ROSALES Philippe
AUGIER Lucie

IMAC1

RAPPORT DE PROJET DE PROGRAMMATION “POKIMAC”

Mode d’utilisation :

But du jeu :

L’objectif du joueur est de découvrir et capturer toutes les espèces présentes sur “Pokemon Island” avec l’aide d’une joyeuse équipe de pokémons et d’un inventaire d’objets qu’il se constitue au fil de ses rencontres.

Initialisation de la partie :

Le jeu commence avec un message de bienvenue de la part du professeur Okitac. Après avoir entré son prénom, le joueur lit ses explications dans le terminal et se voit remettre un Pikachu comme premier compagnon de route. Le professeur explique aussi que le joueur aura la possibilité de consulter son inventaire et le Pokédex durant la partie. L’inventaire est constitué de quatre objets : les potions, les super potions, les pokéballs et les superballs. On détaillera leurs effets dans la suite du rapport. La taille de la carte est initialisée par défaut proportionnellement à la taille du terminal lors du lancement. L’aventure peut commencer !

Déroulement de la partie :

Le joueur est positionné sur la map, et on peut le repérer grâce au “#” sur le terminal. Il est mis en mouvement grâce aux flèches directionnelles. Les pokémons ne sont pas affichés sur la carte. La rencontre a lieu avec une certaine probabilité lors du déplacement du joueur. Elle est spécifiée par l’apparition d’une nouvelle fenêtre. Le joueur se voit proposer plusieurs options: attaquer le pokémon sauvage, changer d’allier pour le combat, accéder à son inventaire ou fuir le combat.

- Attaque : Le pokémon sauvage perd un certain nombre de points de vie (PV) en fonction de l’espèce de pokémon alliée du joueur. Le pokémon sauvage attaque à son tour et inflige un nombre de dégâts dépendant aussi de son espèce. Si le pokémon allié est mis KO, le joueur a le choix de changer de pokémon ou de tenter de s'enfuir..
- Pokémon : Le joueur accède à son équipe. Il voit les pokémons qu’il a déjà capturés et leurs PV. Si son pokémon allié dans le combat n’a presque plus de PV, il peut changer de pokémon en en sélectionnant un nouveau.
- Sac : le joueur accède à son inventaire. Le sac contient des pokéballs, des super balls, des potions et des super potions. Le joueur peut sélectionner un objet pour l’utiliser:

- **potion** : permet de redonner 20 PV au pokémon allié
- **super potion** : permet de redonner 50 PV au pokémon allié
- **pokeball et super ball** : permettent de capturer un pokémon. Si le joueur les utilise, il a une probabilité p1 ou p2 (en fonction du type de pokéball, avec p2 supérieure à p1) de capturer le pokémon. Si le pokémon est capturé, le pokédex du joueur est mis à jour. Le pokémon est ajouté à l'équipe, mais si elle est pleine, un message indique que son pokémon a été envoyé au PC.

- **Fuite** : Si le joueur choisit la fuite, il a une certaine probabilité de pouvoir s'échapper. Si elle se réalise, il sort du combat.

Si le joueur met K.O. le pokémon sauvage, il peut recevoir aléatoirement un objet pour son inventaire. Il poursuit ensuite sa quête.

Fin de partie :

La partie s'arrête sur une victoire si le joueur a réussi à capturer toutes les espèces de pokémons du jeu, ou bien sur une défaite s'il n'a plus aucun pokémon en vie dans son pokédex.

Probabilité de capture en fonction des pokeballs :
Un nombre aléatoire est choisi entre 0 et 100, la capture réussit s'il est supérieur à un certain seuil.
Ce seuil est la somme de 60 multiplié proportionnellement au nombre de PV du pokémon sauvage, ajoutés à une base fixe de 10. En fonction de l'indice de la pokéball (0 correspond à la pokéball de base, 1 la super ball) on soustrait à ce nombre hn multiple de 30, ce qui augmente les chances de capture.

Méthodes utilisées dans l'implémentation:

La méthode d'implémentation générale a été d'expérimenter les grosses fonctionnalités dans le main ou les fichiers cpp associés, puis de les séparer en fonctions et éventuellement de les déplacer dans un autre module. Pour ce faire, nous avons fait appel à plusieurs fonctionnalités et modules détaillés ci dessous.

La bibliothèque Ncurses :

Grâce aux connaissances de Philippe, nous avons pu utiliser cette bibliothèque pour gérer l'affichage graphique. La map est initialisée grâce au module Window et ses fonctionnalités. Il en est de même lors du déplacement du joueur. En appuyant sur les flèches directionnelles, on met à jour ses coordonnées. La création de l'interface utilisateur a également été facilitée par l'utilisation de cette

bibliothèque puisqu'elle permet de créer des fenêtres et des fenêtres dérivées ainsi que d'afficher des messages à l'endroit que l'on veut sur le terminal, relativement à des fenêtres ou non..

Les structures et les énumérations :

Le code repose aussi sur un bon nombre de structures imbriquées les unes dans les autres:

- Le joueur
- Les pokémons
- Le pokédex
- L'équipe
- La map
- L'inventaire

On utilise aussi des énumérations:

- type (le type de pokémon : eau, feu etc.)
- espèce (l'espèce de pokémon : Pikachu, Salamèche etc.)

Elles sont idéales pour la gestion des pokémons et du pokédex puisqu'elles permettent d'associer un type ou une espèce à un entier. Cet entier est ensuite plus facilement exploitable que si l'on conserve une chaîne de caractère.

Les aléatoires :

L'appel à srand(time(NULL)) et à la fonction rand() permet de coder des probabilités d'événements, ce qui contribue à dynamiser notre jeu et à le rendre plus fidèle à la réalité du vrai jeu Pokémon. Ainsi, la rencontre avec un pokémon sur la carte, sa capture ou encore la récupération d'un objet pour l'inventaire sont aléatoires, et apparaissent sous une forme différente à chaque partie. On a d'ailleurs trouvé amusant de coder tout ce côté probabiliste pour chaque action.

L'utilisation de nombreuses fonctions :

Là encore, l'expérience de Philippe permet de simplifier le code, puisqu'il a choisi de créer de nombreuses fonctions aux spécificités variées. Elles permettent d'une part une meilleure compréhension du code à la lecture et d'autre part une généralisation des actions.
Par exemple, les fonctions initialisées dans le fichier "fight" peuvent toutes être utilisées dans les deux sens du combat: le pokémon attaquant peut être l'allié ou bien l'ennemi. Du côté de l'interface utilisateur, les fonctions initWindow, menulist ou msgbox permettent de créer rapidement une fenêtre, un menu ou afficher un message sans avoir besoin de tout coder dans le main. C'est une économie de temps non négligeable.

Difficultés rencontrées :

ROSALES Philippe
AUGIER Lucie

IMAC1

Concernant le code en lui-même, nous avons fait face à plusieurs difficultés:

- Gérer les conflits string/char*/char const* : Il a fallu utiliser uniquement des pointeurs pour les noms, qui pointent vers le tableau des noms. Nous avons volontairement choisi de ne pas utiliser les string, pour s'entraîner à manipuler les char*, et par souci de cohérence, tout le code utilise uniquement des éléments du C.
- Trouver une manière de représenter et générer les pokémons qui soit la plus efficace pour pouvoir en ajouter dans le futur. Et gérer ceux actuellement présents d'une manière suffisamment optimisée et automatisée pour réduire l'utilisation de mémoire et de malloc, ainsi que pour faciliter la gestion des données.
- Choisir comment représenter le terrain en mémoire, comment le relier à l'affichage.
- Séparer les modifications de données des animations et affichages. Il était facile et tentant de faire des fonctions qui font les deux alors qu'il est très important de ne surtout pas les mélanger (sauf dans des fonctions regroupant volontairement des actions entières).
- Optimiser la gestion de la mémoire. Nous avons évité de consommer trop de mémoire en ne faisant pas de malloc pour les noms des pokémons, et en les générant à la volée lors d'un combat.
- Gérer les menus, notamment le scroll quand il y en a besoin. Faire un menu qui cycle sans faire plusieurs if imbriqués était un petit défi.
- Savoir dans quel fichier doit aller telle ou telle fonction qui touche des objets différents.
- Garder une cohérence en choisissant entre valeurs de retour et pointeurs, et s'assurer que les valeurs de retour aient un sens et ne soient pas juste là pour se lier à d'autres fonctions dans l'algorithme. Garder en tête que les fonctions doivent rester indépendantes.

De manière plus générale, la difficulté réside principalement dans l'hétérogénéité du groupe. Philippe a étudié plus longtemps l'informatique tandis que Lucie a plus de mal. La différence globale, parce qu'elle était combinée à une difficulté de travailler en même temps, a gêné Lucie dans sa participation au projet. Cette différence fait que Lucie a dû essayer de comprendre les subtilités du code plus avancé de Philippe et se familiariser avec ce qu'il avait fait pour pouvoir le réutiliser. Il a surtout été difficile de trouver des créneaux communs compte tenu des emplois du temps différents des deux groupes et des autres projets à gérer en parallèle, sans cela une bien meilleure collaboration aurait été possible..

Méthode de travail :

ROSALES Philippe
AUGIER Lucie

IMAC1

Nous avons d'abord cherché à visualiser le comportement du programme. Une fois les étapes bien comprises, nous l'avons séparé en parties distinctes :

- La carte, le joueur
- Le pokédex,
- Les combats
- ...

Nous avons ensuite étudié chaque partie pour les "décortiquer" en étapes :

- Créer et afficher une map
- Créer et déplacer un joueur dans cette map
- Gérer le pokédex
- Initialiser des pokémon
- Gérer l'équipe du joueur, y ajouter des pokémons
- Gérer un combat et chaque action faisable dedans (et au préalable pouvoir créer des menus)
- Gérer les fins de jeu

Nous avons alors prévisualisé les algorithmes et essais, puis nous les avons construits progressivement en partant des fonctionnalités fondamentales pour chaque partie. Le principe était de construire rapidement les éléments nécessaires au fonctionnement, ensuite nous ajoutions les fonctionnalités supplémentaires. A chaque étape d'avancement un commit était effectué sur git.

Améliorations possibles et problèmes connus :

La présence de hautes herbes sur la carte est possible, et fonctionne, le joueur peut s'y déplacer, son caractère remplace celui des hautes herbes lors de son passage, puis l'herbe revient quand il se déplace. La probabilité de rencontrer un pokémon augmente lorsque l'on marche dans l'herbe. L'ajout d'herbe a été désactivé à la création de la map car la génération aléatoire n'était pas optimale, il aurait fallu un peu plus de temps pour avoir un génération réellement homogène et visuellement jolie. Une utilisation des chaînes de Markov aurait été un rêve.

Ajouter des attaques différentes pour les pokémons serait un ajout très intéressant qui complèterait les fonctionnalités principales. Il serait facilement implémentable grâce aux fonctions de menu créées. Nous ne l'avons pas ajouté par manque de temps. Il en va de même pour les ascii art de pokémon, l'endroit prévu pour est présent dans les fenêtres de combat, il manque juste une fonction d'affichage donnant les sprite de face et de dos pour avoir des pokémons dans le terminal.

Pour le moment l'équilibrage du jeu est assez grossier, des ajustements plus fins seraient les bienvenus.

ROSALES Philippe
AUGIER Lucie

IMAC1

Il y a de temps en temps un crash de type “stack smashing”, extrêmement dur à isoler car il ne revient pas dans un cas précis de manière stable.