TP8 - Classes

CAU Simon - COUSSEAU Yanis

Déviations de la conception

Types d'attaque

Nous avons suivi la conception indiquée sur le sujet de TP à un détail près: nous utilisons une énumératon à la place d'un entier pour indiquer le type d'attaque.

```
enum class TypeAttaque
{
    PIERRE, FEUILLE, CISEAUX, MAX_VAL
};
```

MAX_VAL est une valeur sentinelle, elle représente la valeur max (en forme numérique) que peut prendre l'énumération. Cela nous sert à pouvoir rajouter des attaques sans modifier le code génération.

```
//Generate random attaque type
type = (TypeAttaque) (rand() % (int)TypeAttaque::MAX_VAL);
```

getLettre

Pour faciliter l'affichage du plateau, nous avons ajouté une méthode virtuelle pure :

```
virtual char getLetter() const = 0;
```

Chaque Animal doit donc implémenter cette méthode qui renvoie le char représentant l'Animal

```
//Exemple d'implémentation
char Lion::getLetter() const
{
    return 'L';
}
```

Plateau

Une fois toutes les classes définies et implémentées, nous avons créé une classe `Plateau`.

Cette classe est responsable du jeu (affichage et exécution des tours).

```
class Plateau
{
private:
    Animal* plateau[MAXX][MAXY];
public:
    Plateau(/* args */);
    void affichage() const;
    void simulerTour();
};
```

Le plateau est représenté par un tableau de pointeurs à 2 dimensions.

Affichage

L'affichage parcours simplement le tableau `plateau`. Pour chaque case, s'il y a un animal on affiche sa lettre.

Pour déterminer si un animal est dans une case, on vérifie simplement que le pointeur contenu dans le tableau n'est pas nul.

```
Animal* cell = plateau[i][j];
if(cell == nullptr) {
    std::cout << "-";
} else {
    std::cout << cell->getLetter();
}
```

Exécution d'un tour

Un tour se déroule en 2 phases:

1. Déplacement de tous les animaux
2. Résolution des conflits

Pour déplacer tous les animaux on parcours toutes les cases du tableau (ce qui est relativement lent). Afin d'optimiser la deuxième phase, on profite de l'itteration pour remplir une liste de tous les animaux vivants sur le plateau.

La deuxième peut alors simplement parcourir la liste (25 itérations max vs 100 pour le plateau entier).

L'algorithme pour la partie résolution de conflit est très largement améliorable, mais reste très peu optimisé par manque de temps:

Pour chaque animal dans la liste, on vérifie qu'il n'y aie pas un autre animal de la liste qui possède les mêmes coordonnées.

Si l'animal actuel et celui que l'on est entrain de vérifier possèdent les mêmes coordonnées alors les deux se battent.