Projet POO : L'aventure dont vous êtes le héros.

Ce projet se base que le TP bataille : Un ou plusieurs héros se battent contre une horde d'ennemis. L'utilisateur joue un héros (ou plusieurs : cf propositions). Il a pour but de faire les bons choix afin de réussir sa quête.

Univers du jeu

Vous avez le choix de l'univers du jeu. Vous pouvez reprendre le domaine de l'héroïque-fantastique comme vu en TP. Vous pouvez également créer une histoire se passant dans l'espace, dans la vie de tous les jours, créer un jeu dans la veine de Pokémon, ou simuler un match de foot. Néanmoins, l'univers n'est qu'un support. Ne choisissez pas un domaine qui sera trop complexe à implémenter.

Contraintes

- Le travail devra être réalisé par groupe de DEUX uniquement.
- Vous proposez à l'utilisateur l'action à faire. A lui de choisir. Vous pouvez par exemple lui proposer d'attaquer ou de fuir et d'en subir les conséquences.
- Le projet sera en mode console. Vous pouvez faire autrement si vous le souhaiter, cependant cela ne sera que peu intégrer dans la notation.
- Vous n'utiliserez que de la Programmation Orienté Objet. L'objet principal sera un objet « Partie », qui contiendra la méthode « commencer() ». C'est la seule fonction qui sera appeler dans votre programme principal.
- Vous livrerez en même temps que votre code un rapport de projet listant les personnes du groupes, expliquant le contexte/l'histoire, le but que le joueur doit atteindre et les fonctionnalités que vous avez implémentées.
- L'utilisation de module supplémentaires est interdite.
- Aucun retard ne sera toléré.

Quêtes possibles et fonctionnalités minimums

Voici deux exemples de quêtes avec les fonctionnalités minimums que vous devrez proposer à l'utilisateur. Il vous est possible d'en créer d'autre à condition que l'utilisateur ait au moins deux choix d'action possibles. Dans tous les cas, la partie est perdu s'il n'a plus de point de vie.

Bataille

Votre héro doit se battre contre un ou plusieurs ennemis. S'il arrive à les vaincre il a gagné ou passe à un autre ennemi. Il peut soit faire une attaque simple, soit … autre chose comme faire une attaque avancée, ou appeler des renforts ….

Le voyage

Votre héro doit parcourir aller d'un point A à un point B (il doit parcourir un certain nombre de « case »). Par défaut, il avance. A chaque fois qu'il avance, il peut tomber sur un ennemi (ou plusieurs) qu'il doit affronter. Auquel cas, soit il attaque, soit il essaye de fuir. Au cas où il réussit à fuir, un malus devra être appliqué. Le but est de ne pas pouvoir réussir à finir la quête seulement en fuyant.

Proposition d'amélioration

Voici une liste non exhaustive d'amélioration possibles par rapport au projet de base. Ce ne sont que des propositions, et surtout pas une liste des fonctionnalités à implémenter. Vous être libre d'ajouter des points qui ne sont pas listés ci-dessous, qu'ils soient simples ou non à implémenter.

- Gestion d'équipement : Votre héro peut s'équipé avec une arme, un bouclier, une armure qu'il trouve durant sa quête
- Gestion de l'expérience : Votre héro gagne de l'expérience qui lui permet de lui gagner des niveaux et d'être plus fort. (Attention, c'est une amélioration complexe à balancer correctement, soit le jeu devient trop dur ou trop facile)

- Gestion des objets : Votre héro peut utiliser des objets lors de ces combats (ou autre) qu'il amassera au cours de sa quête.
- Gestion de l'argent et d'une boutique : Votre héro pourra tomber sur une boutique lui permettant d'acheter des objets ou de l'équipement avec de l'argent qu'il possède dès le départ ou qu'il gagne au cours de sa quête.
- Gestion d'une auberge/d'un lieu de soin : Votre héro pourra tomber sur une auberge lui permettant de se reposer et de regagner la totalité de ses points de vie.
- Gestion de Check-Point : Si la partie est perdu, il est possible de recommencer à une étape précédente du jeu
- Gestion de l'aléatoire : L'issue des combats est aléatoire. Dans le cas contraire, vous pouvez vous baser sur les valeurs prédéfinis, des listes de monstres fixes …
- Gestion d'une compagnie : Votre héro peut faire partie d'une équipe. Dans ce cas, soit le comportement des autres personnages et laisser à la charge de l'ordinateur, soit l'utilisateur joue l'ensemble des personnages, soit l'utilisateur peux choisir qui va agir.
- Gestion des rencontres : Le héros pourra rencontrer les personnages qui pourront se joindre à lui, l'attaquer, ou même le trahir.
- Gestion des capacités et de mana : Le héros peut utiliser des capacités particulières, lui permettant de faire des actions spéciales contre des points de mana (ou autre) comme se soigner ou utiliser un sort.
- Gestion du temps/du nombre de tour : le personnage qui agira est défini en fonction du temps qu'il a attendu. On peut liés ce paramètre à un paramètre « vitesse ». Plus ses points de vitesses sont élevés, plus il agit vite/souvent.
- Gestion des classes : Le héros peut appartenir à une classe qui définis ces actions possibles.
- Gestion de « easter eggs » et cheat codes : Vous pouvez rajouter les easter eggs à votre code, par exemple un choix/menu caché, une action particulière d'un personnage face à un objet particulier…. Ou encore permettre à l'utilisateur d'utiliser des codes pour gagner des potions, forcer un combat …. Cependant, afin que votre travail ne reste pas dans l'ombre, une liste devra être présente dans votre rapport.

Il est possible de proposer des améliorations sur l'interface :

- Mise en forme de l'affiche : utilisez des couleurs et autre
- Historique des actions : enregistrer l'historique des actions pour pouvoir rejouer la partie.
- Parcours narratif : L'histoire s'écrit au fur et à mesure. Elle peut également être enregistrée dans un fichier texte.
- Des dessins en ascii : Agrémenté l'histoire avec des dessins en ascii dans la console.

Conseils

- Ne voyez pas trop grand au risque de vous perdre dans votre code. Le but n'est pas d'implémenter l'ensemble des fonctionnalités précédentes
- Commentez correctement et au fur et à mesure votre code.
- Prenez le temps de soigner l'affichage de l'histoire et des menus de choix.
- Séparez votre code en plusieurs fichiers par exemple un fichier par classe.
- Enregistrez régulièrement votre travail une fois que vous avez une fonctionnalité terminée.
- L'implémentation de cheat code ne sert pas seulement à avoir de l'argent à l'infini. A la base c'est une fonctionnalité qui permettait aux développeurs de tester leur jeu, sans pour autant devoir faire toute l'histoire. Gardez cette possibilité en tête.
- Utilisez des solutions tel que git pour ce qui connaissent en commentant correctement vos « commits ». Vous pourrez laisser les répertoires git avec le projet.
- Ne vous y prenez pas au dernier moment.

Critères d'évaluations

Le projet sera noté sur les points suivants :

- Uniquement de la programmation orienté objet : Vous ne ferez que de la programmation orientée objet.

- Commentaires des différentes fonctions : Votre code devra être commenté, ainsi que les méthodes comme vu dans les TP précédents.
- Absence de bug : Votre code devra être fonctionnel et prévoir un maximum de cas de figure. Par exemple, si l'utilisateur essaye de faire une action qu'il n'a pas le droit de faire, vous devrez agir en conséquence.
- Fonctionnalité implémentée (nombre et difficulté) : un projet minimaliste sera moins valorisé qu'un projet implémentant beaucoup d'amélioration. Cependant, si les fonctionnalités en question sur sources d'erreur, cela n'ira pas en votre faveur.
- Détail apporté au projet : Chaque détail à son importance. Faites en sorte que votre projet soit agréable à utiliser.
- Qualité du rapport : votre rapport devra être soigné, tant sur la mise en page que sur le contenu, notamment les fautes d'orthographes.