Projet de Programmation Comparée : "Interfaces Utilisateurs"

5 mars 2013

1 Qu'est ce qu'une interface utilisateur et comment en programmer

1.1 Définition des concepts liés à la notion d'interface utilisateur

Qui sont les utilisateurs ?

- Deux catégories : nous utiliserons suivante afin de distinguer les deux catégories d'utilisateurs : utilisateur application, et programmeur.
- Dans les deux cas, il faut avoir une approche orientée utilisateur, les besoins seront différents, mais l'approche à adopter la même.
- Utilisateur application : il faut prendre en compte à la fois l'aspect sensoriel (visuel si écran il y a) et l'aspect "contrôle" (gestion des périphériques d'entrée).
- Interaction utilisateur application - programme : plusieurs "niveaux" de compétences parmi les utilisateurs avec des besoins différents.

Conception - génie logiciel

Utilisation de la conception centrée utilisateur qui est un modèle itératif.

- **Analyse :** analyse des besoins, un panel représentatif d'utilisateurs des deux catégories concernées doit être constitué afin d'établir les dits besoins.
- **Conception :** un prototype doit être conçu en fonction des besoins établis à l'étape précédente.
- **Évaluation :** sur la base du prototype réalisé, une évaluation est faite. Le procédé étant itératif, cette évaluation servira de base à la modification des besoins de la première étape.
- Il faut établir des "topic" d'évaluation - besoins, les deux sont liés :
  - Pour l'utilisateur application :
    - Vitesse d'apprentissage - aide nécessaire / intégrée - nombre d'erreurs commises lors d'un test.
    - Correction des erreurs.
    - Temps de réponse.
    - Efficacité - navigation rapide ?
  - Pour le programmeur :
    - Programmation intuitive.

  - Bonne expressivité - ne pas avoir un code verbeux à produire (capacité d'adaptation au support etc etc ?).
  - Typage fort - sureté.
  - Debugging aisé - analyse de pertinence (détecter le maximum d'absurdités)
  - Intégration aux IDE populaires ? Création d'un IDE ?
- **Les outils :** des outils du génie logiciel peuvent nous aider à l'intérieur des phases décrites précédemment :
  - Scénarios / diagrammes de cas d'utilisation : à la fois pour établir les besoins et faire l'évaluation au niveau de l'utilisateur application. Peut aussi être utilisé pour le côté programmeur bien que le point suivant me semble être plus approprié.
  - Diagrammes relationnels - diagrammes de classes : relations entre les différents objets de l'interface, est utile dans la phase de conception pour "se mettre à la place" du programmeur.
  - Design pattern : beaucoup d'interfaces utilisateurs utilisent des principes similaires d'interaction avec l'utilisateur, l'implémentation "native" de comportements génériques peut être pratique.

1.2 Division du travail

La conception d'une interface graphique par dessus un moteur d'application peut se diviser en quatre parties qui devraient demeurer indépendantes le plus possible.

- Communication entre le moteur et l'interface :
  - Affichage : ensemble des données du moteur affichées (sous diverse forme) par l'interface ;
  - Actions : ensemble des actions de l'utilisateur modifiant l'état du moteur. Tout élément de l'interface permettant d'agir sur le moteur devrait être lié à une "Action". Une même Action doit pouvoir être effectuée par différents éléments d'une interface, et par différentes interfaces.

  Cette section doit être indépendante de l'interface finale (éléments, aspect, etc) et de la plateforme.
- Éléments de l'interface ; dépendant de la plateforme, lié à la partie précédente.
- Personnalisation des éléments : positionnement, taille, aspect...
  Une telle configuration est évidemment dépendante de la partie précédente.
  Certains paramètres de cette configuration doivent pouvoir être modifiés par l'utilisateur final.
  Ces configurations doivent pouvoir être enregistrées et s'échanger facilement.
- Aspect général.

2 Analyse de l'existant

Analyse de Swing / GTK (qui ont évidement des problèmes insurmontables) ainsi que du couple HTML CSS et le concept intéressant de la séparation du style de la déclaration des objets.

3 Outils à développer

3.1 Bindings

Liaison d'une propriété d'un élément de l'interface graphique à une valeur du moteur de l'application : par exemple, la "clickabilité" (oui) d'un bouton à un booléen, le champs d'un label à une

chaîne de caractères, la valeur d'un curseur à une valeur du moteur logique, le contenu d'un panel à une image...

La liaison doit être effective dans les deux sens : le changement du champs du label par l'utilisateur doit modifier la valeur de la chaîne de caractère, tout changement tiers de la valeur doit être répercuté dans le label de l'interface.

Solution : la librairie fournit à l'intention du moteur de l'application des classes pour des types simples comprenant un pattern observer à destination des éléments de l'interface.

3.2 Actions

Je prépare juste la sous-section des actions car je vais certainement en avoir besoin pour le modèle relationnel présenté dessous.

3.3 Le modèle relationnel

Les librairies d'interfaces graphiques populaires ne fournissent pas de moyen haut-niveau de décrire les relations entre les différents éléments de l'interface utilisateur. Seule une relation de parenté purement liée au placement des objets les uns dans les autres (le contenant est automatiquement parent du contenu) est possible, tout autre système de mise en relation entre les éléments n'est pas directement supporté.

Il serait cependant intéressant de pouvoir définir plus finement les relations entre ces différents éléments, plus précisément, l'idée serait de ne pas avoir à décrire l'emplacement d'un élément par rapport à un autre dans le "code principal" mais juste les relations entre ces éléments, un peu à la manière du couple HTML / CSS, une "feuille style" dépendant du support / des préférences de l'utilisateur application / du système (cf. langages intermédiaires) se charge d'associer une relation à un emplacement / visuel / un lien.

3.4 Langages intermédiaires

Plus approprié ici, à compléter entre autre quand l'idée d'un interpréteur pour le visuel de l'interface aura été tranchée.