PROJET JAVA

ALPHONSE LE CHEVALIER AVEUGLE

SOMMAIRE

1 - OBJECTIFS

CHAQUE PHASE AJOUTE UN DEGRÉ DE DIFFICULTÉ

PHASE 1 : ATTEINDRE LA SORTIE (AUTOMATIQUE), DÉFINIR UN ALGO

PHASE 2 : COUCHE GRAPHIQUE & INPUT CLAVIER (SEMI AUTO)

PHASE 3 : CONTRÔLE DU HÉRO & MONSTRE « INTELLIGENT »

2 - BRAINSTORMING & ALGORITHME

CHOIX PORTÉ SUR « MAIN DROITE »

-COUVRE UNE MAJORITÉ DE CAS (MAIS PAS 100%)

-SIMPLE À COMPRENDRE

CHAQUE TYPE DE CASE A SES PROPRES RÈGLES

COMMENT GÉRER L'AJOUT D'UN MOB ?

RÉPARTITION DES TÂCHES

BERTRAND: ALGORITHME, GESTION MOB

KARINE: ALGORITHME, DÉPLACEMENT HÉRO,

MATTHIEU : ALGORITHME, IMPORTER CARTE

LUCIEN : ALGORITHME, TESTING

ROADMAP & ÉTUDE FONCTIONNELLE

ÉTABLISSEMENT DU MODÈLE

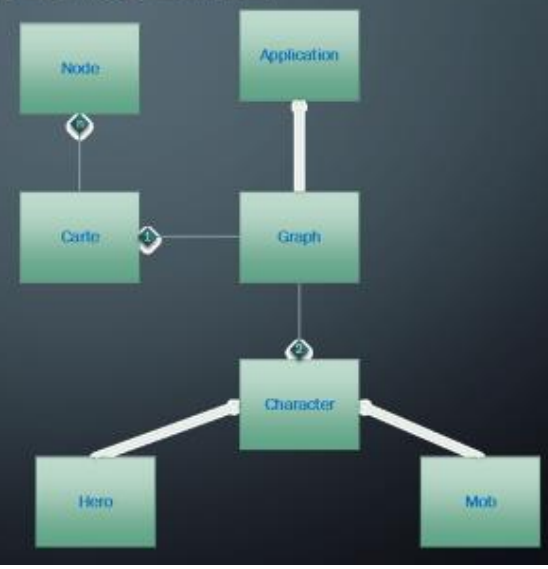
CONFORMITÉ A L'ALGORITHME DE DÉPART

TEST UNITAIRE A CHAQUE ÉTAPE

IMPLÉMENTER LES CLASSES

LE MODÈLE

SCHÉMA :



LES CLASSES

UNE CLASSE APPLICATION QUI LANCE LE PROGRAMME

UNE CLASSE GRAPH POUR LE CANVAS

DEUX CLASSES HERO ET MOB QUI HÉRITENT DE CHARACTER

UNE CLASSE NODE POUR LE PATHFINDING

TEST UNITAIRE

VÉRIFICATION DE TOUS LES CAS POSSIBLES

REMONTER LES CAS PARTICULIERS

CONFORME AUX ATTENTES

4 - CONCLUSION

PROGRAMME EN PLUSIEURS ÉTAPES

PLUSIEURS POSSIBILITÉS POUR CHAQUE PROBLÈME

PRODUIT INDÉFINIMENT AMÉLIORABLE

Algorithme du projet d'Alphonse

Declaration

Char case_courante → conserve la lettre de la case courante où se trouve Alphonse

Int x -> 1 // position horizontale d'Alphonse

Int y -> 1 // position verticale d'Alphonse

Char [][] Donjon → conserve la disposition des cases afin de le passer en paramètre d'entrée

Nous avons débuter en utilisant une fonction spécifique pour la case de départ à la position (1,1).

Au cours de l'implémentation, nous avons remarquer que la case de départ était une case comme les autres et qu'il n'y avait pas de raisons d'avoir une fonction spécifique.

Début DeterminerDeplacementDepart(Char [][] Donjon)

String deplacement=0

Si(case_courante = "F")

deplacement → "A"

direction = « Droite »

x ++

Sinon

deplacement → "DA"

direction = « Bas »

y ++

Fin

ModifierCaseCourante(x,y)

Fin

Annexe → l'algorithme des 4 fonctions de déplacements :

- void ToutDroit(Char [][] Donjon)
- void DemiTour(Char [][] Donjon)
- void Angle90(Char [][] Donjon)
- void Carrefour(Char [][] Donjon)

Affichage graphique en Java FX sous Eclipse

La class Graph nous a été transmise, on a créé un projet AlphonseFX.

On a commencé par importer les images pour les labyrinthes et reconfigurer les liens pour le projet.

On a remarqué que le système de notation des cases du labyrinthe était différent.

Tableau de correspondance des cases

-	L	f	F
t	T	e	D
l	M	p	U
m	H	v	R
+	arrivée		

Pour les cases restantes, nous avons dû réutiliser les images déjà existantes et les retourner avec setRotate(angle)

Annexe 1 :

Algorithme de la méthode ToutDroit

```
Début
      Cas RecupererCaseCourante() Parmi
       cas 'L':
           Si (direction="Droite")
                  Alors
                        x++;
                        ChangerDirection("Droite");
           Sinon //direction gauche
                  Alors
                        x--;
                        ChangerDirection("Gauche");
       case 'M':
           Si (direction="Haut")
                  Alors
                        y--;
                        ChangerDirection("Haut");
           Sinon //direction bas
                  Alors
                        y++;
                        setDirection("Bas");
      Ecrire("A");
      ModifierCaseCourante(donjon[x][y]);
Fin
```

Algorithme de la méthode DemiTour

```
Début
```

```
        Cas RecupererCaseCourante() Parmi
                cas 'C':
                        x--
                        ChangerDirection("Gauche");
                cas 'D':
                        y--
                        ChangerDirection("Haut");
                cas 'E':
                        y++
                        ChangerDirection("Bas");
                cas 'F':
                        x++
                        ChangerDirection("Droite");

        Ecrire("GGA");
        ModifierCaseCourante(dongeon[x][y]);
Fin
```

Algorithme de la méthode Angle90

```
Début
    Cas RecupererCaseCourante() Parmi
        cas 'R':
            Si (direction="Droite")
                Alors
                y++
                ChangerDirection("Bas")
                Ecrire(“DA“)
            Sinon
                Alors
                x--
                ChangerDirection("Gauche")
                Ecrire(“GA“)
        cas 'S':
            Si (direction="Bas")
                Alors
                x--
                ChangerDirection("Gauche")
                Ecrire(“DA“)
            Sinon
                Alors
                y--
                setDirection("Haut")
                Ecrire(“GA“)
        cas 'T':
            Si (direction="Haut")
                Alors
                x++
                ChangerDirection("Droite")
                Ecrire(“DA“)
            Sinon
                Alors
```

```
                        y++;
                        setDirection("Bas")
                        Ecrire(“GA“)
            cas 'U':
                  Si (direction="Bas")
                        Alors
                        x++
                        ChangerDirection("Droite")
                        Ecrire(“GA“)
                  Sinon
                        Alors
                        y--
                        setDirection("Haut")
                        Ecrire(“DA“)

      ModifierCaseCourante(dongeon[x][y])
Fin
```

Algorithme de la méthode Carrefour

```
Début
    Cas RecupererCaseCourante() Parmi
        cas 'G':
            Si (direction="Droite")
                y++
                ChangerDirection("Bas")
                Ecrire(“DA“)
            Sinon Si (direction="Haut")
                x++
                ChangerDirection("Droite")
                Ecrire(“GA“)
            Sinon
                x--
                ChangerDirection("Gauche")
                Ecrire(“A“)
                Sinon
        case 'H':
            Si (direction="Droite")
                y++
                ChangerDirection("Bas")
                Ecrire(“DA“)
            Sinon Si(direction = "Haut")
                y--
                ChangerDirection("Haut")
                Ecrire(“A“)
            Sinon
                x--
                ChangerDirection("Gauche")
                Ecrire(“DA“)
        case 'I':
            Si (direction="Gauche")
                y--
                ChangerDirection("Haut")
```

```
                    Ecrire(“DA“)
             Sinon Si(direction = "Haut")
                    x++
                    ChangerDirection("Droite")
                    Ecrire(“DA“)
             Sinon // direction Bas
                    y++
                    ChangerDirection("Gauche")
                    Ecrire(“A“)
      case 'J':
             Si (direction="Droite")
                    x++
                    ChangerDirection("Droite")
                    Ecrire(“A“)
             Sinon Si(direction = "Bas")
                    x--
                    ChangerDirection("Gauche")
                    Ecrire(“DA“)
             Sinon // direction haut
                    y--
                    ChangerDirection("Haut")
                    Ecrire(“DA“)


ModifierCaseCourante(dongeon[x][y])
```