Projet DeViNT
IHM - MVC

Thomas Jalabert
Thibaut Gonnin

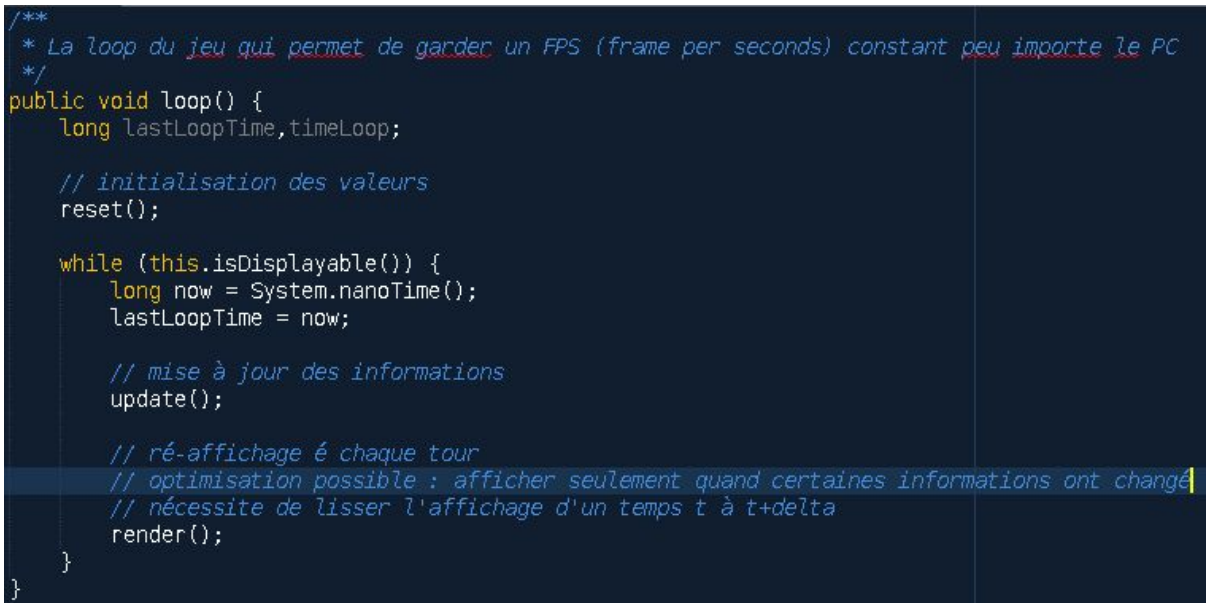
Groupe 1 - SI3
Polytech'Nice - Sophia

Sommaire :

1. Le code du modèle de jeu ne suit pas l'architecture MVC ? Donnez les parties du code qui ne respectent pas ce pattern. Argumentez. En quoi l'architecture mise en place limite l'évolution du code ?
    a. Le code du modèle de jeu ne suit pas l'architecture MVC ?

En effet, le modèle de jeu suis l'architecture GameLoop.

    b. Donnez les parties du code qui ne respectent pas ce pattern. Argumentez.



```java
/**
 * La loop du jeu qui permet de garder un FPS (frame per seconds) constant peu importe le PC
 */
public void loop() {
    long lastLoopTime,timeLoop;

    // initialisation des valeurs
    reset();

    while (this.isDisplayable()) {
        long now = System.nanoTime();
        lastLoopTime = now;

        // mise à jour des informations
        update();

        // ré-affichage é chaque tour
        // optimisation possible : afficher seulement quand certaines informations ont changé
        // nécessite de lisser l'affichage d'un temps t à t+delta
        render();
    }
}
```

Ceci représente l'architecture du jeu devint. On remarque bien le partern Game Loop ici. Avec la fonction reset() qui réinitialise le jeu. La boucle de jeu qui se déroule tant que le jeu continue, update() qui s'occupe respectivement de mettre à jour la structure de donnée en fonction des actions de l'utilisateur et render() qui s'adapte en modifiant l'interface graphique.

    c. En quoi l'architecture mise en place limite l'évolution du code ?

L'architecture mise en place ne respecte pas une structure objet, mais plutôt une structure procédurale. En effet, chaque étape du jeu est développé dans une fonction à l'intérieur de Jeu (reset, init, update et render).
Une architecture procédurale limite l'évolution du code car si l'on veux ajouter une autre fonctionnalité (un autre mode de jeu par exemple), la majorité des fonctions devrons être recodée. Il en résulte un travail de maintenance plus conséquent.
De plus, l'architecture game loop limite l'évolution du code. En effet, lorsque le jeu est terminé l'ajout de nouvelles fonctionnalitées est difficile. Il est bien plus simple en MVC d'ajouter une vue ou un nouveau controleur que dans le pattern game loop.

2. Comment pouvez-vous modifier le code du modèle de jeu pour qu'il respecte MVC ? Donnez les parties de code à modifier et leurs modifications. Justifiez.

a. Comment pouvez-vous modifier le code du modèle de jeu pour qu'il respecte MVC ?

Il faudrait tout d'abord créer les objets Modèle, Vue et Contrôleur qui vont interagir lors de l'exécution du jeu. Ensuite Il faut segmenter les fonctions existantes dans jeu (reset, init, update et render) en fonction de leur appartenance à la vue, au modèle ou au contrôleur. Par exemple, la fonction reset réinitialise la partie graphique de l'application mais aussi certaines variables du modèle pour recommencer le jeu.

b. Donnez les parties de code à modifier et leurs modifications. Justifiez.

Pour avoir une architecture MVC il faut diviser le code en plusieurs partie.

Tout d'abord nous avons le contrôleur qui permet de faire le lien entre la vue et le modèle. on voit que notre classe possède en attribut notre modèle et 2 fonctions permettant de mettre à jour la vue et de contrôler la vue pour vérifier les actions utilisateurs.

Ensuite nous avons la vue qui doit pouvoir recevoir des informations de la part du modèle et du contrôleur et d'envoyer des informations au contrôleur. C'est pour cette raison que le modèle et le contrôleur sont des attributs de la classe et que ces différentes actions sont représentées par les différentes méthode de la classe. la méthode display étant faite pour afficher l'interface à l'utilisateur.

Dans notre exemple la classe modèle est vide car elle dépend vraiment de chaque utilisation et donc de chaque programme.

Enfin, la classe Jeu permet de mettre en place toute l'architecture. On remarquera que cette classe peut être en lien avec plusieurs contrôleurs, vues et modèles à l'aide d'Arraylist.

```java
/**
 *
 * @author Thomas Jalabert <thom.jalabert@gmail.com>
 */
public abstract class AbstractControler {

    AbstractModel model;

    public AbstractControler(AbstractModel model) {
        this.model = model;
    }

    public void updateview(AbstractView view) {
        view.update();
        view.display();
    }

    public abstract void notifyview(AbstractView view);

}
```

```java
/**
 *
 * @author Thomas Jalabert <thom.jalabert@gmail.com>
 */
public abstract class AbstractView {

    AbstractModel model;
    AbstractControler controller;

    public AbstractView(AbstractModel model, AbstractControler controller) {
        this.model = model;
        this.controller = controller;
    }

    public abstract void display();

    public abstract void update();

    public void notifyController() {
        controller.notifyview(this);
    }
}
```

```java
/**
 *
 * @author Thomas Jalabert <thom.jalabert@gmail.com>
 */
public abstract class AbstractModel {

    public AbstractModel() {
    }


}
```

```java
/**
 *
 * @author Thomas Jalabert <thom.jalabert@gmail.com>
 */
public class Jeu extends Fenetre {

    List<AbstractModel> models;
    List<AbstractView> views;
    List<AbstractControler> controllers;
    int current;

    public Jeu() {
        models = new ArrayList<>();
        views = new ArrayList<>();
        controllers = new ArrayList<>();

        this.pack();
        this.requestFocusInWindow();
        this.setVisible(true);
        current = 0;
    }

    public void run() {

        while (this.isDisplayable()) {
            views.get(current).update();
            views.get(current).display();
        }
    }

}
```

La vue accède au modèle pour des accès en lecture et dois passer par le contrôleur pour des accès en écriture.

3. Quelles sont les spécificités des jeux DEVINT ? Est ce que la solution MVC respecte les besoins et si oui comment ? Comparez les 2 architectures et leurs avantages par rapport aux besoins de DEVINT.
    a. Quelles sont les spécificités des jeux DEVINT ?

Les jeux DEVINT sont amené à respecter certains critères. En effet les jeux sont crées en Java et en Swing. de plus, ils doivent utiliser la synthèse vocale SIVoX mis à disposition et obligent à utiliser des touches spécifiques pour que les utilisateurs puissent changer la couleur de l'interface ou bien réécouter les informations données par la synthèse vocale par exemple. Les jeux devint sont des jeux pour malvoyant ou non voyant, et développé dans un temps assez court. Ce sont donc des jeux peu gourmand en ressource graphique et processeur. Le travail réalisé est surtout axé sur l'ergonomie et la facilitée de prise en main du type d'utilisateur visé.

    b. Est ce que la solution MVC respecte les besoins et si oui comment ?

Une architecture MVC pourrait donc subvenir aux besoins du projet DEVINT puisque un modèle MVC permet de créer une architecture modulable et la création de plusieurs vues, modèles et contrôleurs. L'architecture MVC réduit légèrement les performances du jeu, cependant les jeux devint n'ont pas la nécessité d'être performant pour fonctionner.

    c. Comparez les 2 architectures et leurs avantages par rapport aux besoins de DEVINT.

Au final malgré le fait que le modèle MVC est utilisable pour les jeux DEVINT l'architecture présente des inconvénients tel que une certaine complexité de mis en oeuvre ce qui entraîne un manque de performance. On notera toutefois que c'est ses difficultés de mis en place qui permettent d'avoir une modularité dans le projet.
Mettre en place une autre architecture qui serait plus adapté au jeu tel que l'architecture gameloop est possible. Une architecture gameloop permet de mettre en place un système peu gourmand en ressources et donc permet de ne pas limiter les possibilités de jeu que l'on peut produire. Néanmoins la séparation entre la vue, le contrôleur et le modèle n'est pas présente dans ce pattern.