Patrons de conception - MVC

420-V31-SF – Programmation de Jeux Vidéo III

fparadis@cegep-ste-foy.qc.ca

MVC

MVC (ou Modèle-Vue-Contrôleur) est un patron de conception (design pattern) qui permet une séparation entre la vue, les interactions avec l'utilisateur et le modèle de données.

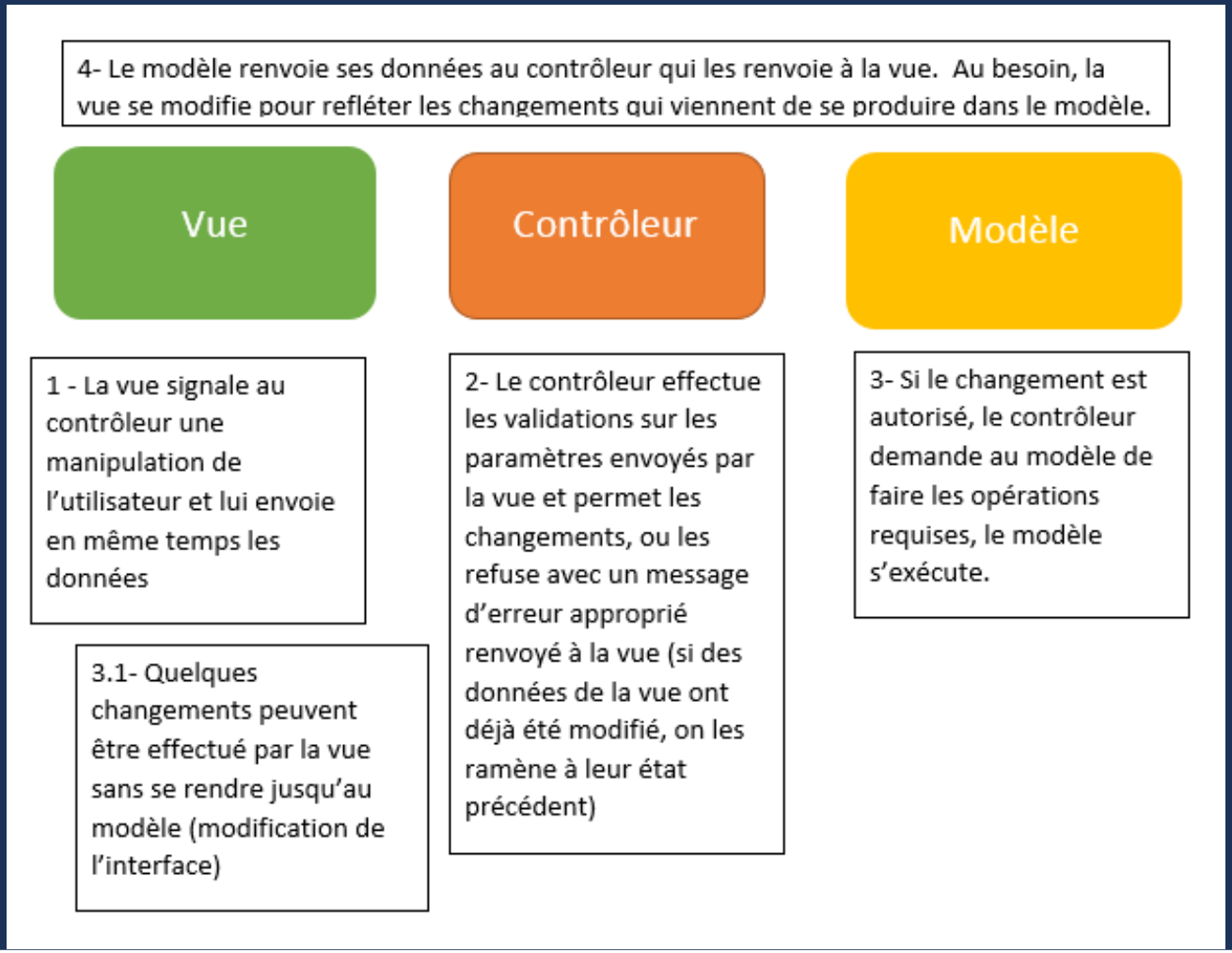
Aide à la compréhension et à la maintenance du code…

- *En réduisant la quantité de code dans un même fichier.*
- *En séparant des concepts bien précis dans des classes distinctes.*
- *Pour faciliter la création de test unitaires.*

Ce patron n'est pas parfait, mais Il amène quand même une structure intéressante. L'idée principale reste la séparation de la vue et des interactions utilisateurs, de la validation des données que l'utilisateur veut entrer dans le modèle de données et le modèle de données lui-même.

MVC

Les trois concepts de MVC sont la vue, le contrôleur et le modèle.



La vue

Son rôle est de:

1- Présenter l'information du modèle de données
2- Renvoyer les interactions de l'utilisateur vers le contrôleur.

Cette séparation permet d'avoir plusieurs vues sur un même modèle de données. La vue…

…n'effectue aucun traitement sur les données.

…redirige tous les événements utilisateurs vers le contrôleur.

…se met à jour lorsque le contrôleur lui permet de le faire.

Le contrôleur

Son rôle est de gérer les entrées de données provenant des manipulations de l'utilisateur. Selon le type de requête, il prendra la décision de modifier soit le modèle de données, soit la vue elle-même..

Comme la vue, le contrôleur n'effectue aucun traitement sur les données. Il ne fait qu'appeler les fonctions adéquates (sur la vue ou le modèle de données) selon les demandes de l'utilisateur.

Un événement utilisateur peut mener à la modification du modèle de données ou de la vue. Par exemple :

- Si l'utilisateur appuie sur le bouton "Supprimer" alors qu'une donnée quelconque est sélectionnée à l'écran, le contrôleur demandera au modèle de supprimer la donnée en question.

- Si l'utilisateur appuie sur un bouton "Afficher les options avancées", le contrôleur demandera à la vue de rendre visible ces options.

Le modèle

Le modèle regroupe toutes les structures de données ainsi que leur manipulation. C'est le cœur de toute application.

Tout traitement des données s'effectue à ce niveau.

Le modèle est la partie la plus autonome de ce patron de conception: il ne connait ni la vue, ni le contrôleur.

Le modèle suppose que les données qu'il reçoit sont valides et correct, puisque la validation se fait au niveau du contrôleur.

Puisque c'est le contrôleur qui appelle ses méthodes c'est à lui qu'il retourne le résultat.

La séparation des concepts

MVC n'est pas le seul patron de conception cherchant à définir comment séparer les diverses parties d'une application ou d'un système. Il y a entre autres MVVM (Model-View, View, Model) ou MVP (Model, View, Presenter).

Malgré certaines différences, tous recherchent à séparer la vue et le modèle. C'est la séparation **la plus importante** et qui apporte le plus d'avantages.

(En jeu, c'est le même principe. En théorie, tout jeu devrait pouvoir se dérouler conceptuellement, sans référer à l'affichage. L'affichage ne fait que refléter visuellement ce qui se trouve dans le modèle (en y ajoutant du "Groovy"). Cependant, avec la complexité grandissante des jeux modernes, la chose est de moins en moins possible.)

420-V31-SF – Programmation de Jeux Vidéo III

Notions d'animation 2D

fparadis@cegep-ste-foy.qc.ca

Notions d'animation 2D

À la base, fonctionne comme un dessin animé

On a une texture qui comporte l'ensemble des "poses" de notre animation. Le sprite n'utilise pas la texture complète, mais un "découpage" représentant une image parmi le lot

Normalement toutes les images ont la même taille

Le programmeur doit changer le découpage utilisé sur le sprite, au rythme approprié (animation lente ou rapide)

Notions d'animation 2D

Normalement, on cycle sur les image de l'animation. On va de l'animation 0 à l'animation n-1, puis, on revient à 0.

On parle ici d'animation cyclique.

Par contre il est aussi possible d'avoir des animations qui vont de 0 à n-1, puis retournent à 0. Selon le besoin, ça repart à nouveau (animation idle) ou on en reste là (animation de coup porté).

On parle ici d'animation en va et vient
ou animation en pendule

Notions d'animation 2D

Normalement les images d'une même animation sont placées de manière continue en X.

On parle ici de nombre d'animations ou de nombre de frames.

Le même personnage peut avoir plusieurs animations. Celles-ci sont placés une par-dessus l'autre en Y. Comme s'ils étaient en étage.

C'est pour cela qu'on parle du niveau de l'animation, celle étant le plus proche du haut étant 0, comme un écran le 0 en Y est en haut.

"Découpage" de la sprite

Pour connaître le nombre d'animation et de niveau, on doit aller voir la spritesheet.

Pas vraiment d'autre moyen, sauf si les sprites sont très standardisés d'avance.

Pour le Galaga

Nous avions 2 animations.

Nous avions 3 niveaux d'animations

Une pour chaque galaga.

Un personnage en vue isométrique (faux 3D vu de haut) qu'on déplace avec une croix directionnelle aurait combien de niveaux d'animation?

Pour le Galaga

L'animation d'un galaga change au 6 fois par seconde.

Puis, si on divise notre variable d'animation par 10, elle sera pair ou impair. Donc le résultat du modulo sera 0 ou 1, exactement ce qu'on aura besoin puisqu'on a deux cadres.

Gérer les animations

Pour une animation cyclique, aller simplement de l'image 0 à n-1, puis revenir à 0.

À vous de trouver la vitesse de déroulement pour que ça soit fluide et beau.

Déterminez un nombre de "frame d'animations" et déterminez le cadre affiché avec [frame d'animations] % [Nombre d'animation].

Plus le nombre de frames est bas, plus l'animation est rapide.

Une fois au bout de votre cycle, revenez à 0 (ceci dit une opération par modulo fera ça tout seul)

Gérer les animations

Pour les animation en va et vient, ajoutez toujours un compteur qui vaudrait 1 (par exemple)

Quand vous êtes au bout de n-1, multipliez votre compteur par -1, même chose quand vous êtes à 0.

Et dans tous les cas, ajoutez toujours votre compteur au total de frame en cours, vous ferez +1 ou -1 selon la direction.

Pour le déplacement d'un personnage dans 8 directions

La direction du personnage décidera du niveau de l'animation (hauteur dans la spritesheet)

Mais le compteur d'animation, lui devrait continuer normalement.