RAPPORT VEGADOUA

I. Introduction

Vegadoua, c'est quoi ?

Vegadoua c'est un jeu inspiré de l'univers Pokémon ! Nous incarnons un dresseur de petites créatures appelées vegamons, et notre objectif est de devenir le meilleur dresseur du jeu. Pour cela nous devons parcourir les différentes cartes, vaincre tous les dresseurs qui se mettent en travers de notre chemin pour arriver jusqu'à l'arène finale et battre le champion.

Le nom du jeu « Vegadoua » est un mixte entre l'endroit où se déroule le jeu, la Doua, et « vega », acronyme de nos prénoms.

Qu'est-ce qu'un vegamon ?

Les vegamons sont les petites créatures du jeu qu'il faut battre. Au total il y en a six, chacun a son propre nom, ses propres attaques, un certain nombre de points de vie, d'attaque, de défense, et une certaine probabilité d'esquiver une attaque adverse. Ils ont également un « type » comme eau ou feu par exemple, et, de ce fait, ils ont des faiblesses et des résistances contre certaines attaques.

Pour exemple, une attaque de type eau mettra beaucoup plus de dégâts à un vegamon de type feu, qu'à un vegamon de type plante. Il appartiendra au dresseur de trouver les faiblesses de chaque vegamon qu'il affronte.

Enfin, les vegamons ont des points d'expériences. Plus un vegamon est expérimenté, plus ses points de vie sont élevés, et plus ces attaques sont efficaces. Pour gagner des points d'expériences, il faut vaincre d'autres vegamons.

Description du problème posé

Nous avons notamment abordé dans notre programme la problématique du déplacement d'un personnage dans différentes cartes à l'aide de la souris, avec des combats, des interactions avec d'autres personnages et des quêtes à accomplir. Le but du jeu étant de vaincre le champion de l'arène finale. Nous avons également mis en place des musiques et une sauvegarde automatique.

Cahier des charges

Différentes fonctionnalités sont proposées au joueur. En effet, le personnage se déplace dans les différentes cartes en un clic, il peut alors rencontrer certains obstacles : maisons (à visiter), arènes de combat, personnages (PNJ) à qui parler ou contre qui se battre, des hautes herbes symbolisées par des trèfles. Lorsque le personnage se déplace sur des cases trèfles, il y a une certaine probabilité de se faire attaquer par des vegamons sauvages. En combat, on peut choisir ses attaques à l'aide de boutons, et celles-ci sont plus ou moins efficaces selon le type de l'adversaire (exemple : une attaque de type feu inflige plus de dégâts à un vegamons de type plante qu'à un vegamons de type eau). Après un combat, notre vegamon retrouve ses points de vie. Si le joueur perd le combat, il retourne à la maison du début de jeu afin que son vegamon se repose. S'il gagne le combat, le vegamon du joueur gagne des points d'expérience augmentant ses statistiques (attaque, défense…). Aussi, plus le joueur avance dans le jeu (sur la carte) plus les vegamons sauvages seront difficiles à battre.

Sur le menu principal, on peut : soit reset sa partie (on efface donc la sauvegarde), soit reprendre là où l'on s'était arrêté, soit cliquer sur le bouton scénario où les différentes parties du jeu sont

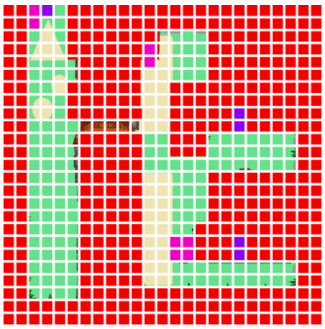
expliquées, soit activer ou non la musique, soit cliquer sur Vegadex. Ce bouton permet d'accéder à liste les différents vegamons et leurs caractéristiques.

II. Principe de l'algorithme

Principe de l'algorithme

Notre projet se décompose en deux parties. D'abord des classes dites techniques. La classe VariablesDeJeu regroupe toutes les variables du jeu : les tableaux caractéristiques des cartes, les textes des dialogues, les noms des musiques en fonction de la carte etc. Certaines de ses variables évoluent au cours de la partie et sont sauvegardées à l'aide de la classe sauvegarde dans un fichier csv. Par ailleurs, la classe Musiques permet de jouer les musiques du jeu et les classes Attaques et VEGAMONS concernent les informations des vegamons et leurs attaques. Enfin, la classe CJframe, initialisé au lancement du programme, permet de lancer toutes les fenêtres du jeu dans une seule.

Ensuite, nous avons des classes dites d'affichage. La classe Accueil est celle lancée au début du jeu, elle contient la méthode main et permet d'accéder aux autres fenêtres du jeu. La classe FenetreVegadex déclenche la classe FenetreVega et permet de visualiser la liste des vegamons et leurs attaques respectives. Ensuite, la classe Accueil permet de lancer la classe FenetreCarte qui correspond à la classe principale du jeu, là où le jeu se déroule majoritairement. Cette classe est particulièrement intéressante au niveau algorithmie. Elle permet d'afficher le personnage qui se déplace dans une carte. Nous avons décidé de quadriller les cartes comme ceci pour le déplacement du personnage :



Chaque case a une couleur qui correspond à une action (déplacement possible ou non, interactions avec pnj, changement de carte...). Et chaque couleur est représentée par un chiffre dans notre programme. Ainsi, il a fallu écrire une dizaine de tableaux en 2D de taille 25x25 avec différents chiffres. Le personnage se déplace dans la carte en cliquant à l'endroit où il souhaite se déplacer. Si c'est accessible (Pas un arbre ou un mur par exemple). Un algorithme de Pathfinding trouve un chemin pour accéder à la destination et immédiatement après, le personnage entame son déplacement. Lors de son parcours, le personnage peut déclencher plusieurs actions (Changement de carte, dialogue avec un dresseur, combat etc) inscrites dans les tableaux. L'algorithme de recherche de chemin est l'élément scientifique de notre projet. Après s'être renseignés sur les différents algorithmes de Pathfinding nous avons choisi l'algorithme A* (A étoile) qui est un bon compromis entre rapidité et efficacité. C'est un algorithme de recherche de chemin dans un graphe entre un nœud initial et un nœud final. Chaque case du tableau est associée à un nœud (classe Node). L'algorithme utilise une évaluation heuristique sur chaque nœud. C'est-à-dire qu'il évalue une sorte de probabilité (un coût minimal) qu'un nœud voisin permette de se rapprocher du nœud final. L'algorithme parcourt alors chaque nœud jusqu'à trouver un chemin satisfaisant. Une fois le chemin définit, celui-ci est stocké dans un tableau puis trié par ordre de parcours (déplacement de case voisine en case voisine). Cet algorithme A* fût assez fastidieux à mettre en place. Nous avons essayé de nombreuses solutions trouvées sur internet et c'est un combiné d'entre elles qui nous a permis d'arriver au résultat qui est selon nous très satisfaisant. Enfin, la classe FenetreCombat gère les combats entre les différents vegamons avec notamment une gestion du niveau des vegamons adversaires en fonction de l'avancement dans le jeu, une gestion des points de vie et une efficacité variable des attaques selon les types des vegamons.

Diagramme UML

Vous trouverez le diagramme UML en annexe.

III. Organisation générale et améliorations possibles

Comment nous sommes nous organisés pour réaliser ce jeu ?

Nous avons d'abord divisé le travail en quatre, chacun s'est occupé d'une classe en particulier, puis, au fur et à mesure, nous nous sommes réparti les tâches en fonction de nos aptitudes et des domaines dans lesquels nous étions le plus à l'aise.

Pour faciliter la mise en commun de notre travail et le partage de celui-ci, nous avons utilisé SVN durant une grande partie du projet. Toutefois, la version d'essai arrivant à expiration quelques jours avant la date de rendu final, nous avons été contraints de nous accommoder à une nouvelle application : GitHub (qui s'est avérée bien plus fonctionnelle).

Pour la rédaction de ce rapport, celle-ci a été effectuée au fur et à mesure du projet grâce à Teams.

Carnet de route, échéancier

Nous avions commencé par une semaine de réflexion pour savoir assez précisément ce que nous voulions faire et ne pas faire. Puis nous nous sommes réparti le travail afin de pouvoir travailler chacun de notre côté. Nous avons alors commencé par faire en parallèle, les fenêtres d'accueil (Gianni), la fenêtre de combat (Estelle & Adrien), l'algorithme de déplacement (Victor), les graphismes (Estelle), et les tableaux de valeurs (quadrillant les maps). Dans un second temps, nous nous sommes occupés de la fenêtre de carte (Gianni). Puis, nous avons fait le lien entre toutes les fenêtres. La mise en commun de toutes les parties s'est bien déroulée puisque nous avions tous réalisées nos parties en lien avec les autres. Dans un second temps, nous nous sommes attachés pendant plus d'un mois à résoudre les nombreux bugs que nous rencontrions, tout en continuant à apporter de nombreuses améliorations en développant notamment le scénario (Dialogues etc). Enfin, Victor a mis en place la musique et les sauvegardes automatiques.

Quels sont les problèmes que vous avez rencontrés lors de ce projet ?

Les problèmes ont été nombreux, en voici une liste non exhaustive :

- Le quadrillage des cartes est un des problèmes majeurs que nous avons rencontrés. Tout d'abord. Nous avons dû remplir les tableaux avec 12500 valeurs. (20 tableaux de taille 25*25) ce qui fût long et fastidieux. Ensuite, le problème avec ce quadrillage est que certaines actions étaient imprécises (par exemple certains pnj remplissent une demi-case au lieu d'une) et à cause de cela, à plusieurs endroits notre personnage marchait sur des pnj ou sur des petits bouts de tables. Pour rectifier cela, il a été nécessaire de photoshoper plusieurs éléments des maps. Enfin, les déplacements du joueur sont saccadés puisque la résolution est faible et le personnage se déplace de case en case.
- Nous avons aussi eu pas mal de problèmes d'affichages notamment pour les boutons de la fenêtre de combat.
- Bien ajuster les gains d'expériences des vegamons, la force des adversaires, équilibrer le niveau du jeu n'a pas été simple.

- Nous avons également fait face à de nombreux problèmes de timers qui fonctionnaient mal, notamment dans la mise en place de la classe FenetreCombat, ce qui a remis en cause plusieurs fois la structure de la classe.
- Nous avons également dû trouver des solutions pour le déplacement automatique du personnage lors du clic sur la map. En effet, nous avons examiné différentes méthodes sur internet afin d'en trouver une efficace, compréhensible et facile à adapter à notre jeu, et cela n'a pas été une tâche facile.

Quelles sont les forces du jeu ?

Le jeu étant inspiré de Pokémon, nous aurions simplement pu copier les images, le son et le scénario du jeu de base pour nous simplifier la vie mais nous avons décidé de tout créer nous-même : nous avons réalisé le son nous-même, ainsi que tous les graphismes, l'image de l'écran d'accueil, les maps, les personnages, les bâtiments, les vegamons… toute l'identité du jeu a été réalisée par nos soins.

Le gros point fort du jeu est la façon de se déplacer, grâce à la souris en un clic notre personnage se rend où on souhaite. La rapidité du déplacement a même été optimisée au fur et à mesure du projet.

La sauvegarde automatique du jeu est également un point appréciable du jeu, on peut revenir sur le jeu autant de fois que l'on veut sans perdre nos données, et nous pouvons reset la partie quand on le désire. Cela nous a en outre permis d'appréhender cette fonctionnalité de java.

Quelles sont les faiblesses du jeu ?

Quelques imprécisions sur les déplacements du personnage, le quadrillage des cartes n'est pas toujours très précis comme évoqué plus haut. Enfin, contrairement au vrai jeu, il y a peu de vegamons et assez peu de cartes, le jeu est donc relativement vite terminé (Environ une heure de jeu).

Quelles sont les améliorations que nous aurions pu apporter à Vegadoua ?

Un très grand nombre d'options supplémentaires auraient pu être ajoutées à notre jeu. En effet, le point positif (et le point négatif) de ce jeu est qu'il ne peut jamais réellement être terminé. Nous aurions pu ajouter toujours plus de vegamons, plus de cartes, plus de quêtes, plus de dresseurs, plus de musiques…

Voici une liste non exhaustive des améliorations envisagées mais non réalisées par manque de temps :

- Choix du niveau de difficulté du jeu,
- Possibilité de régler le volume et changer la musique,
- Lire des pistes audios pour les dialogues qui s'affichent actuellement uniquement sur l'écran
- Possibilité de capturer des vegamons, et de changer de vegamon
- Possibilité de ramasser des objets sur les cartes pour nous aider à battre des dresseurs ou des vegamons sauvages,

Implication de chaque membre du groupe

Nous considérons avoir été chacun très impliqués sur le projet et à la même hauteur, en témoigne la consistance du jeu, bien qu'aucun de nous ne soit particulièrement expérimenté.

Cependant, il est de rigueur d'ajouter une mention spéciale à Victor pour son apport de connaissance (sauvegarde, musique) et surtout pour l'algorithme de recherche d'itinéraire, point fort du jeu.

Nous avons donc attribué 29% à Victor, 24% à Gianni et Estelle, et 23% à Adrien.

IV. Conclusion

Nous avons consacré beaucoup d'heures à ce projet mais nous y avons surtout tous pris beaucoup de plaisir ! Ce jeu nous a beaucoup fait progresser en programmation et nous sommes très fiers du résultat final ! Résultat qui a pu être obtenu grâce à une bonne organisation et un travail d'équipe bien mené. SVN, puis GitHub auront été des supports primordiaux et leur maîtrise fut une clé du déroulé et de l'avancement du projet.

V. Bibliographie

Algorithme de recherche de chemin :

1. http://fr.wikipedia.org/wiki/Algorithme_de_Dijkstra
2. http://fr.wikipedia.org/wiki/Algorithme_A*
3. http://fr.wikipedia.org/wiki/Algorithme_de_parcours_en_profondeur
4. http://fr.wikipedia.org/wiki/Algorithme_de_parcours_en_largeur

Sites plus détailés pour la solutions retenues (A*) :

https://openclassrooms.com/forum/sujet/labyrinthe-connaissant-sortie-trouver-chemin)

https://complex-systems-ai.com/cours-lessons-theory/recherche-de-chemin-theorie-des-graphes/algorithme-a-etoile/

https://stackoverflow.com/questions/735523/pathfinding-2d-java-game

Lecteur de musique : https://www.geeksforgeeks.org/play-audio-file-using-java/

Sauvegarde à l'aide d'un fichier csv : https://openclassrooms.com/fr/courses/4975451-demarrez-votre-projet-avec-java/5005441-travaillez-avec-un-fichier-csv