de la RED Thomas — PALUMBO Adrian
3 avril 2016

Rapport MCV

DeViNT — 2016

Rapport MVC

Dans ce rapport, nous allons répondre à différentes questions autour du modèle MVC et des projets DeViNT.

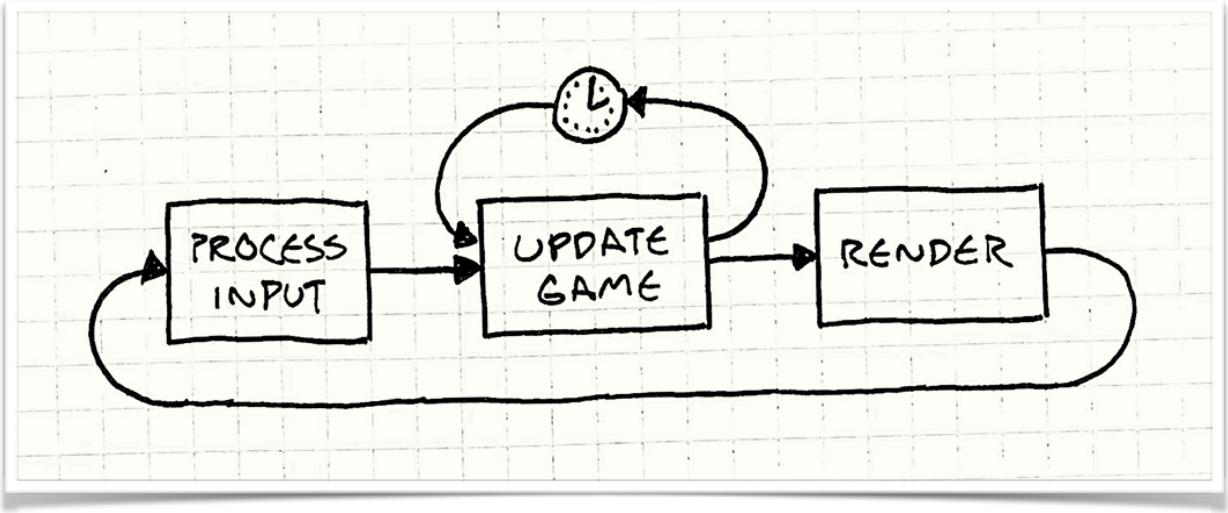
Question 1

– 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 ?

Les jeux DeViNT ne suivent pas l'architecture MVC mais l'architecture du Game Loop.



Modèle Game Loop

Ce modèle de jeu va simplement se charger de rafraichir la fenêtre toutes les *x* secondes, en appelant les méthodes « **update()** » et « **render()** ».

On remarque aisément que la classe « Jeu », par exemple, respecte totalement le principe du Game Loop et donc ne respecte pas du tout le modèle MVC ; on y retrouve les méthodes « **init()** », « **render()** », « **update()** ».

Et globalement, les jeux DeViNT se devant d'implémenter cette classe Jeu ainsi que ses méthodes, aucun ne respectera le modèle MVC.

```
public abstract class Jeu extends Fenetre {
    private static final long serialVersionUID = 1L;

    /**
     * Permet de creer et initialiser la fenetre de jeu
     */
    public Jeu() {...}

    /**
     * La loop du jeu qui permet de garder un FPS (frame pe
     */
    public void loop() {...}

    /**
     * méthode appelée à la création de l'objet
     * initialisation des composants graphiques et des vari
     * association des actions aux touches via les méthodes
     */
    public abstract void init();


    /**
     * méthode appelée à chaque tour de boucle
     * mise à jour des variables de jeu
     */
    public abstract void update();

    /**
     * méthode appelée à chaque tour de boucle
     * mise à jour des composants graphiques
     */
    public abstract void render();

    /**
     * méthode appelée avant la boucle
     * utilisée pour la réinitialisation
     */
    public abstract void reset();
}
```

Classe Jeu

En Game Loop, on ne peut pas « simplement » ajouter une vue par exemple. Il faut tout ajouter d'un coup, vu que rien n'est découpé, à l'inverse de MVC.

Question 2

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

– Donnez les parties de code à modifier e leurs modifications. Justifiez.

L'abréviation « MVC » signifie « Modèle Vue Contrôleur ». Ainsi, pour que le modèle de jeu respecte MVC, il faudrait clairement identifier ces trois parties.

Actuellement, en Game Loop, concernant les évènements, on « attend » qu'ils se produisent, on « écoute » en permanence afin de savoir s'il y a un évènement qui est survenu ou non.

En MVC, il faudrait faire différemment : on ne doit pas écouter indéfiniment mais chaque évènement doit notifier lui-même le contrôleur. Le contrôleur une fois notifié transmet le message au modèle (par exemple, il indique à un pion qu'il doit avancer). Enfin, le modèle modifie la vue.

Question 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.

Les jeux DeViNT ont plusieurs spécificités de part le nombre d'intervenants sur les projets.

En premier lieu, et bien entendu, les premiers concernés son les déficients visuels (et autres) qui joueront aux différents jeux.

Ces jeux, justement, sont développés par des élèves SI3 qui doivent pouvoir facilement utiliser un modèle de jeu, la synthèse vocale, etc.

Enfin, les enseignants doivent pouvoir aisément tout contrôler afin de garantir de bonnes livraisons au utilisateurs.

La solution MVC respecterait les besoins des projets DeViNT car la vue est séparée du reste. Ainsi, on pourrait créer différentes vues (selon les différentes déficiences) et en changer au moment voulu.

On ne peut pas dire « telle architecture est meilleure que l'autre », ça dépend vraiment des cas. Si on prend l'exemple de notre jeu, qui est un jeu de plateau, on peut facilement penser que la fenêtre n'a pas besoin d'être rafraichie toutes les *x* secondes. Ici, le modèle MVC conviendrait parfaitement : dès lors que l'on appuierait sur une des flèches directionnelles, on modifierai le modèle puis la vue.

Cependant, pour bien d'autres jeux, la fenêtre a constamment besoin d'être rechargée. Prenons l'exemple d'un casse brique : la balle bouge tout le temps et on ne peut donc pas utiliser le modèle MVC. On doit rafraichir la fenêtre à intervalles réguliers afin de constamment mettre à jour la trajectoire de la balle.

En conclusion, le modèle MVC comme le modèle actuel (Game Loop) trouvent tous deux leur intérêt dans de nombreux jeux. Il faut juste regarder et analyser les besoins de notre projet afin d'utiliser l'architecture la plus adaptée qui soit.