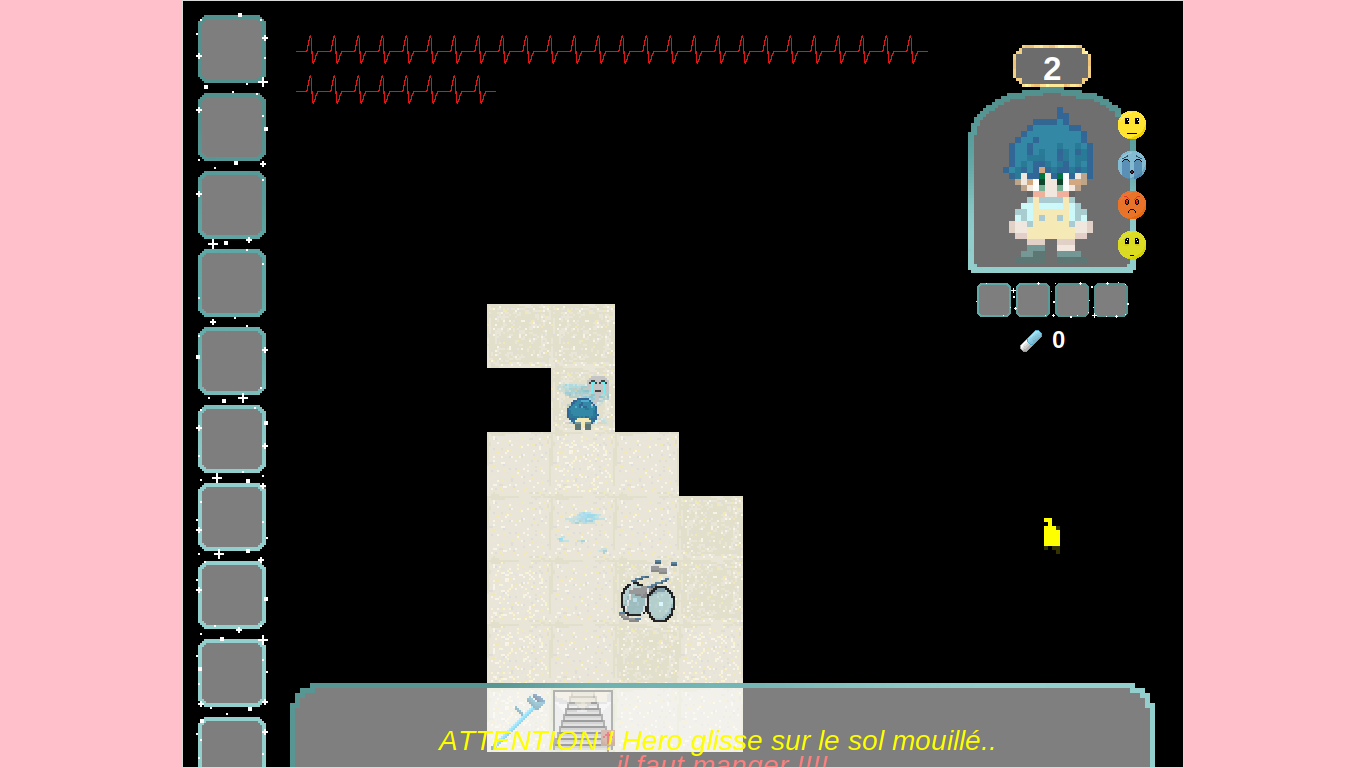
**Mental Maze**

Projet de Programmation Orientée Objet



*Interface graphique (avec le nuage de visibilité, la mini map en bas à droite, et la description du personnage)*

*Kylian, Vaiki, Nino, Hugo, Ilane, Arwen ont décidé de se mettre ensemble dans l’optique de réaliser un RogueLike.*

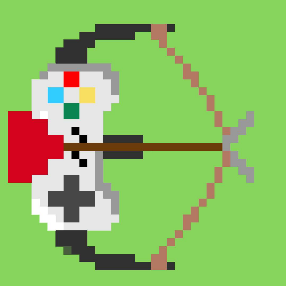
**Synopsis**

Un jeune garçon a perdu le sourire. Déprimé et coincé dans le plus haut étage de cet hôpital, il ne souhaite qu’une chose, descendre et sortir de ce lieu angoissant et qui lui fait percevoir des monstres s’en prenant à lui. Son humeur et certains lieux influencent ses actions et ses capacités. Guide-le et aide-le à survivre !

Gameplay:

**(4) Une interface graphique :**

Nous avons décidé de représenter notre jeu avec une interface graphique Tkinter. Les images ont été dessinées par les artistes de notre équipe de programmation. Chaque potion, chaque monstre, chaque arme est faite de notre imagination.



Les fonctions utilisées pour l’interface sont Game.initgraph(), game.begingame() et Game.updategraph() qui se chargent de l’affichage de la map.

1. **(1) Étages:**

Lors du commencement, on se trouve à l’étage 10 de l’hôpital. Plusieurs types de sols ont été créés grâce à des attributs qu’on a ajoutés dans la classe Map (.`,´) . Afin de sortir de l'hôpital, on a mis en place des étages. Pour cela, on a créé une classe Stairs qui a hérité de Special\_ground. Cette dernière, ainsi que la méthode meet permet au héro de changer d’étages.

1. **(1) Point d’expérience (XP)**:

Nous avons créé les fonctions Hero.level up() et Hero.Gainxp() pour gérer cette fonctionnalité. L’expérience est représentée par la hauteur des paillettes autour du héros et le level du héros s’affiche dans le cadre des émotions. Lorsque la barre des paillettes atteint son maximum, on gagne en niveau.

1. **(1) Inventaire limité**:

L’inventaire est affiché à gauche de l’écran sur l’interface. C’est un attribut de Créature sous forme d’une liste à laquelle on ajoute des éléments avec la fonction take. Ce dernier est limité, nous pouvons jeter un objet avec la touche j grâce à la fonction throw. L’or en est séparé. De plus, nous ne le représentons pas par des pièces, mais par des médicaments. Nous avons donc créé une classe Pilules.

1. **(3) Déplacements intelligents:**

La fonction MoveAllMonsters a été modifiée pour permettre aux monstres d’éviter les obstacles avec un déplacement intelligent de type A\*.

1. **(3) Nuage de visibilité et**  **(1) Nuage de visibilité+:**

Ils ont été réalisés avec la fonction Game.seeMap(). Elle permet de regarder dans toutes les directions avec un rayon variant de 0 à 6 cases de distance. Le champ de vision s’arrête au premier obstacle non transparent rencontré, comme les escaliers par exemple.

Ainsi, on peut voir sur la minimap le nuage de visibilité+. La mini-map affiche toutes les salles traversées par le personnage et colorie en jaune la partie visible à l’écran.

1. **(1) Diagonales**:

Le déplacement du héros et des monstres en diagonale ont été implémentés avec a,e,w,c grâce au dictionnaire des actions.

Actions

1. **(2) Jet:**

On a créé la fonction Creature.throw() qui est utilisée pour jeter les équipements en surplus de l’inventaire. De plus, certains objets comme le chewing-gum disposent de certaines fonctionnalités. En effet, cela permet d’immobiliser l’ennemi pendant 8 tours ou bien vient se coller au sol.

1. **(1) Repos**:

On peut dormir en se plaçant à proximité du lit. Nous avons fait le choix de ne pas limiter le repos pour faciliter le jeu, mais les hps gagnés en dormant ne peuvent pas dépasser les hpmax.

1. **(2) Magie:** Les mp n’ont pas été programmés.
   1. Téléportation : En appuyant sur o, on peut se téléporter, ça coûte 1 mp et cela coûte pas un certains niveau .
   2. Soin : De même, les potions de base permettent de se soigner.
   3. Invisibilité : En appuyant sur la touche l, le héro devient invisible pendant 6 tours, ou jusqu’à ce que le héros frappe un monstre.
2. **(1-3) Magie+:** D’autres sorts, par exemple : boule de feu (avec attackfire dans Map), vision (avec nuage de visibilité).

Objets

1. **(1) Nourriture**:

On utilise la fonction Hero.food() et eat() dans le usage() de certains objets pour gérer la nourriture et le niveau de satiété.

1. **(1) Armes:**

Nous avons créé une Class Weapon. Étant donné que le héros se trouve dans un hôpital, on a choisi comme arme: une béquille. Cette dernière lui permet d’attaquer les monstres avec plus de puissance.

1. **(1) Armes de jet:**

On a créé la fonction Tir. L’équipement est représenté par le lance pierre sur l’interface du jeu.

Le fonctionnement est similaire au chewing-gum, il évalue la première case et regarde s’il y a une créature. Sinon, il continue jusqu’à 6 cases de distance ou jusqu’à l’encontre d’un mur, et fait une rencontre (grâce à meet() ) à distance.

1. **(1) Armures:**

On a créé la classe Armure. Son fonctionnement est identique à la classe Arme

1. **(1) Amulettes:**

On a créé la classe Amulette qui fonctionne comme la classe Arme, excepté le fait qu’elle puisse seulement être équipée au Hero.

1. **(1) Solidité:**

C’est réalisé avec la méthode breakable() appelée dans meet.

Salles

1. **(1) Gestion des salles:**

Les salles spéciales sont triangulaires (créées grâce à un patron d’une matrice triangulaire). Elles sont créées avec la classe SpeRoom. Nous avons implémentés toutes les fonctions nécessaires pour effectuer les tests comme pour la classe Room, sauf qu’elle est triangulaire.

1. **(1) Pièges:**

Avec la classe Special\_Ground, nous avons créé un piège: une flaque d’eau. Ce dernier, nous fait perdre des pv une fois sur 5 en passant dessus.

1. **(2) Boutique:**

On a réalisé une classe Seller qui permet de créer un marchand. Ce marchand apparaît dans une salle et si le héros se dirige sur sa case, elle nous propose des objets qu’elle intègre directement à son inventaire.

1. **(2) Trésor:**

Le coffre est réalisé par la classe Coffre: il est apparaît tous les deux étages en commençant du deuxième. Si on a la clé, le coffre délivre, en disparaissant, des éléments sur les cases de la salle dans un rayon de 3 au alentours du coffre.

Monstres

1. **(1) Poison.**

Il a été réalisé grâce à la class Status. Le monstre araignée engendre des dégâts aux héros pendant 5 tours

**(1) Rapides.**

Pour cela, on a modifié l’attribut vitesse du monstre Creature("Angry emotivo",6,"O",strength=2). L’attribut vitesse a un impact dans Moveallmonster(), il frappe et se déplace, le nombre de fois que le permet son attribut

1. **(2) Archers.**

Nous avons crée une classe Archers. ses instances tire sur le héro à distance.

1. **(1) Invisibles**.

Nous avons fait des monstres invisibles avec la fonction Creature.unhide() en l’assignant à l’attribut spécial.

Suggestions supplémentaires

1. **Emotion**

Tous les 7 tours, on exécute la fonction emotioneffect(). Les émotions ont été mises en place grâce à la méthode emotion\_effect dans la class Hero. Les émotions entraînent des améliorations pour le héros mais également des dégâts. En effet, la joie permet de gagner en pv tandis que la tristesse diminue le champ de vision et baisse les pv si celle-ci est trop basse. Quant à la peur, elle a pour conséquence de bloquer le héros de jouer pendant un tour, il est paralysé. Si le héros est colère, il tape plus fort et se blesse donc il perd en pv.

1. **Décoration**

Nous avons implémenté dans les salles des objets décoratifs tels qu’ un arbuste et un fauteuil. De plus, on a ajouté à notre interface, un lit qui permet au héros de se reposer et ainsi de gagner des points de vie. Cependant, lorsque le héros se repose il ne peut pas jouer pendant 10 tours. Les décorations ont été ajouté grâce à la class Decoration qui hérite de Element.