

Documenation technique

S1.01 - Implémentation d'un besoin client

Monster Hunter : New World

Sommaire

Table des matières

[I. Description succincte du jeu et de ses fonctionnalités 2](#_Toc89470143)

[II. Description justifiée de la structure du programme 3](#_Toc89470144)

# Description succincte du jeu et de ses fonctionnalités

Le jeu Monster Hunter : New World est un concept original de jeu vidéo dans lequel le joueur est amené à prendre le contrôle d’un personnage vivant dans un univers fantasy et dont le métier est de chasser des monstres gigantesques aux aspects préhistoriques (dinosaures, dragons...). Afin de progresser dans l’histoire et pouvoir chasser des monstres de plus en plus puissants, le joueur doit planifier avec soin ses chasses, s’entrainer et récolter des pièces d’équipement lui permettant de forger de nouvelles armes et armures. (L’équipe de gamedesign envisageait initialement l’ajout d’une composante MMO-pvp centrée sur un principe de contrôle de territoires mais cette idée n’a finalement pas été retenue).

Le studio de développement vidéo-ludique de l’IUT souhaite que le développement du jeu prenne la forme d’un jeu textuel en console. Il devra être réalisé intégralement en freePascal en utilisant une bibliothèque IHM fourni par le studio. Le studio attend de l’équipe de développement qu’elle produise une première version du jeu comportant les éléments suivants :

* **Un menu initial** qui devra proposer à l’utilisateur plusieurs choix dont ceux de commencer une nouvelle partie et de quitter.
* **Un écran de création de personnage** dans lequel le joueur pourra personnaliser son personnage : Nom, Sexe, Taille...

Une fois la partie lancée, le joueur aura alors accès à une ville dans laquelle il pourra préparer ses différentes chasses. Le personnage contrôlé par le joueur devra disposer :

* **D’un inventaire** comptant des objets variés (potions, bombes...)
* **D’un équipement** composé d’une arme et d’une armure divisée en 5 parties : casque, torse, jambières, bottes et gants. La ville devra comporter les lieux suivants :
* **Une chambre** dans laquelle le joueur pourra se reposer et gérer son équipement (armes et armures).
* **Une forge** dans laquelle le joueur pourra faire fabriquer de nouvelles armes et armures à l’aide de parties récupérées sur les monstres qu’il a déjà chassés.
* **Un marchand** chez lequel il pourra acheter et vendre des objets (sauf armes et armures).
* **Une cantine** dans laquelle il pourra prendre un repas lui fournissant un bonus pour la prochaine chasse.

Une fois sa chasse préparée, le joueur pourra décider d’aller affronter un monstre. Durant l’affrontement qui aura lieu au tour par tour,

* L’équipement (arme et armure) du joueur devra impacter sur les dégâts infligés et reçus par le joueur.
* Le joueur devra pouvoir utiliser les objets de son inventaire.
* Le joueur devra pouvoir récupérer des parties du monstre une fois celui-ci vaincu.

# Description justifiée de la structure du programme

Pour répondre aux attentes du studio, nous avons décidés de structurer le programme en plusieurs partie. Le programme est donc séparé en 12 parties, dont 11 unités :

* logique
* menu
* outils
* uniteforge
* uniteCantine
* uniteChambre
* uniteMarchand
* uniteChasse
* controle
* personnage
* GestionEcran

et le programme de lancement du jeu :

* Monster

Le programme principal, Monster, est relié à toutes les unités. En lançant celui-ci, on exécute la fonction start, fonction faisant partie de l’unité logique, qui permet de commencer à jouer. Nous avons décidé de ne mettre qu’une fonction dans celui-ci, puisque nous souhaitons ne pas en utiliser d’autre dans le programme principal.

L’unité Logique, sert à réaliser des calculs, charger les données des objets, des équipements et des monstres. Elle comporte également la partie logique des menus, c’est-à-dire, lorsqu’un joueur doit effectuer un choix tel que devoir choisir entre utiliser le lit ou l’armoire dans la chambre, le joueur effectue son choix dans la partie graphique du jeu. Son choix est donc pris en compte grâce à cette unité. Dans ce cas-ci, si le joueur décide d’utiliser l’armoire, dans l’unité logique se trouve une procédure chambre, dans laquelle est lancé la fonction menuChambre qui retourne le choix du joueur. Puisque le joueur a décidé d’utiliser l’armoire, la procédure chambre, exécute donc le code qui permet d’utiliser l’armoire. Nous avons décidé de séparer les choix graphiques, des choix logiques pour notre confort d’utilisation.

L’unité menu regroupe la plupart fonctions et procédure principale que l’on utilise pour réaliser le visuel de notre programme. Dans celle-ci, par exemple, nous pouvons y retrouver la fonction menuForge, qui permet de lancer la forge et ses choix. Nous avons décidé de ne pas mettre toutes les fonctions nécessaires à l’affichage dans cette unité, puisque nous avons choisis de créer d’autres unités, dans lesquels nous regrouperons les parties logiques et affichage, de certaines parties conséquentes du jeu.

L’unité outils, regroupe toutes les procédures nécessaires à la lecture de fichier JSON et ASCII, ainsi qu’une procédure qui nous permet de réaliser une animation dans la ville. Nous avons décider d’utiliser des fichiers de type JSON, puisque grâce à eux, nous pouvons y insérer l’ensemble des données d’un monstre, d’un objet, d’un équipement et ensuite en convertissant ces données pour qu’elle soit lisible et utilisable par le langage freePascal. Nous avons créé un type de fichier nommé ascii, dans lequel nous dessinons des formes, des menus, des personnages et des monstres pour qu’il n’y ait aucun problème de lecture de ceux-ci. L’unité outils a été créé dans le but de réaliser une sorte de boite à outils, dans laquelle nous utilisons des fonctions nécessaires telle que celle qui nous permette de lire les données inscrites dans les fichiers JSON et d’autres, que l’on peut qualifier de fonctions bonus qui nous permette de faire des animations et de l’art.

L’unité uniteforge, est composée d’une unique fonction, getCraftResult, qui retourne à l’entrée de deux types item (deux objets), un équipement, si sa fabrication en fonction de la quantité et des objets donnés, est possible. Nous avons décidé de créer cette unité, puisque le code est plutôt long, pour une seule fonction. La deuxième raison est que si le code rencontre un problème au niveau de la forge, nous n’ayons à chercher dans toutes les autres unités dans laquelle nous avons réaliser des fonctions/procédures qui font références à des parties de la forge, pour le résoudre.

L’unité uniteCantine comporte la fonction qui permet d’afficher le menu de la cantine, ainsi que les avantages donnés par la consommation de plats. Les plats peuvent vous donnez des avantages de dégâts, de défense et de vie. Nous avons décidé de créer cette unité, pour avoir une unité unique à la cantine et ses consommables.

L’unité uniteChambre est composée de deux fonctions, une pour afficher et gérer le menu et l’autre pour afficher et gérer l’armoire. L’unité comporte également une procédure qui affiche le lit. Cette unité a été créé pour la consacrer à tout ce qui est possible de faire dans la chambre. C’est-à-dire, se reposer, gérer ses équipements, mettre des objets dans son armoire.

L’unité uniteMarchand comporte toutes les fonctions et procédures liées à l’achat et la vente d’objets à un marchand. Par exemple, pour l’achat, un certain nombre d’objets sont mis en vente, excepté les armes et équipements, qui eux doivent être forgés pour être obtenu. La vente fonctionne avec le même système, mais tous les objets sont vendables hormis les équipements et armes. Nous avons décidé de créer cette unité, parce que le code est très long et qu’il n’avait rien à voir avec les autres unités. Nous avons donc décidé l’ajouter à une nouvelle.

L’unité uniteChasse comporte toutes les fonctions et procédures lié à l’événement qu’est, la chasse. C’est-à-dire, lorsque l’on décide d’aller chasser, nous avons possibilité de choisir quel monstre nous souhaitons rencontrer. Le combat doit s’effectuer en tour par tour, les dégâts du monstre doivent s’effectuer de manière aléatoire, nous devons actualiser les points de vie ainsi que les barres de vie. Nous devons donner la possibilité au joueur de fuir, d’utiliser son inventaire lors du combat. Enfin, nous devons donner une récompense au joueur après son combat effectué et gagné. Nous avons décidé de créer cette unité, puisqu’elle est totalement différente des autres et qu’il s’agit d’une des composantes principales du jeu.

L’unité contrôle nous permet de nous déplacer à l’aide des flèches directionnelles. Cet unité gère les entrés d’un clavier et doit être appelé dans les autres unités si l’on souhaite réaliser un moyen de se déplacer grâce à des touches. Nous avons donc créé une unité réservée.

L’unité personnage comporte toutes les fonctions en rapport aux personnages. Par exemple, son inventaire, ses dégâts, ses informations, etc… Nous avons créé cette unité et nous avons trouvé qu’il était indispensable de ne pas mettre les codes liés au personnage dans d’autres unités.

L’unité GestionEcran est une unité fournie qui permet d’afficher du graphique sur l’invite de commande. Cette unité est indispensable pour l’affichage, le coloriage, etc…