectifs suivants :  => Le temps de jeu des répondants étant compris entre 1H et 3H, il faut réussir à développer un jeu qui sache ne pas être trop répétitif et surtout avec une durée de vie à minima correcte. Implémentation : parties aléatoires, niveaux de difficultés et un système de score. Jouer sur la fréquence d’apparition des ennemies, la vie du joueur, les zones d’apparition des ennemies, les dégâts des ennemies... Objectif : Replay value via l’amélioration continue du joueur.  =>Pour pouvoir écouter d’autres choses en parallèle. En effet, nos futurs utilisateurs ont exprimé le besoin d'écouter d’autres choses (musique, chat vocal…) en parallèle du jeu. Implémentation : Réglage précis du volume, monter/baisser le niveau de la musique et des bruitages depuis le jeu. Objectif : Personnalisation et adaptabilité de l’interface.  => Les utilisateurs ont souvent des habitudes de commandes antérieur au jeu, l’idée et donc de ne pas changer ces habitudes. Implémentation : Bouton de sélection des commandes de jeu. Objectif : Permettre la personnalisation des commandes de jeu (choix possible entre les flèches et ZQSD), pour une meilleure flexibilité vers les différents niveaux d’expérience de nos utilisateurs.  => Permettre le choix de la langue (Le besoin se fait ressentir en Français/Anglais). Implémentation : Tableau de sélection entre les langues disponibles, affiché au lancement du jeu pour favoriser le guidage de l’utilisateur et éviter de le perdre dans les menus, pour une fonction aussi essentielle. Objectif : Personnalisation de l’interface et flexibilité envers tous types d’utilisateurs, indépendamment de leur langue natale.  => Nous pensions permettre le choix homme/femme du personnage : mais nous nous sommes rendu compte que cela était la dernière des priorités pour nos répondant, concernant la personnalisation de l’expérience de jeu.  =>Soigner les effets sonores, musique et bruitages. Implémentation : Préparer plusieurs bandes son en fonction de la situation (Menu, en jeu, victoire/défaite de fin de partie…) et pour appuyer les informations à texte à l’écran (en cas de victoire ou défaite par exemple). Objectif : L’UX et le guidage de l’utilisateur, pour que ce dernier comprenne rapidement et efficacement dans quelle zone du programme il se situe, sans doute possible sur l’information transmise, tout en minimisant la charge mentale.  => Apporter un soin particulier aux couleurs, à la forme des boutons, leur intégration dans les menus et sur l'écran ou encore le format des titres. Implémentation : penser par exemple aux feedbacks immédiats, lors de chaque interaction de l’utilisateur avec le système, comme la surbrillance des boutons au survole de la souris. Objectif : Favoriser la lisibilité des informations au possible, ainsi que le guidage de l’utilisateur dans les menus, de sorte que la personnalisation reste un point positif et non un fardeau.  => Adapter notre jeu aux lancements parallèles de logiciel, tel que Discord pour le chat ou Spotify pour la musique. Implémentation : Faire en sorte que l'app ne prenne pas tout l'écran, avec une fenêtre de taille restreinte. Le réglage du volume participe également à la réussite de cet objectif. Objectif : Adaptabilité à un maximum de situations possibles d’utilisation.  => Créer et soigner le lore du jeu, sa présentation et son intégration. Pensez à un bouton skip pour s’adapter à tout type d’utilisateur. Implémentation : Capsule texte ou vidéo optionnelle, pour introduire ou poursuivre le récit de l’histoire, au début, pendant et à la fin du jeu. Objectif : Améliorer l’UX (expérience utilisateur) sans détériorer le guidage.  => Soigner les textures, les jonctions et fonds d'écran en jeu. Implémentation : Sprite des textures (sol…) avec éventuellement plusieurs sorte/couleurs de platforms. Objectif : Améliorer l’UX et le guidage du joueur dans le jeu via une compréhension rapide et claire de ce qui est bon pour lui et de ce qui est mauvais.  La répartition des tâches s’est alors faite comme ci-dessous :   |  |  |  |  | | --- | --- | --- | --- | | Qui ? | Raphaël et Solenne | Raphaël | Yiwen | | Quoi ? | - Recherches d’images, de Sprites Sheets et détourage  - Commentaires du code  - Réflexion algorithmique face aux problèmes  - Optimisation et fluidification du code | - Réflexion sur l’histoire du jeu  - Création du menu principal  - Création du menu règles  - Création des crédits  - Création des fonctions d’optimisation (bouton et texte)  - Animation et apparition des ennemis  - Insertion et travail de la bande son et bruitages  - Gestion des modules *time* et *mixer*  - Création du game-over  - Création du menu pause  - Gestion sous forme de *class*  - Animation globe et personnage  - Création du niveau de jeu (plates-formes, blocs)  - Travail sur les déplacements du personnage  - Gestion des collisions  - Gestion de la vie  -Primo-écran, choix de la langue | - Changement de la langue |  1. **Backend**   Dans un premier temps, nous avons pu expérimenter, ce qu’offrait la bibliothèque Pygame et nous avons ainsi pu à partir de là, avorter certaines idées pour notre jeu, ou à contrario, trouver de nouvelles idées avec toutes ces nouvelles possibilités.  Nous avons d’abord créé une première version de notre jeu, nous permettant ainsi de nous expérimenter un peu avec ce nouvel outil qu’est Pygame. Malheureusement, nous avons fini par laisser de côté cette première version (très peu aboutie), car les lignes de codes commençaient à se faire nombreuses et donner une impression de désordre dans lequel il devenait difficile de naviguer. Nous avons alors repris notre projet sur la base de la programmation orientée objet (POO) et surtout les « sprites », que l’on peut aussi appeler entités pour cette dernière (cela désignant les globes, les ennemis, les personnages… etc, tout ce qui peut finalement être source d’interaction visuel à l’écran.)  Après un travail acharné, nous avons aboutis à un certain nombre de class et fonctions. Pour ne parler que des plus importantes, il y eu création du menu dans son entièreté (Des sous menus jusqu’aux boutons, sous forme de méthode de class) comprenant les méthodes règles, bouton, option et texte. Ceux-ci permettent notamment une meilleure organisation de notre jeu, tout en lui donnant une interface de lancement et les principales informations qui doivent être connues du joueur.  La partie option permet, elle, une meilleur expérience jeu, avec une personnalisation de notre partie, que ce soit les niveaux de jeu (difficulté), l’activation, ou non, des musiques et sons intégrés au jeu ou encore la modification de la langue et des commandes de jeux. Nous reparlerons de l’intégration de ces options plus tard dans le rapport, notamment concernant leur aspect ergonomique pour notre IHM, dans la partie front-end non visuelle.  La fonction la plus représentative de cette partie reste la fonction règles, accessible depuis le menu principale, qui propose plusieurs pages sur lesquelles nous pouvons naviguer à l’aide de boutons intégrés à celle-ci (cf capture d’écran jointe). Celle-ci pouvait en effet se faire sur une unique page, mais le but était de pouvoir placer autant d’éléments que nécessaire sans limite de place. La persévérance dans cette voie nous a permis de nous essayer à bon nombre de méthodes pour un même rendu. Cela nous a montré qu’il existait une infinité de méthodes possibles pour une même tâche, mais que certaines avaient des avantages sur d’autres, dépendant des situations et des besoins, notamment en termes d’économie de ligne de code. En effet, nous avons pu, dans un premier temps, imbriquer plusieurs boucles « while » les unes dans les autres, les pages les plus lointaines étant les boucles les plus profondes. Mais cette solution m’a causé beaucoup de problème, notamment concernant les économies de lignes, mais surtout dans l’utilisation de boutons interactifs, me forçant à les rappeler dans chaque boucle, tout comme la condition « pygame.QUIT » (fermeture de la fenêtre) essentielle à chaque boucle, sans quoi le programme se fige et ne fonctionne plus.  *Page 3 des règles du jeu*  Boutons d’interaction    Les mises à jour de la fenêtre était un autre frein à cette méthode, que nous avons finalement choisi de rapidement abandonner pour cette utilisation. Préférant la garder pour des boucles moins communes telles que les différents menus présents sur la page d’accueil, les pages des règles faisant partie du même ensemble, je me suis alors tourné vers l’idée d’un compteur. Le principe est simple, chaque page correspond à un numéro et l’activation des flèches de navigations fait monter ou diminuer d’un le compteur. Cette mise en place ne fut toutefois pas si simple et de loin, car une fois passés les nombreux problèmes de boutons inactifs, invisibles ou totalement absents, l’oubli d’actualisation ou le « referanced before assigment » un autre problème arriva, celui des ralentissements. En effet, rafraichir la totalité de la fenêtre en continu (lignes de texte particulièrement) n’était pas de tout repos pour Python, avec lequel l’utilisation des boutons devenait compliquée. Ainsi, l’utilisation de booléen m’a permis de ne faire apparaitre qu’une fois le texte par page nouvellement affiché et de ne mettre à jour que les boutons. Ce principe de booléens permet, en quelque sorte, de tenir au courant le programme de notre avancé ainsi que de nos intentions. Ici, elles permettent simplement d’éviter des actualisations inutiles du texte. Mais celles-ci se sont avérées plus d’une fois utile tout au long du programme, permettant aux fonctions de communiquer entre elles et de s’informer mutuellement sur leurs avancées pour savoir si nous remplissons les conditions nécessaires ou non pour telle ou telle activation. En plus de soulager notre unique Thread, elles permettent à notre utilisateur de bénéficier d’un temps de chargement réduit, améliorant de fait l’Ergonomie au globale et les interactions entre l’utilisateur et l’IHM, qui serait d’une efficacité réduite si son temps de réaction l’était aussi.  *Page 4 des règles du jeu (dernière page)*  Une seconde fonction, ou plutôt class, qui a synthétisée l’apport progressif de connaissances au fur et à mesure de l’avancement du projet, est la class ennemis. Bien que celle-ci reprenne certaines caractéristiques de la class personnage, tel que la gravité ou la gestion de la physique, elle est, et ce pour bon nombre de points, différente de la class personnage. D’une part, la gestion des évènements liés à la class, tels que l’avancée, sa vitesse et sa direction, ne sont plus gérés que part le module Random. En effet, le but était ici de pouvoir nous même jouer au jeu sans avoir connaissance de tout, tel que la position des ennemis, leur sens…etc. La gestion du hasard constitue donc un pilier fondamental de cette class, au même titre que le module temps. Le temps est le second moteur de cette class, puisqu’il permet de déterminer la dernière fois qu’est apparue un ennemi (soit la durée séparant deux ennemis) ou encore le temps qu’un ennemi reste en vie. Tout ceci étant constitué par l’unification de ces deux modules. Concrètement, On invoque deux horloges, une mise à jour en permanence et l’autre mise uniquement à jour une fois l’action que l’on surveille finie. Dans le cas de l’apparition d’ennemis par exemple, la seconde horloge n’est mise à jour qu’à chaque apparition. Une variable détermine ensuite aléatoirement quelle durée devra séparer ces deux horloges, soit le temps entre cette apparition et la prochaine d’un nouvel ennemi. On procède de façon identique avec la durée de vie en surveillant cette fois le temps de marche de l’ennemie. La position est de même définie aléatoirement entre la position du personnage (300 pixels plus loin pour éviter que quelque chose ne lui apparaisse dessus) et un peu plus loin que le bord de la fenêtre pour que certains arrivent par surprise. Toutefois, il existe une certaine variante avec les ennemis allant vers la gauche, qui apparaissent de fait, derrière le personnage et non devant. Pour cela on surveille la variable aléatoire qui donne son sens à l’ennemi, pour savoir quel intervalle de coordonnées peut servir au tirage au sort. Enfin, des améliorations notables ont été effectuées, notamment au niveau de la gravité, pour éviter les tremblements, de sorte que celle-ci diminue après le premier contact avec le sol (voir devient absente pour les ennemis immobiles). Dans la même ligné, j’ai fait en sorte que les dégâts subit par la collision avec les ennemis ne soit possible que si ceux-ci avaient fini d’apparaitre, pour éviter de rendre le jeu injuste et de laisser au moins au joueur le temps de le voir venir. Enfin, la création d’une image vide aux dimensions de l’ennemi, permet de le laisser tomber sur une plateforme sans que l’on ne le voie, pour ne le faire apparaitre qu’une fois posé sur celle-ci et donner, de fait, l’impression au joueur que celui-ci sort du sol. Il est notable de souligner que ces interactions entre la fonction jeu et la fonction ennemie se fait, entre autres, grâce à des booléens. C’est par exemple le cas pour vérifier que l’ennemi est posé sur sa plateforme ou encore qu’il ait fini d’apparaitre et qu’il peut infliger des dégâts. Ces booléens servent également à éviter la répétition d’un effet sonore pour une action qui perdure, pour ne l’activer uniquement en cas de nouvelle action, comme c’est le cas pour les collisions avec les ennemis. En bref, j’ai pu programmer et mettre au point de nombreuses fonctions, mais les deux précédemment citées fournissent les enseignements les plus complets et diversifiés, et m’ont permis, de par les problèmes qu’elles ont posés, de mieux cerner de la logique à user en algorithmique et en programmation.  *Marcheur noir immobile*  *Marcheur noir courant vers la droite*  *Marcheur noir courant à gauche*  Amélioration de la gestion des ennemies et leur despawn pour une optimisation et moins de lag, sans dégrader l’expérience du joueur, bien au contraire :  Si les ennemies tombent dans le sol avant même être apparu, on les supprime et on en crée un nouveau en remplacement. Un ennemi qui tombe dans le vide, despawn, plutôt que de continuer à s’actualiser hors de l’écran.  Peaufiner le backend a ainsi permit une meilleure entrée en matière dans le jeu, grâce notamment à la programmation du menu et ses nombreux boutons interactifs, mais aussi grâce aux décors choisi et aux musiques et sons que l’on a pu remanier, comme c’est le cas pour le menu justement. Cette entrée en matière permet notamment de prendre connaissance des principales règles et de l’histoire qui précède le jeu, ainsi que de certaines sources présentes dans le jeu, comme les musiques ou les décors, cités dans les crédits.  La création des fonctions bouton et texte ont quant à elles permis de faciliter l’utilisation d’écrits et de boutons au sein du jeu pour les membres du groupe, tous deux essentiels à l’ergonomie du jeu, pour la cohérence des textes (taille, typographie…). Ces fonctions permirent également d’améliorer et surtout réduire le nombre de lignes utiliser dans le programme, permettant de le parcourir plus aisément et de faciliter l’utilisation de ces outils, indispensable dans notre quête d’une bonne IHM.  Enfin, l’ajout des ennemis a permis de donner un enjeu au jeu, un principe de tension constante et de vigilance pour éviter de perdre plus de vie que l’on en perd déjà (replay value et challenge demandés par les utilisateurs). On insert ainsi au jeu une autre source de réels mouvements que le personnage principal que l’on incarne. L’amélioration constante et mutuelle de la class jeu, et plus particulièrement « update », « event » ou encore « nouveau », en font des éléments centraux qui permettent une connexion entre les fonctions et leurs coordinations, mais qui n’en font pas des éléments individuels à part entière.  Fonction d’optimisation à la frontière avec l’optimisation ergonomique de l’IHM : Fonction boite noir de Pygame : Pygame.time.delay(T-MS)  Remplacé par Jeu.delais, qui permet de garder notre thread actif pendant le délai imposé. Cela permet notamment à notre programme et à la fenêtre de continuer à réagir aux interactions utilisateur, notamment en ce qui concerne la fermeture de la fenêtre (la précédente commande de delay de pygame donnait une information erronée à l’utilisateur en cas de tentative d’interaction pendant l’utilisation de cette fonction : « le programme ne répond pas »)  Tentative Thread implémentation (expliquer les problèmes de réadapter un programme mono-Thread à un multi-Thread) + Multiprocessing(erreur non résolut et une perte de temps déjà massive sur le sujet, avec des résultats médiocre )  Réalisation d’un Niveau 2, pourquoi (replay value), comment ? modifier le programme qui accueillait à la base qu’un seul niveau dans sa V1  Création d’un point EXE en fin de conception pour pouvoir être ouvert et utilisé par tous les niveaux d’utilisateur (programmeur ou non) + Facilité d’usage, rien d’autre à télécharger que le fichier (adaptabilité, flexibilité et compatibilité)  Bien qu’une première vague d’amélioration ait été effectuée, il est encore possible d’améliorer notre jeu notamment avec, par exemple :  -Un nouveau niveau à part entière pour la Fabrique  -Faire du survivant croisé durant le jeu, une intervention plus pertinente, avec une quête par exemple ou des objets à collecter, lui donnant un pouvoir plus crucial dans l’avancée du jeu.  -Une meilleure personnalisation du jeu dans les options, avec des paramétrages de difficultés plus précise, modifiant par exemple la durée de vie des ennemis ou leur taux d’apparition.  -Intégration d’une vie pour les ennemis, rendant possible le fait de les tuer, moyennant l’intégration de collectable de type arme par exemple.  -Enfin, il est possible de trouver d’autre fonction améliorant la création et l’ergonomie du programme, sur le principe de la fonctions bouton ou texte.  Cependant l’essentiel est là et répond aux attentes de nos utilisateurs.  Finalement, à la manière des TPE, ce projet m’a rappelé que le travail en groupe impose certaines contraintes, mais propose des possibilités de création très vaste. Les idées de chacun font la force du jeu et les forces de chacun permettent au jeu de bénéficier d’une meilleure conception, certains maitrisant certains domaines mieux que d’autre et inversement. Se plonger à plusieurs dans la création d’un bien commun est passionnant et permet d’aboutir à un bien meilleur résultat qu’en étant seul.  A terme, il est envisageable de passer notre programme et ses fichiers en exécutable pour faciliter la diffusion de celui-ci. On pourrait imaginer un droit de diffusion libre, mais non commerciale à l’image de la licences Creative Commons « NC ».   1. **Front-end** 2. **Visuel**   Lorsque nous avons conçu le jeu, nous l'avons doté de la possibilité de changer la difficulté du jeu, de régler le volume, de personnaliser les commandes du jeu et de sélectionner la langue en fonction des réponses au questionnaire que nous avons recueillies.    Sur la base des réponses au questionnaire, nous nous concentrons également sur les points suivants.    a)Visuel  Dans nos projets, nous faisons beaucoup de travail pour bricoler le front-end. En raison du thème de notre jeu, nous avons choisi un ton relativement sombre.  Je vais vous les présenter un par un, en commençant par notre écran initial.  Pour faciliter la tâche des joueurs parlant des langues différentes, nous avons conçu le jeu de telle sorte que lorsque le joueur ouvre le jeu, il peut sélectionner une autre langue.  Une image contenant texte, extérieur, nuages, jour  Description générée automatiquement  Une image contenant texte, extérieur, signe  Description générée automatiquement    Nous avons utilisé la disposition verticale classique du jeu dans l'écran initial, c'est-à-dire que tous les boutons sont placés sur la même ligne verticale.  Pour des raisons d'ergonomie, les boutons changent de couleur lorsque l'on passe la souris dessus, afin que l'utilisateur ait un retour sur son comportement.  Une image contenant texte, extérieur, passage, jour  Description générée automatiquement  Une image contenant texte, extérieur, signe  Description générée automatiquement  Dans l'écran des règles, les utilisateurs peuvent naviguer à travers les 4 pages de règles en cliquant sur les boutons avant et arrière. En même temps, nous avons ajouté des images pour rendre l'interface plus facile à lire.  Une image contenant texte  Description générée automatiquement      Lorsque nous changeons les touches d'action dans le jeu, l'écran des règles change également avec notre choix d'images.      De cette manière, l'utilisateur est directement informé de la sélection de la touche qu'il a effectuée.  Comme nous tenons compte des préférences des différents utilisateurs dans le jeu, nous utilisons les deux moyens les plus classiques de changer de direction dans le jeu, l'un étant la touche ZSQD et l'autre la flèche. Nous avons choisi d'utiliser la barre d'espace pour les sauts.  Une image contenant texte  Description générée automatiquement  Lorsque nous atteignons la dernière page des règles, nous pouvons sélectionner le bouton de sortie pour revenir à l'écran du menu.  Sur la page de sélection, on peut choisir la difficulté du jeu, le niveau sonore, la langue du jeu et les boutons à utiliser pour jouer en cliquant sur les différents boutons.    Comme la plupart des gens choisissent le Réglage précis du volume, nous avons créé une barre de progression que l'on peut faire glisser pour contrôler le volume.      Il convient de mentionner que nous utilisons deux méthodes différentes pour régler le volume, nous pouvons cliquer sur le bouton pour activer ou désactiver la musique, et nous pouvons également contrôler le volume en faisant glisser la barre de progression avec un retour d'information en temps opportun, pour la barre de progression du volume, nous utilisons le design le plus classique et utilisons la valeur 0-100 pour indiquer le niveau de volume, de sorte que l'utilisateur aura une expérience plus confortable.  Dans l'écran des credits, nous pouvons voir une brève description de la conception de ce jeu    Lorsque nous cliquons sur le bouton de démarrage, nous avons conçu une animation avec des effets sonores et le bouton de démarrage est présenté seul à l'écran pour améliorer l'expérience de l'utilisateur.  Une image contenant texte, extérieur  Description générée automatiquement  Voici l'écran lorsque le jeu est en cours et que le joueur doit reconstituer le sang qui diminue avec le temps en utilisant le petit globe, tandis que le joueur doit faire attention à éviter les pièges et les hommes en noir dans le jeu.  Dans le jeu, nous avons également conçu les effets d'animation.  Une image contenant texte, extérieur, vapeur, fumée  Description générée automatiquement    Nous avons également ajouté des dialogues au jeu.  Les joueurs seront invités, au cours de la conversation, à faire attention à éviter l'homme en noir.    Nous avons conçu deux niveaux dans le jeu, le premier est plus simple que le second et ils sont également très différents en apparence.  Une image contenant texte, intérieur, cuisinant, gril  Description générée automatiquement  Pendant que le jeu est en cours, le joueur peut appuyer sur le bouton de pause pour redémarrer, retourner au menu, démarrer ou modifier les paramètres du jeu (sauf que vous ne pouvez pas modifier la difficulté d'un jeu en cours).  Une image contenant texte, extérieur  Description générée automatiquement      Si le joueur échoue le défi, il recevra un rappel lui indiquant qu'il n'a pas réussi à sauver la planète.  Si le joueur réussit le défi, il recevra un écran indiquant votre réussite.    Une image contenant texte, crépuscule  Description générée automatiquement     1. **Non visuel**   **Musique (Game-over ;Win) Bruitage(Bruitage pression bouton…) : Guidage implicite**  Nous avons pré-écrit les versions française et anglaise et affichons la langue en fonction de la sélection du bouton de langue par le joueur.    Sur la base des réponses à notre questionnaire, nous avons donc choisi deux langues, l'anglais et le français.  Une image contenant texte, équipement électronique, capture d’écran, ordinateur  Description générée automatiquement  Dans nos jeux, nous avons également conçu beaucoup de musique et d'effets sonores, qui apparaissent lorsque le joueur clique sur un bouton ou déclenche une scène dans le jeu.  Pour donner une représentation visuelle plus claire de la façon dont le jeu se déroulera lorsque nous effectuerons différentes actions, nous avons créé un diagramme de transition d'état pour mieux le montrer.    L’évaluation de l'ergonomie et conclusion :  Notre interface est très facile à apprendre et à utiliser, car nous avons des conseils très clairs et nous tenons compte des habitudes des différentes personnes qui jouent au jeu. L'interface est très efficace car elle permet au joueur d'accomplir des tâches de manière efficace, il est facile de se rappeler comment utiliser le jeu car nous disposons d'indications sur les règles, il est facile de revenir au menu si nous démarrons accidentellement une partie ou si nous modifions les paramètres du jeu, et le joueur aura un haut niveau de satisfaction car nous avons pris en compte ses besoins, et l'interface est réactive car les boutons changent de couleur lorsque le joueur clique dessus, et le joueur peut faire glisser la barre de progression pour modifier le volume. |