Projet de POO

Gustav Hubert, Marwan Ait Addi,
Bernard Paul-Antoine

Plan de la soutenance

(selon l'énoncé à modifier) :

- Introduction
- Fonctionnalités implémentées
- Diagramme de classe, structure du code, techniques poo
- Organisation (temps, partage des tâches, groupe)
- Problèmes rencontrés, limitations
- Conclusion (bilan critique)

Fonctionnalités implémentées

Généralités

- Sauvegarde
- Différentes difficultés
- Champs de vision des soldats
- Chemin le plus court
- Les Éléments interagissant avec les points de vie du soldat
- Génération aléatoire de la Carte et placement aléatoire des soldats
- Gestion des actions des soldats
- Navigation et zoom sur le plateau de jeu
- Affichage du plateau avec profondeur
- Animation de dégâts
- Mode performance

Fonctionnalités implémentées

Calcul de la vision

```java
private void calcVisRec(int distance, Position pos, boolean cache){
    if(distance < 1 || !pos.estValide()) return;

    /*On rend tout d'abord visible la case en position pos*/
    if(cache){
        getCarte().getElement(pos).setCache();
    }else getCarte().getElement(pos).setVisible();

   /*On parcours ensuite toutes les positions voisine pour appeller recursivement la fonction*/
    Position test = new Position();
    for (int x = -1; x < 2; x++) {
        for (int y = -1; y < 2; y++) {
            test.set(pos.getX()+x, pos.getY()+y);
            if (pos.estVoisine(test))
                calcVisRec(distance-1, test, cache);
        }
    }
}
```

Fonction récursive de calcul de la vision (Heros.java)

```java
public void calculerVision(boolean cache){
    calcVisRec(this.getPortee(), this.getPos(), cache);
}
```

Fonction principale (Heros.java)

Fonctionnalités implémentées

Action des Héros

```java
public boolean actionHeros(Position pos, Position pos2){

    /*Vérifications préalables :*/
    /*Si pos2 n'est pas valide ou qu'il n'y a pas de soldat en pos ou que les cases ne sont pas adjacente*/
    if(getElement(pos).getSoldat() == null)
        return false;

    /*Si le soldat n'est pas un heros ou si un héros se trouve déjà en pos2*/
    if(!(getElement(pos).getSoldat() instanceof Heros) || getElement(pos2).getSoldat() instanceof Heros)
        return false;


    /*On va maintenant déterminer l'action a effectuer :*/
    /*Si la case n'est pas voisine on tente une attaque a distance */
    if(!pos.estVoisine(pos2)){
        /*Si le combat ne se fait pas on retourne false */
        if(!getElement(pos).getSoldat().joueTour()) return false;
        if(!getElement(pos).getSoldat().combat(getElement(pos2).getSoldat())){
            return false;
        }

    }

    /*On essaye de deplacer le soldat dans la case pos2, si on ne peut pas c'est qu'il y a un monstre*/
    if(!getElement(pos).getSoldat().getTour()) return false;
    if(!deplaceSoldat(pos2, getElement(pos).getSoldat())){
        if(getElement(pos2).getSoldat() != null){
            getElement(pos).getSoldat().combat(getElement(pos2).getSoldat());
            getElement(pos).getSoldat().joueTour();
        }else return false;
    }else getElement(pos2).getSoldat().joueTour();
    return true;
}
```

Récupération et traitement des données (Carte.java)

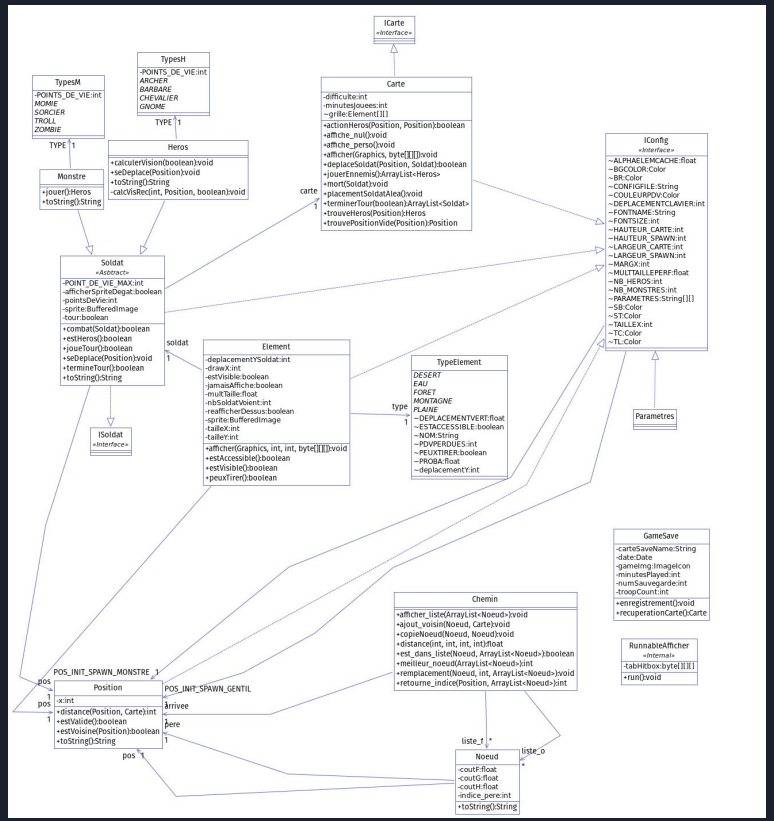
```java
public boolean combat(Soldat soldat){ /*On considere que l'instance pour laquelle est appelée c
    /*Premièrement le cas d'un combat au corps-a-corps (adjacent)*/
    if(getPos().estVoisine(soldat.getPos())){
        /*On fait un tirage entre 0 et la puissance du soldat*/
        soldat.pointsDeVie = soldat.pointsDeVie - (int)(Math.random() * this.PUISSANCE);
        if(soldat.pointsDeVie <= 0){ /*Si le soldat adverse est mort sur le coup*/
            carte.mort(soldat);
        }else{ /* Si il n'est pas mort il peut attaquer a son tour*/
            this.pointsDeVie = this.pointsDeVie - (int)(Math.random() * soldat.PUISSANCE);
            if(this.pointsDeVie <= 0) carte.mort(this);
        }
    /*Ensuite le cas d'un combat a distance (on utilise le tir et non la puissance)*/
    }else if(getPos().distance(soldat.getPos(), carte) <= this.getPortee()){ //Si le soldat adv
        /*On fait un tirage entre 0 et la puissance de tir du soldat*/
        soldat.pointsDeVie = soldat.pointsDeVie - (int)(Math.random() * TIR);
        if(soldat.pointsDeVie <= 0){ /*Si le soldat adverse est mort sur le coup*/
            carte.mort(soldat);
```

Gestion du combat entre deux soldats (Soldat.java)

```java
public boolean deplaceSoldat(Position pos, Soldat soldat){
    /*Si la position ou l'on veut se deplacer est vide et adjacente au soldat*/
    if(pos != null && getElement(pos) != null && getElement(pos).estAccessible() &&
        getElement(pos).getSoldat() == null && pos.estVoisine(soldat.getPos())){
        getElement(soldat.getPos()).setSoldat(null); /*On libère la case ou se trouvais le soldat*/
        soldat.seDeplace(pos);
        getElement(pos).setSoldat(soldat); /*On sauvegarde le soldat dans l'element de pos*/
        return true;
    }else return false;
}
```

Déplacement du soldat (Carte.java)

Diagramme de classe



Organisation du travail
répartition des tâches

Interfaces Wargame

IConfig
ICarte
ISoldat

Misc

Element : Gustav
GameSave : Paul-Antoine et Gustav
Position : Paul-Antoine et Marwan
Paramètres : Gustav

Unités

Soldat : Marwan
Héros : Marwan
Monstre : Marwan

Interface Utilisateur

Menus : Gustav
PanneauJeu: Gustav
Composants: Gustav
GenPolice: Gustav
TailleFenetre: Gustav

Terrains

Carte : tous
Chemin : Paul-Antoine
Noeuds : Paul-Antoine

Organisation du travail
temps

- Début tard
- Fixation de priorités
  - jeu fonctionnel avec le minimum nécessaire
    - Génération de la carte
    - Placement des soldats
    - Combat
    - Affichage
  - Améliorations progressives
    - Niveau de difficulté
    - Recherche de chemins
    - Navigation du plateau
    - Animation de dégâts
    - Champs de vision
    - ...

Problèmes rencontrés
(code : hexagone)

Screenshot d'une partie de la carte de jeu

- Adaptation en grille hexagonale
- Grille transformée à l'affichage
- 6 cases accessibles contre 8
- Numérotage des cases adjacentes inconsistent

Problèmes rencontrés
(technique : environnements utilisés)

- utilisation de VSCode pour l'écriture du code
  - module Java (sonarlint)
    - bugs
    - erreurs
    - compilation
    - extensif
- décision d'utiliser Eclipse pour la compilation
  - gestion correcte des erreurs
  - compilations fonctionnelle
- partage des fichiers : Github
  - gestion problématique des .class
  - modifiés à chaque utilisation d'un membre du groupe
  - nécessaires compilation

S

- Mauvaises p
  - Platea
  - Créati
  - Charge
  - Grand
- Meilleur per
  - Mode
  - Taille d
  - Rafraî

Conclusion

- Plusieurs domaines que l'on aurait aimé approfondir si le temps n'était pas limité :
  - Les I.A des monstres
    - implémenter des IA et plusieurs types
  - Des animations dans l'affichage
    - tir des soldats
    - déplacement
    - sélection visuelle
  - Une amélioration de la carte de jeu
    - construction de fortifications pour s'abriter
    - ajout de régions différentes
- Beaucoup de compétences acquises
  - Travail de groupe
  - Séparation des tâches
  - Utilisation de java en situation "réelle"
  - Aisance avec les IDE