Projet : Spaceship breach

Équipe et Contributions :

Storck Jefferson : L'intégralité du projet

Introduction :

L'objectif du projet est d'implémenter un moteur de jeu en 2D pouvant être enrichi sans modifier le code.

Le type de jeu que le projet cherche à implémenter est un jeu de stratégie tactique en tour par tour.

Il devra être possible d'ajouter des cartes, combat (ennemies, joueurs, …) à partir de fichiers JSON et de cartes Tiled.

Présentation du projet :

Outils Utilisés : LibGDX, Tiled(Cartes), Eclipse(IDE) et Gimp(Sprites)

Fonctionnalité :

-Un system de tout par tour

-Un lecteur de fichier JSON(sous un format donné)

-Gestion des collisions avec Tiled

-Gestion de différents comportements pour les ennemis (classe Behaviour et ses sous-classes)

-Possibilité de jouer avec plusieurs personnages différents dans la même escouade

Projet : Spaceship breach

Ajouter une carte :

Pour qu'une carte Tiled soit éligible à être ajouter au jeu elle doit respecter les contraintes suivantes :

- La cartes doit posséder 10 tuiles de largeur et longueur chacun de 32 pixels

-Toutes les cartes doivent avoir une Object Layer Wall(Rectangle Objet pour mur) et une Object Layer Next(Rectangle Objet pour passage à la carte suivante)

-Toutes les cartes doivent être nommer spacex.tmx avec x un nombre quelconque

-Chaque Object de Wall et Next doivent avoir leur attribue x,y,width and height comme multiple de 32

Ajouter un combat :

Combat sont ajouter à l'aide de fichier JSON pour ajouter un combat le fichier doit être nommer cx.json avec x un quelconque non égale à 0, il doit aussi exister un fichier spacey.tmx correspondant à une carte avec x=y,

Ensuite le fichier doit suivre le format ci-dessous :

Avec la premiere partie du fichier représentant le type d'ennemi possible et la seconde une listes des ennemis présent sur la carte avec leur type et coordonnées

Un type d'enemie posséde au moins sept variable :

hp(point de vie),mp(mouvement),init(initiative),ap(portée),dam(dommage),

sprite(chemin vers le sprite de l'ennemi) et enfin type : le comportement adopté par l'enemie il en existe quatre implémentés de base :Melee, Ranged,Officer et Commander, d'avantage peuvent être implementé en créant une nouvelle classe héritant de Behaviour.

Projet : Spaceship breach


```
{
  "Mbot": {
        "hp": "6",
        "mp": "4",
        "init": "2",
        "ap": "1",
        "dam": "1",
        "type":"Melee",
        "sprite":"enemy/Mbot.png"
  },
  "Rbot": {
        "hp": "4",
        "mp": "2",
        "init": "3",
        "ap": "4",
        "dam": "1",
        "type":"Ranged",
        "sprite":"enemy/Rbot.png"
  },
  "Enemy": [
        {
                "type": "Mbot",
                "cox": "5",
                "coy": "4"
        },
        {
                "type": "Mbot",
                "cox": "5",
                "coy": "6"
        }
  ]
}
```


Modifier les joueurs :

Toutes les informations relatives au joueur son contenues dans le fichier c0.Json il suit le même format qu'un fichier de combat classique à l'exception que le tableau est nommer « Player » au lien « Enemy ».

Un type joueur doit aussi avoir un attribut « level » et ses types sont différent de ceux d'un ennemi, il en existe deux implémentés de base : Player et Support

D'avantage peuvent être implémentés en créant une classe héritant de Player

Projet : Spaceship breach

Compilation et exécution :

1.exporter le projet Gradle dans eclipse

2.modifier le build.gradle fichier

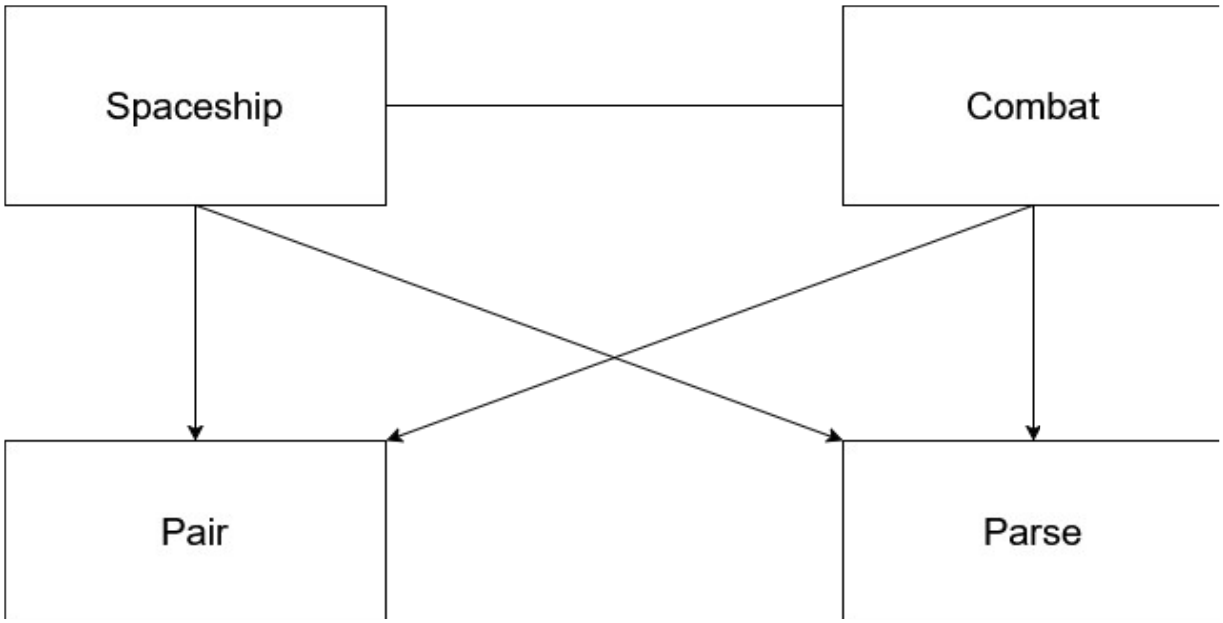
3.commencer la tache build du projet

GitHub : https://github.com/idees1/POOProjet

Présentation technique du projet :

La classe principale du projet est Spaceship, c'est elle qui affiche les sprites et les cartes, quand un combat commence elle créé une instance de la classe Combat qui est utilisé pour toutes les fonctions de combat de la génération des ennemis au calcule des mouvements/cibles possibles des personnages.

Ces deux classes utilisent la classe Parse pour lire les fichier JSON et la classe Pair pour représenter les coordonnées.



Projet : Spaceship breach

Toutes les classes de personnages (ennemi ou joueur) héritent de la classe abstraite Char, toutes les classe d'ennemi hérite ou son de la classe Enemy, pareil pour les classe joueur avec Player

Pour ajouter un type de jouer il suffit de créer une nouvelle classe qui hérite de Player et de la référencer dans c0.json.

Pour ajouter un type d'ennemis il faut créer un nouveau type d'ennemi il faut créer un nouveau comportement, pour se faire il faut créer une nouvelle classe qui hérite de Behaviour.

Toutes les classes qui définisse un comportement (Melee,Ranged, …) hérite de la classe abstraite Behaviour.

Conclusion et Perspectives :

En conclusion l'application atteint l'objectif donné d'implémenter un moteur de jeu en 2D pouvant être enrichi sans modifier le code, à l'aide de son lecteur de fichier JSON et de classe comme Char et Behaviour elle peut de nouvelle cartes, ennemis et joueurs peuvent être ajouter facilement.

On peut imaginer beaucoup de nouvelle fonctionnalité comme la capacité de voir les ennemis jouer durant leur tour ou l'ajout de capacités personnalisables à donner au personnage.