Architecture modèle-vue-contrôleur (MVC)

- Architecture logicielle (design pattern) qui permet de séparer :
  - les objets applicatifs(Model)
  - leur représentation (View)
  - les interactions qu'ils subissent (Controler)
- Utilisée pour la programmation de client/serveur, d'interface graphique
- Généralisation du pattern Observer/observable vu en algo avancé

Architecture MVC en général

- Schéma de programmation qui permet de séparer une application en 3 parties :

Côté applicatif

- Le modèle contient la logique de l'application

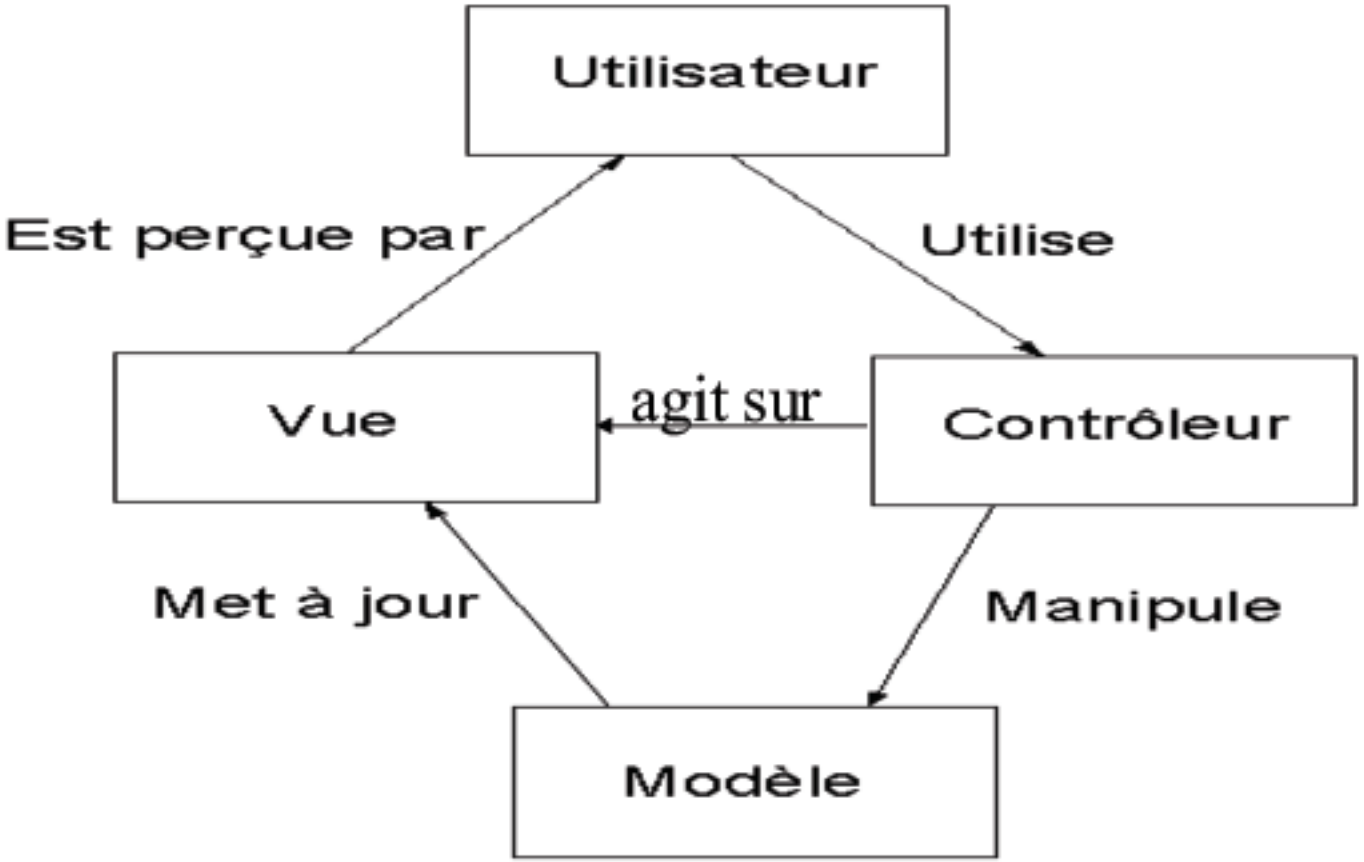
Côté visuel

- Les vues affichent à l'utilisateur des informations sur le modèle

Côté événementiel

- Le contrôleur agit sur demande de l'utilisateur et effectue les actions nécessaires sur le modèle.

Aperçu


Utilisateur
Est perçue par
Utilise
Vue
agit sur
Contrôleur
Met à jour
Manipule
Modèle

architecture MVC : exemple1

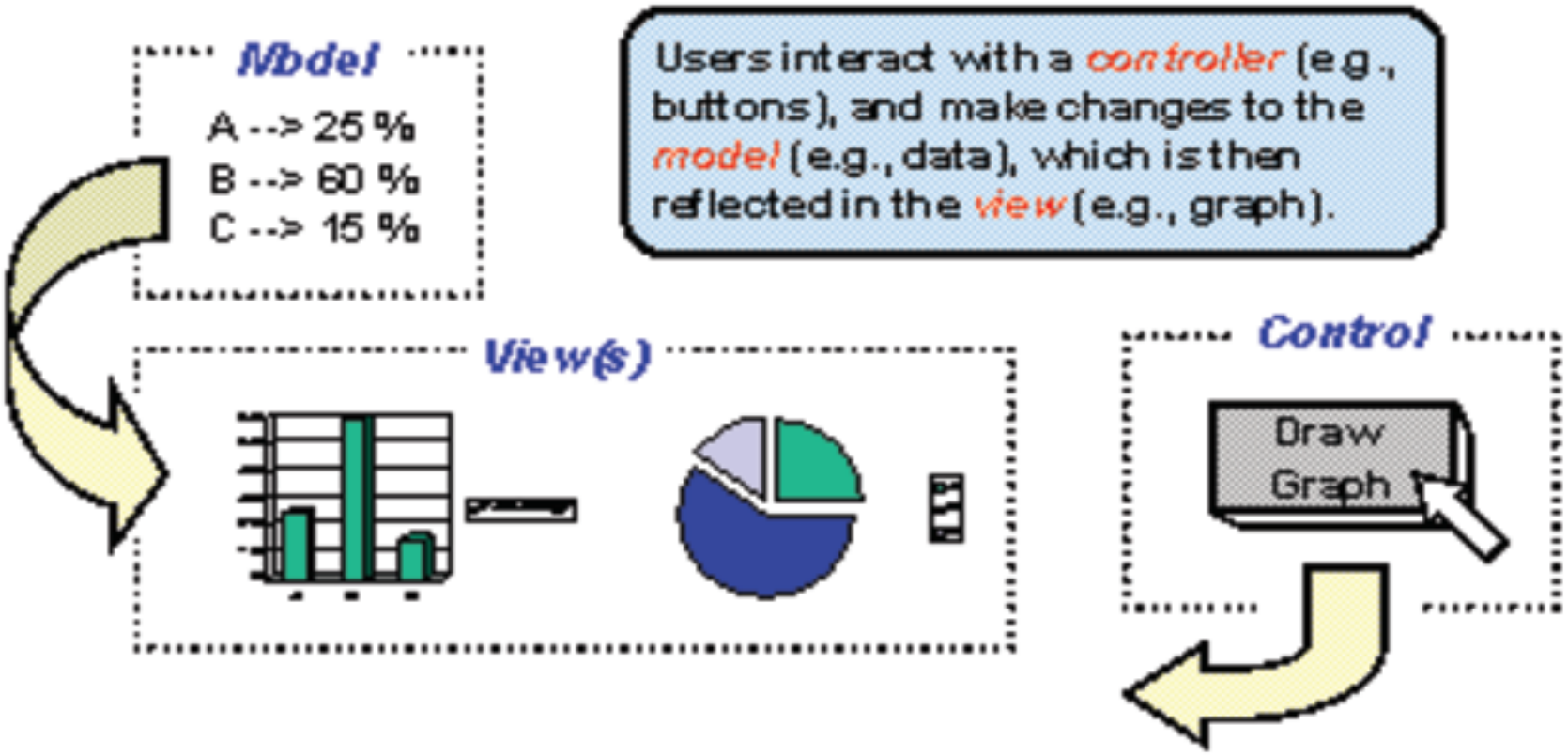
- Document texte (modèle : chaîne de caractères html) peut avoir deux vues :
  - une vue WYSIWYG (What You See is What you Get) vue par un navigateur
  - une vue brute : code html source avec balises vue par un éditeur de textes

- Toute modification sur le modèle se répercute dans les deux vues

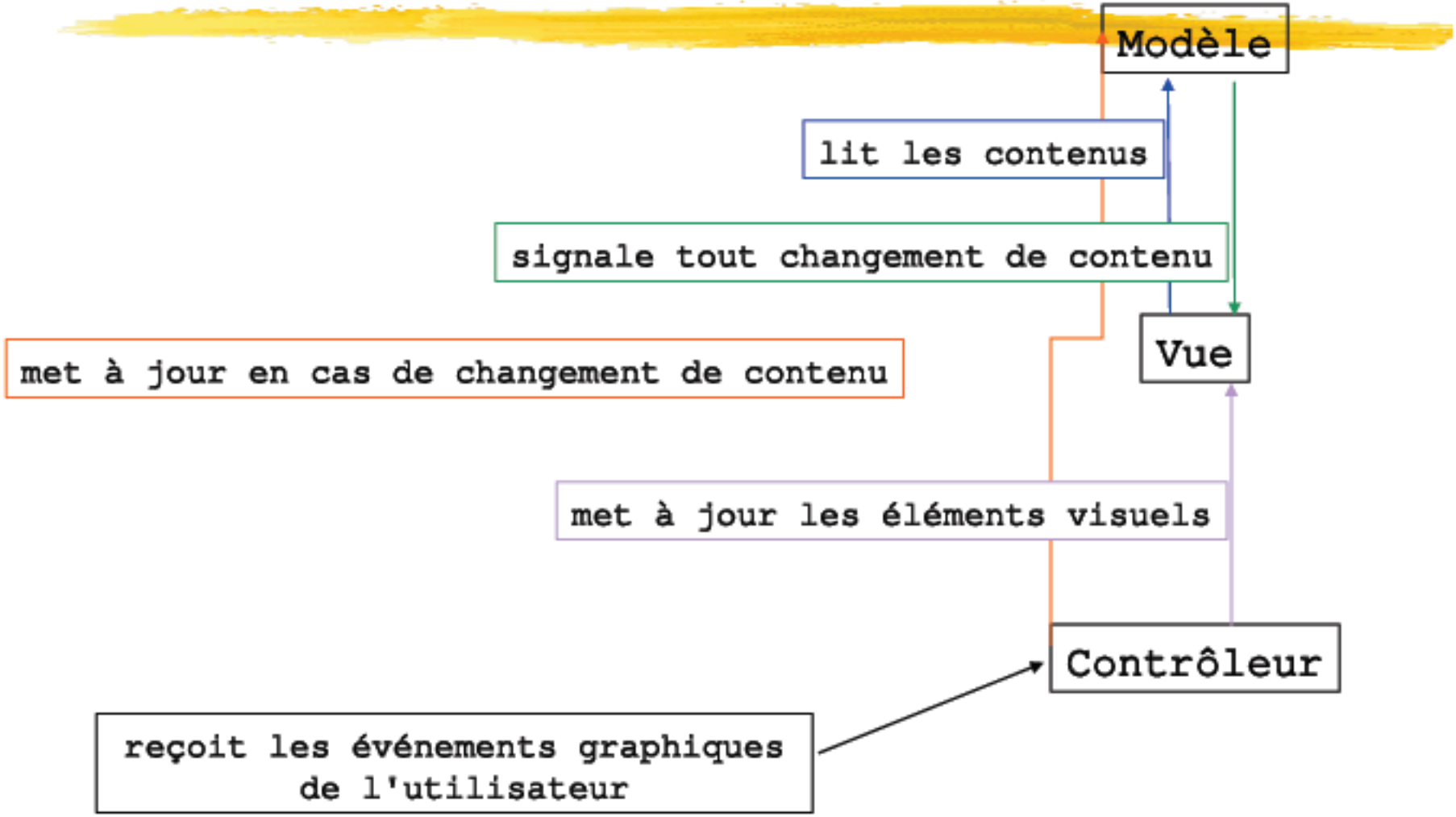
architecture MVC : exemple2

- Application de calcul du budget d'une société
  - le modèle : les données et le procédures de calcul pour le budget prévisionnel de la société
  - les vues : une représentation sous forme de tableau, une sous forme graphique, une vue pour la direction, une pour la DRH,...
  - le contrôleur : des éléments de l'interface pour faire varier les paramètres du calcul

Exemple 3 : différentes vues sur un ensemble de pourcentages



Interactions entre le modèle, la vue et le contrôleur



Rôle des 3 éléments : le contrôleur

- Reçoit les événements de l'interface utilisateur
- Les traduit :
  - en changement dans la vue s'ils agissent sur le côté visuel
    - exemple : clic dans l'ascenseur de la fenêtre
  - en changement dans le modèle s'ils agissent sur le contenu du modèle

    exemple : on réajuste le camembert, on modifie le diagramme en bâtons en agissant dans la vue

Rôle des 3 éléments : le modèle

- Il peut être modifié sur ordre du contrôleur
- Il signale à ses vues tout changement de contenu en leur envoyant un événement qui leur spécifie de se mettre à jour
- Mais il ignore :
  - comment il est affiché
  - qui lui a notifié un changement d'état

Rôle des 3 éléments : la vue

- La vue se met à jour dès qu'elle reçoit un ordre de notification
  - du contrôleur
  - du modèle

- Quand la notification vient du modèle, elle va consulter le modèle pour se réafficher de manière adéquate

Intérêts de l'architecture MVC

- **Indépendance** entre :
  - la représentation logique d'une application (modèle)
  - la représentation visuelle qu'on en donne (vue)
  - les actions que l 'utilisateur effectue (contrôleur)

⇒ **Séparation** claire entre les données du programme et l'interface graphique affichant ces données

⇒ **Possibilités de vues différentes d'un même modèle**

L'application peut montrer l'état du modèle de différentes façons, avec différentes interfaces utilisateurs

Intérêts de l'architecture MVC

- Cette indépendance favorise le développement et la maintenance des applications :
  - Modularité dans la conception
    - vue et contrôleur peuvent être développés indépendamment du modèle (pourvu qu'une interface entre les deux soit définie)
  - Meilleure répartition des tâches
    - développeurs du modèle/développeurs de l'interface ont des compétences différentes
      - développeurs du modèle : connaissance métier
      - développeurs de l 'interface : connaissance des besoins utilisateurs, souci d 'ergonomie...

Architecture MVC et composants swing

- Chaque composant swing s'appuie sur une architecture MVC en l'adaptant : la vue et le contrôleur sont combinés en un même objet
- Pour beaucoup de composants, cette architecture n'est pas exploitée (JButton,…)
- Pour d'autres, elle est très importante : JList, JTable
- L'architecture MVC fonctionne en utilisant la notion d'événement

Fonctionnement du MVC avec la notion d'événement notifié aux vues

- Le modèle doit connaître les vues qui dépendent de lui

- **Dès que le modèle change,**
  - il crée un événement correspondant à ce changement
  - il envoie cet événement à toutes ses vues
  - ses vues, à la notification de l'événement, réagissent en conséquence, en mettant à jour leur apparence visuelle.

Implémentation en java des modèles

- Tous les composants swing reposent sur le même principe:
  - Les modèles sont des interfaces `XXModel`
  - Des classes abstraites `AbstractXXModel` implémentent l'interface en fournissant le code de certaines méthodes
    - de gestion du modèle (modification,…),
    - d'ajouts de vue
    - des méthodes de notification d'événements
  - Le modèle mémorise l'ensemble de ses vues

Implémentation en java des vues/contrôleurs

- Les vues sont interfaces ZZListener correspondant à des écouteurs d'événements
- Les événements correspondant à des changements dans le modèle sont des YYEvent
- Dès qu'un tel événement est notifié à une vue, elle réagit en déclenchant la méthode appropriée
- Les vues doivent connaître le modèle

vue sur un modèle = composant qui s'est déclaré comme écouteur d'événements de changements sur le modèle

En général : comment réaliser un MVC en java

- Définir un modèle
- Définir le composant qui sera une vue sur ce modèle :
  - Il doit se déclarer écouteur des événements de changement dans le modèle en implémentant l'interface adéquate
  - Il doit mémoriser le modèle en variable membre
  - Il doit donner un corps à chaque méthode de l'interface
  - Il doit se faire ajouter à la liste des écouteurs du modèle