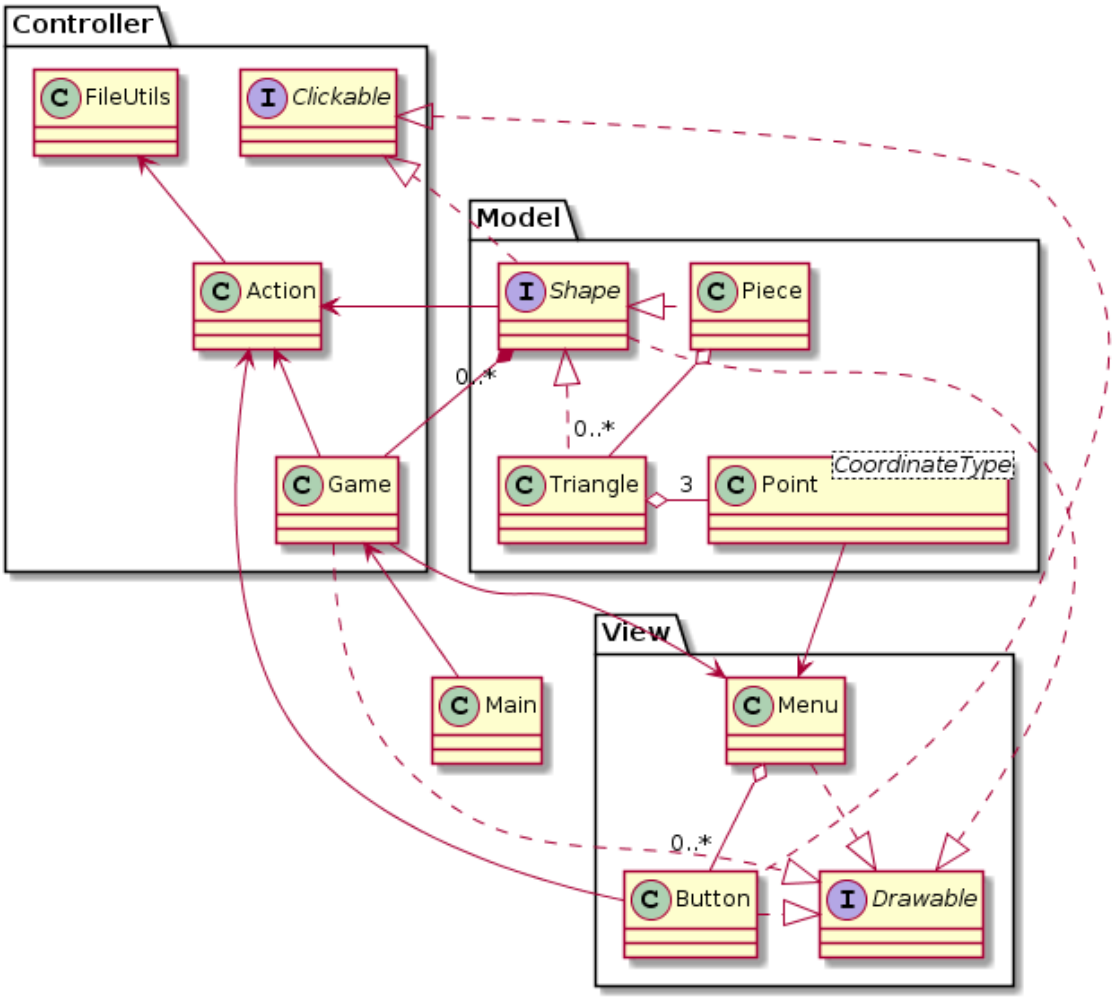
GOSSET Séverin
BIGUENET Denis

Rapport UML Tangram

Schéma UML


Controller
FileUtils
Clickable
Action
Game
Model
Shape
Piece
Triangle
Point
CoordinateType
0..*
0..*
3
View
Main
Menu
Button
Drawable
0..*

Rôle des classes

Point

La classe la plus basique du projet est la classe Point, qui représente un point à partir de deux coordonnées (x, y). Il peut être judicieux de faire de cette classe un template, cela pourrait avoir plusieurs avantages. Premièrement, les coordonnées x et y pourront être de n'importe quel type (int, double ou float) sans que l'on ait à surcharger les méthodes ou modifier leur prototype. Deuxièmement, si le code est récupéré par une autre personne pour le modifier, il n'aura pas à se sourcier du type des valeurs entrées lorsqu'il crée des points.

Triangle & Pièce

Un triangle est composé de trois points distincts (relation d'agrégation). Ces triangles servent à créer nos pièces. En effet, plutôt que de créer une classe par type de pièce (triangle, parallélogramme…) implémentant une interface, il peut être plus logique de composer nos pièces de triangles (relation d'agrégation). Chaque pièce est donc composée d'une lise de triangles, sur lesquels on peut effectuer des opérations.
Par exemple, on cherche à déterminer si un point est dans une forme ou non, dans le modèle «une classe par type de pièce », on doit implémenter cette méthode pour chacun de ces types. Dans ce nouveau modèle, il suffit d'implémenter la méthode pour le triangle et d'appeler celle-ci sur tous les triangles de la pièce.
Plusieurs avantages :

- Inutile de créer une classe par type de pièce.
- Une seule implémentation pour chaque méthode peu importe le nombre de forme possible, toute pièce est traitée comme un ensemble de triangles.

Shape

Shape permet de définir une interface commune entre les pièces et les triangles, car certaines pièces sont en fait composées d'un unique triangle. Cette interface implémente elle-même l'interface Clickable, car une pièce doit être sélectionnable, ainsi que l'interface Drawable, car on peut dessiner une pièce.

Button

Un bouton est un carré ou un rectangle définit par un point haut-gauche et un point bas-droit. Etant donné que l'on peut interagir avec un bouton, cette classe implémente donc l'interface Clickable.

Clickable

Interface commune pour tous les objets alors lesquels le joueur peut interagir, en l'occurrence, les pièces et les boutons.

Menu

La classe Menu est une agrégation de boutons accompagnés d'une interface, un menu peut être dessiné et implémente donc l'interface Drawable.

Drawable

L'interface Drawable permet de définir une interface commune pour tout ce qui peut être dessiné, notamment les pièces, les boutons, et le menu.

Action

Une Action représente quelque chose à faire sur un objet, à un ou plusieurs moments. Cette classe sera un Foncteur, et prendra en argument une fonction lors de sa création. Cette fonction sera appelée lorsque l'on fera appel au foncteur. Une action est une classe qui sera utilisée entre autres par les boutons, pour définir leur action (relation d'association).

Game

La classe Game sert à définir ce qu'est une partie, c'est-à-dire :

- Un menu (des boutons, une interface…)
- Une liste de pièces
- Un certains nombres d'interactions possibles suivant un contexte

Cette classe implémente l'interface Drawable car elle contient tout ce qui peut être et doit être dessiné à un moment donné de la partie.

Main

Cette classe crée une partie (Game) et n'appelle que des méthodes de celle-ci (condition de victoire, affichage…), afin de déléguer un maximum de tâches et d'écrire un minimum de code dans ce Main.

FileUtils

Cette classe statique sert à gérer les écritures et lectures de fichiers (charger une partie, sauvegarder une partie)

MVC

Ce projet suit une architecture MVC, on peut alors le diviser en trois packages Model View et Controller dans lesquels sont répartis nos classes :

- Model : Shape, Piece, Triangle, Point
- View : Drawable, Menu, Button
- Controller : Game, Action, FileUtils, Clickable