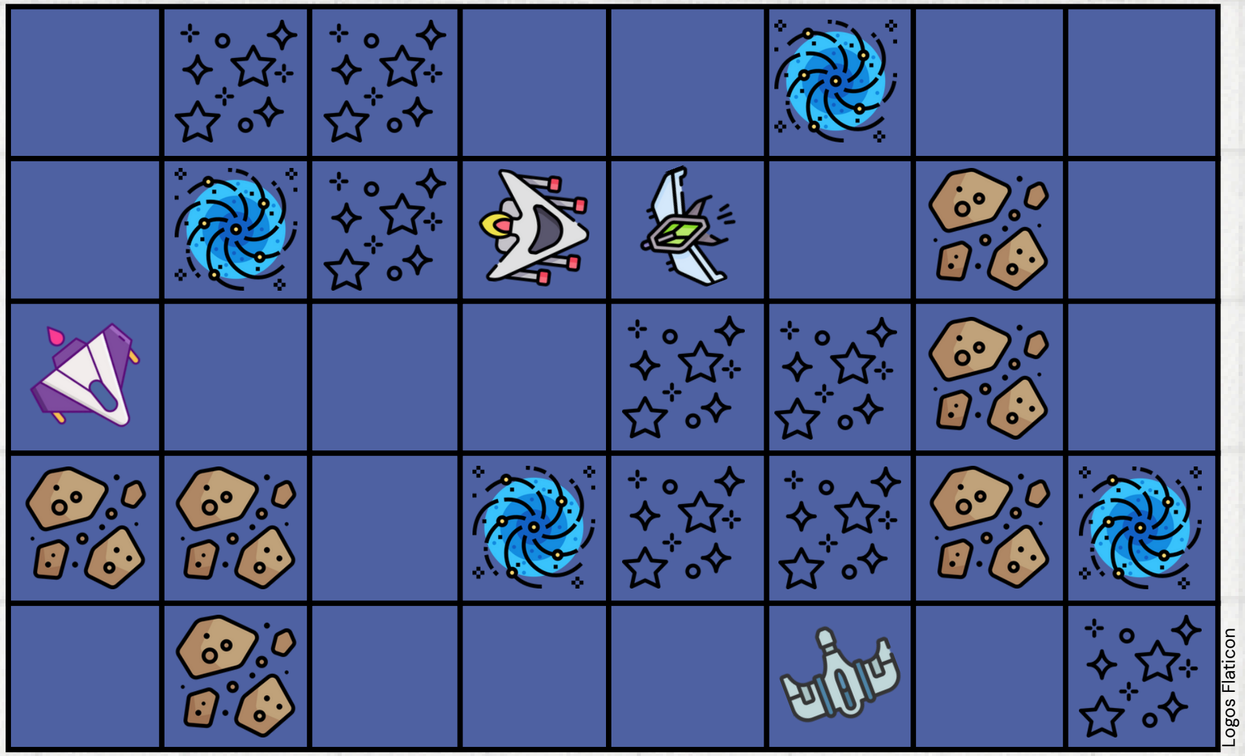
Space Game Project

Lemarié Bastien – Penarroya Solenn

Objectifs du projet



Logos Flaticon

Architecture

Organisation de notre binôme

- Coder ensemble
- Utilisation de GitHub
- Sessions de travail à distance

Processus de création

01
Diagramme de classes

02
Les vaisseaux et régions

03
Gestion des combats

04
Gestion des flottes

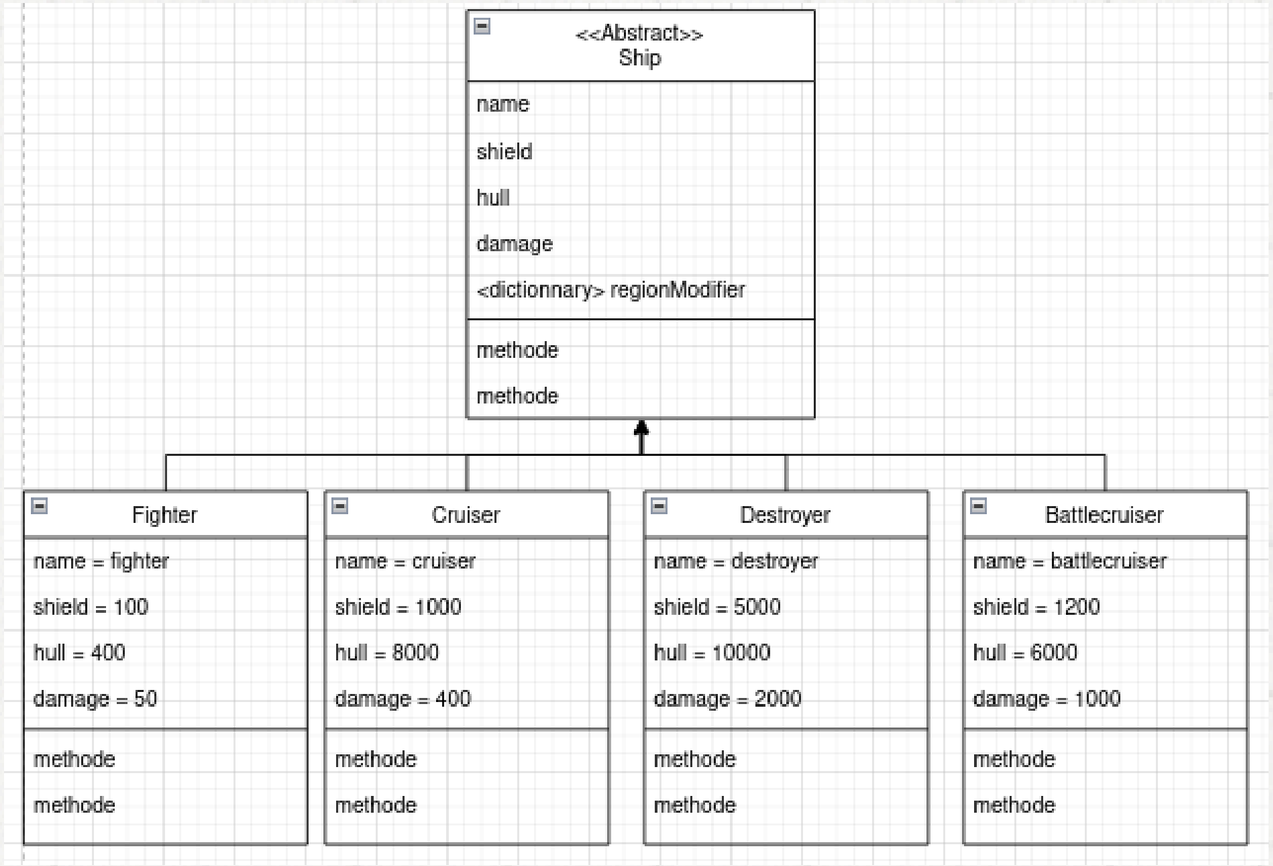
05
Gestion des logs

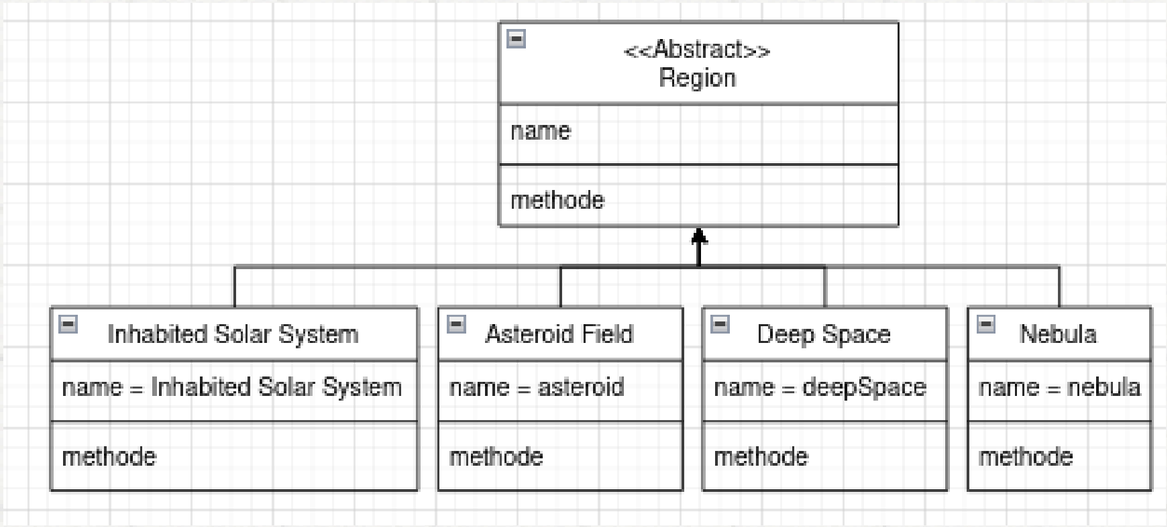
06
Le board et les déplacements

07
Le visuel

En parallèle : Rédaction de tests unitaires

02. Vaisseaux et Régions





02. Vaisseaux et Régions

Double dictionnaire

```
Object << #Ship
    slots: { #name . #parentFleet . #shield . #hull . #damage . #regionModifier };
    package: 'SpaceGamePackage'
```

Dans Ship

```
initializeRegionModifier

    regionModifier := Dictionary new.

    regionModifier at: 'shieldModifier' put: Dictionary new.
    regionModifier at: 'hullModifier' put: Dictionary new.
    regionModifier at: 'precisionModifier' put: Dictionary new.
    regionModifier at: 'damageModifier' put: Dictionary new.
    regionModifier at: 'defenseModifier' put: Dictionary new.

    ^ regionModifier
```

Dans BattleCruiser

```
initializeRegionModifier

    | shieldModifier hullModifier damageModifier precisionModifier defenseModifier |
    super initializeRegionModifier.
    shieldModifier := regionModifier at: 'shieldModifier'.
    shieldModifier at: 'DeepSpace' put: 1.
    shieldModifier at: 'Nebula' put: 1.
    shieldModifier at: 'AsteroidField' put: 1.
    shieldModifier at: 'InhabitedSolarSystem' put: 1.
    regionModifier at: 'shieldModifier' put: shieldModifier.

    hullModifier := regionModifier at: 'hullModifier'.
    hullModifier at: 'DeepSpace' put: 1.
    hullModifier at: 'Nebula' put: 1.
    hullModifier at: 'AsteroidField' put: 1.
    hullModifier at: 'InhabitedSolarSystem' put: 1.
    regionModifier at: 'hullModifier' put: hullModifier.
```

Pattern Strategy ?

03. Gestion des combats

Double Dispatch

Destroyer !

```
attackedBy: anAttackingFleet in: aRegion

    ^ anAttackingFleet attackedByDestroyer: self in: aRegion
```

Ship !

```
attackedByDestroyer: aDestroyer in: aRegion

    | dmg remaining |
    dmg := self damage
           * (self regionModifier at: 'damageModifier' at: aRegion id)
           *
           (self regionModifier at: 'defenseModifier' at: self region id).


    remaining := shield - (dmg * 2).
    remaining > 0
       ifTrue: [ shield := remaining ]
       ifFalse: [
          shield := 0.
          remaining := hull + remaining / 4.
          remaining > 0
             ifTrue: [ hull := remaining ]
             ifFalse: [
                hull := 0.
                parentFleet
                   otherShipTakingDamages: 0 - (remaining * 4)
                   ship: aDestroyer ] ]
```

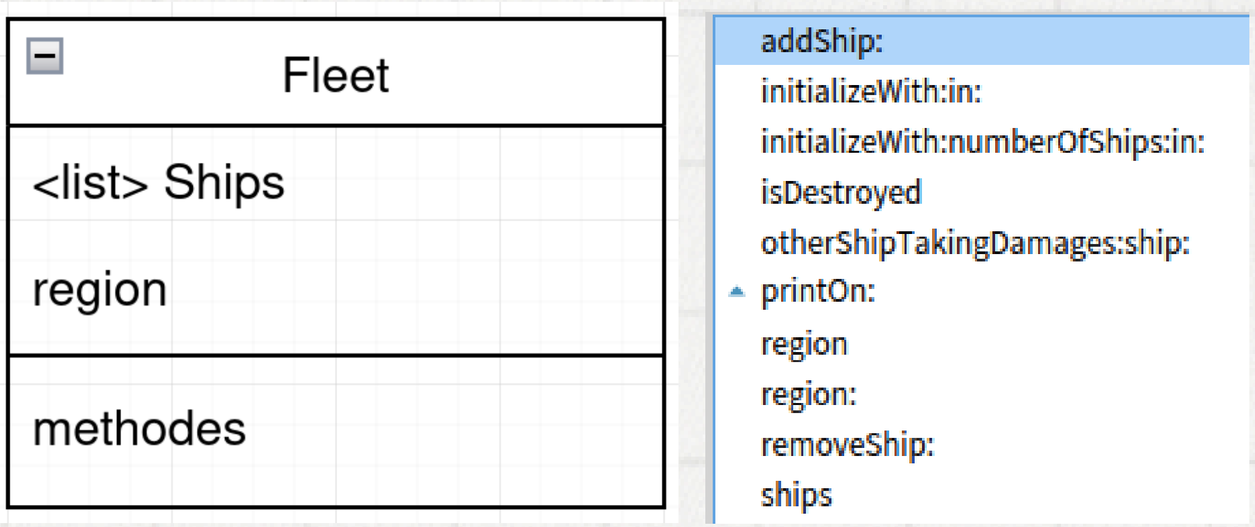
03. Gestion des combats

Classe

startBattle

```
| defender |
self addLogStart.
[ fleetA isDestroyed or: [ fleetB isDestroyed or: [ turn > 3 ] ] ]
   whileFalse: [
      fleetA ships do: [ :attacker |
         defender := fleetB ships first.
         defender attackedBy: attacker in: defender region ].
      fleetB ships do: [ :attacker |
         defender := fleetA ships first.
         defender attackedBy: attacker in: defender region ].
      self addLogTurn.
      turn := turn + 1 ].
self addLogResult
```

04. Gestion des flottes



addShip:
initializeWith:in:
initializeWith:numberOfShips:in:
isDestroyed
otherShipTakingDamages:ship:
printOn:
region
region:
removeShip:
ships

Impact : Modifications des combats

05. Gestion des logs

Il y a de l'envoi de logs dans la classe Battle et dans les méthodes de combat

Réflexion sur ce qu'on a implémenté :

- Voir pour utiliser la délégation en créant une classe "Logger"

06. Board et déplacements

- Grille
- Utilisation du NullObject

Réflexion sur ce qu'on a implémenté :

- Découpler le Board et les Fleet
- Découpler Tile et Board

-> Donner au board la responsabilité (logique) du déplacement des flottes

- Voir pour utiliser le pattern Composite

Démonstration : les tests

Conclusion