

Architecture du projet

2015-2016

|  |  |
| --- | --- |
|  |  |

Découpage des modules

Je vous propose de découper le projet en 5 modules :

## Gestionnaire central

Un module pour les gouverner tous

Ce module sert à faire le lien entre tous les modules, c’est le main de notre programme. Il contiendra le minimum de code possible.

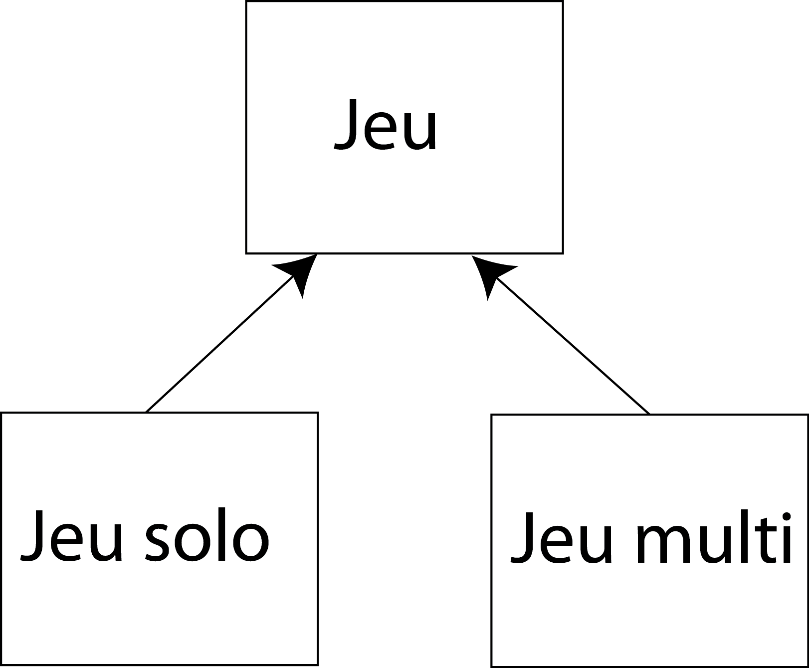
## Moteur de jeu

Ce module contiendra la partie logique du jeu de Tetris (pas de graphisme).

Il définira la représentation d’un tableau de jeu, des différentes pièces, du mouvement des pièces, calcul du score, etc…

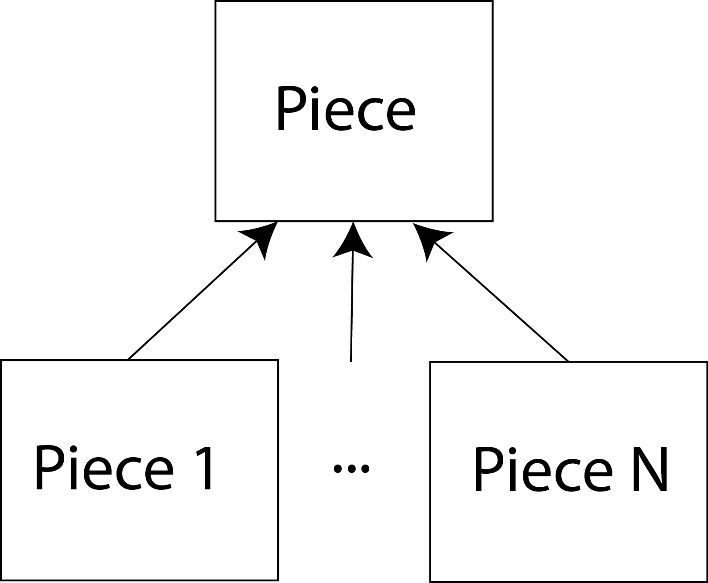
Je propose ce début d’architecture pour cette partie :

### Jeu (classe centrale)



La classe Jeu contient les informations relatives à un jeu dans sa généralité, les classes JeuSolo et JeuMulti héritent de Jeu et le spécialise. J’ai créé ces fichiers dans le package Moteur de jeu avec quelque attributs et méthodes à compléter.

### Pièces



Pour la représentation des pièces je pense qu’il serait judicieux de s’inspirer du design pattern de la Factory (<http://www.jmdoudoux.fr/java/dej/chap-design-patterns.htm>).

### Plateau de jeu

J’ai écrit une classe PlateauJeu à compléter. Un plateau de jeu est une représentation d’un jeu Tetris, c’est la structure de donnée de base. Je pense qu’il vaut mieux l’implémenter comme une simple matrice d’entiers.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | 0 | 2 | 2 | 0 | | 0 | 2 | 2 | 0 | | 0 | 0 | 0 | 0 | | 1 | 0 | 0 | 3 | | 1 | 1 | 0 | 3 | | C:\Users\Abi\Documents\GitHub\Tetris\doc\plateau de jeu.png |

Plateau de jeu contiendra une bonne partie de la logique du jeu Tetris.

J’ai défini les méthodes à compléter suivantes :

* ExploserLigne() : Fait exploser les lignes si possible
* Run() : Fait tomber les pièces
* AddPiece (pièce) : rajoute une pièce au jeu
* A compléter…

## Moteur graphique

Le moteur graphique aura pour rôle d’afficher le jeu solo ou multi, en mode console ou en mode fenêtré. J’ai écrit une classe simple qui contient uniquement les méthodes dont le Gestionnaire central aura besoin.

* Init\_graphique() : démarre l’interface graphique.
* Update\_graphique(Plateau de jeu, prochaine pièce, score1, score2, malus, …) : Met à jour l’affichage du jeu.

L’interface devra permettre d’afficher :

* Le plateau de jeu
* Le(s) Score(s)
* La prochaine pièce qui va tomber
* Le Malus actif (si nécessaire)

L’architecture interne reste à définir.

Görkem tu t’occupes de cette partie ?!

## Interfaces logiciel

Cette partie regroupe l’interface de lancement d’une partie et l’interface de fin de partie.

Mieux vaut faire le minimum ici :

### Ecran de démarrage

* Jeu solo => lancement de la partie
* Jeu Multi =>
  + En attente d’un autre joueur => lancement de la partie (si on reçoit une invitation)
  + Inviter un joueur =>
    - Entrez IP => lancement de la partie (si l’invitation est acceptée)
* Meilleurs scores

### Ecran de fin de partie

* Score final (partie solo)
* Perdu/Gagné (partie multi)
* Meilleurs scores

Je pense que celui qui s’occupe de la partie moteur graphique devrai aussi faire cette partie.

## Moteur Réseau

Cette partie regroupe toutes les interactions réseau.

Le moteur doit implémenter ces différentes fonctionnalités :

* Envoyer/Recevoir une invitation
* Envoyer/Recevoir Score
* Envoyer/Recevoir Malus
* Envoyer/Recevoir ordre de fin de partie

Le faire ainsi suppose un fonctionnement assez simple du jeu :

* Le joueur ne voit que le score de son adversaire.
* Lorsque l’on casse 10 lignes le jeu envoi un ordre de malus à l’autre joueur.
* Lorsque l’on perd le jeu envoi un ordre de fin de partie à l’autre joueur.
* Si on perd la connexion le jeu s’arrête.