Feuille de TD n°8 :

Exercice de synthèse : modélisation UML / Programmation événementielle / C++ / bibliothèque wxWidgets

Objectif pédagogique

Cette feuille de TD a pour but proposer un exercice récapitulatif :

- Mettant en œuvre les enseignements de ce module en lien avec la modélisation objet : cas d'utilisation => diagramme de classes => diagramme états-transitions => conception d'une interface ergonomique => Traduction de l'analyse sous la forme d'un programme événementiel à interface graphique avec C++ et wxWidgets.
- Permettant aussi de se préparer au travail de conception et de développement en équipe

Ressources nécessaires - Consignes

~ Le cours d'ihm, + extrait de documentation de wxWidgets
~ Le cours d'UML
~ Cette feuille de TD n°8

SUJET :

Il s'agit de réaliser un jeu de memory **simple**, précédé d'une rapide analyse UML.

SPECIFICATIONS EXTERNES DU JEU DE MEMORY :

- Lorsque l'utilisateur lance le programme, le plateau se remplit d'un nombre pair de 'cartes' disposées en lignes et colonnes, faces retournées. Les cartes disposent d'un côté 'face' représentant une image et d'un côté 'dos' neutre. Exple : cf. figures ci-contre, 2 cartes 'faces' visibles. Les cartes viennent par paires : 2 cartes ayant le coté 'face' identique.[1]
- L'utilisateur dispose d'un temps limité pour reconstituer les paires. Dans ce temps limité, il doit tenter de retrouver le maximum de paires.
- Le mode opératoire est le suivant : toutes les cartes sont posées, faces retournées. L'utilisateur retourne une carte, puis une deuxième. Si les 2 cartes ont la même image, elles sont retirées du plateau (par le système) et le point est attribué au joueur. Si les 2 cartes sont différentes, elles sont retournées (par le système) et le joueur peut recommencer. Le jeu s'arrête lorsque le temps alloué est terminé ou qu'il n'y plus de carte sur le tableau… ou encore lorsque le joueur abandonne.
- A tout moment, le joueur est informé du temps écoulé (ou restant), et du nombre de paires qu'il a dévoilées.
- Le jeu démarre lorsque l'utilisateur déclenche le compte-à-rebours.

TRAVAIL À FAIRE :

- Vous travaillerez par groupes de 3 à l'intérieur de votre groupe de TP.
- Le travail s'organisera autour des étapes décrites ci-dessous. Le livrable final à rendre sera constitué des sources de l'application développé et de la documentation décrite ci-dessous.

0. **Spécifications externes du produit réalisé.**
   Préciser **uniquement** les points qui vous ont semblé flous ou bien incomplets.

[1] .extensions possibles : l'utilisateur peut choisir le jeu de 'cartes' cad les images des cartes qu'il utilisera, il peut aussi choisir la difficulté du jeu, en termes de nombre de lignes/colonnes du plateau, ou encore s'il souhaite constituer des PAIRES, des TRIPLETS ou QUADRIPLETS de cartes. On pourrait aussi imaginer que le joueur ne joue pas seul, mais contre l'ordinateur, qu'il peut interrompre et sauvegarder la partie pour continuer plus tard… Le système peut garder mémoire des meilleurs scores établis….

1. **Scénario nominal**
   Décrire sous forme textuelle le scénario nominal d'utilisation de ce jeu correspondant au cas d'utilisation fourni en Annexe1.
   Les **messages** à destination du système seront bien mis en évidence (par exemple soulignés) de sorte à faciliter l'identification des **méthodes** (préparation pour la question 2.-)

2. **Diagramme de classe (UML)**
   (a) Élaborer ***manuellement*** *(*sans atelier !*)* le diagramme de classes UML du jeu Memory, en se focalisant sur les classes **métier**, c'est à dire celles décrivant le plateau et les cartes, indépendamment des éléments d'interface.
   (b) Établir succinctement le dictionnaire des éléments et des méthodes :
   - nom, type et signification pour les éléments
   - nom, but, nom-type des paramètres et type de la valeur résultante pour les méthodes

3. **Diagramme états-transitions**
   (a) Lister les différentes valeurs d'**états du jeu**.
   (b) À partir de cette liste, établir le **diagramme états-transitions** du jeu
   (c) Établir le dictionnaire des états (nom état, signification) et le dictionnaire des événements (nom événement, signification) correspondants
   (d) Élaborer la table **`T_EtatsEvenementsJeu`** correspondant à la version matricielle du diagramme états-transitions prédédent :
   - en *ligne : les **événements*** faisant changer le jeu d'état
   - en *colonne : les **états*** du jeu
   Un exemple est fourni en Annexe 1.

4. **Interface**
   (a) Compléter la table **`T_ EtatsEvenementsJeu`** avec les noms des objets qui généreront ces événements
   (b) À l'aide du générateur d'interfaces wxSmith, créer l'interface de l'application à partir des éléments d'interface cités précédemment
   (c) Dans votre documentation projet, établir le dictionnaire des éléments d'interface : nom, type, rôle dans le programme, événements du dictionnaire des événements auxquels ils sont associés
   (d) Établir une fiche descriptive de l'interface associée à état de l'application (du jeu) :
   - nom de l'état
   - état/propriétés de chacun des objets d'interface pour cet état de l'application (du four) :
     actif/inactif, visible/non visible, focus,…

5. **Spécifications internes du programme réalisé**
   (a) Organisation du code :
   - liste des classes contenue dans le projet, et, si nécessaire, description minimale de chacune d'elles
   - liste des fichiers présents dans le projet, et, si nécessaire, description minimale de chacun d'eux – éléments d'information complémentaires sur l'organisation du code
   (b) *Si cela est nécessaire*, lister des éléments du programme autres que ceux déjà décrits dans les questions précédentes :
   - définition de types
   - variables / constantes : nom, type, signification
   (c) *Si cela est nécessaire*, décrire succinctement les sous-programmes utilisés autres que ceux déjà décrits dans les questions précédentes
   - nom, éventuellement type de la valeur retournée, liste des paramètres et but

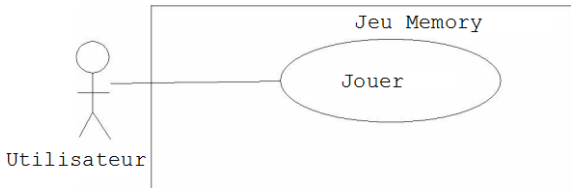
6. **Bibliographie / webographie**

7. **Bilan des activités**
   Temps passé (en heures/groupe, y compris le temps prévu à l'edt) , apprentissages/pratiques à retenir pour de futurs projets.

8. **Annexe :** Code source du projet, dûment documenté

ANNEXE 1 - DOCUMENTS DISPONIBLES

1. Cas d'Utilisation du système



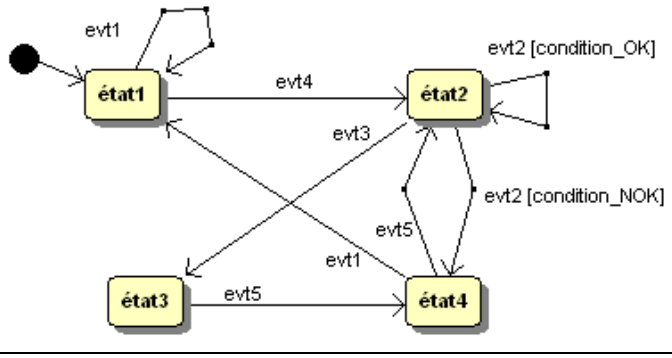
2. Scénario nominal

Titre : Chauffer un aliment
Résumé : L'utilisateur souhaite chauffer son aliment. **Acteur** : Utilisateur (acteur principal)
Pré-condition : Le four à micro-ondes est sous tension, porte fermée et minuterie à 0.
Post-condition : néant
Date de création : 10/09/2013 **Date de mise à jour** : 23/05/2014 **Version** : 1 .5
Créateur : …

Utilisateur	Système
1. Ouvre la porte.	2. Allume la lumière.
3. Règle la minuterie.	4. Indique le temps demandé.
5. Ferme la porte.	6. Éteint la lumière.
7. Démarre la cuisson.	8. Allume la lumière et met en marche le canon.
	9. Quand la minuterie arrive à 0, le four éteint la lumière et le canon, et émet un « bip » sonore.
10. Ouvre la porte.	11. Allume la lumière.
12. Ferme la porte	13. Éteint la lumière.

3. Élaboration de la version matricielle d'un diagramme états-transitions

Considérons le diagramme états-transitions suivant :



La version matricielle de ce diagramme états-transitions est la suivante :

Événements → / Etats ↓	évt1	évt2	évt3	évt4	évt5
état1	état1			état2	
état2		[condition_OK] état2 [condition_NOK] état4	état3		
état3					état4
état4	état1				état2

Ce tableau se lit de la manière suivante, par exemple, pour la ligne de l'état2 :
Depuis l'état2 :

- Si l'événement evt3 a lieu, le système reste à l'état3
- Si l'événement evt2 a lieu ET lorsque la garde est vérifiée (**condition OK**), le système reste à l'état2

- Si l'événement evt2 a lieu ET lorsque la garde est vérifiée (**condition NON OK**), le système passe à l'état4

L'étape suivante consiste à associer des éléments graphiques de l'interface aux événements identifiés par l'analyse UML.

Éléments d'interface déclencheurs des événements →	**bouton_X**	**bouton_Y**		**molette_Z**	**timer_T**
Événements → / **Etats ↓**	**évt1**	**évt2**	**évt3**	**évt4**	**évt5**
état1	état1			état2	
état2		[condition_OK] état2 [condition_NOK] état4	état3		
état3					état4
état4	état1				état2

Comme on peut le remarquer, un même élément graphique de l'interface peut prendre en charge un ou plusieurs événements. Dans cet exemple :

- bouton_X prend en charge l'événement évt1 : il reste à préciser quelle action sur bouton_X déclenchera évt1. Cela peut être clic, double-clic, focus, …. ..à définir selon la liste d'événements associés à chaque élément d'interface.
- bouton_Y prend en charge les événements évt2 et évt3 (par exemple, clic associé à évt2 et double-clic associé à évt3)

Cette étape rapproche l'analyse de la programmation en gardant la cohérence globale de la démarche.

Scénario

Titre :
Résumé : . **Acteur** : Utilisateur (acteur principal)
Pré-condition :
Post-condition :
Date de création : **Date de mise à jour** : **Version** : .
Créateur :

Utilisateur	Système

Dictionnaire des Classes : attributs et méthodes

Classe XXX

Nom attribut	Signification	Type	Taille	Exemple

Entête méthode	But

Diagramme états-transitions

Dictionnaire des états du jeu (=du système)

Nom état	Signification

Dictionnaire des événements faisant changer le jeu (=système) d'état

Nom événement	Signification

Version matricielle du Diagramme états-transitions

événement *nomEtatJeu*	Evt1	Evt2	Evt3	Evt4	DecAutoMin
Etat1	--	Etat2	--	--	[tempsJoueur> 0] Etat1 --------- [tempsJoueur = 0] etat4
Etat2	--	--	[tempsJoueur > 0] Etat3 ---------- [tempsJoueur = 0] Etat3	Etat2	--
Etat3	Etat1	Etat2	--	[temps > 0] Etat3 ---------- [temps = 0] Etat4	--
Etat4	--	Etat2	--	--	--

Liste des éléments d'interface

Dictionnaire des éléments d'interface

Type	**Nom**	**Signification**	**Evénements générés réagit**

Description de l'interface de l'application – état par état

Cette description peut être textuelle (cf. texte ci-dessous).
Elle peut aussi être traitée en faisant des copies d'écran de l'interface à partir de simulations réalisées avec wxSmith.

Nom état 1

Elément d'interface	**Description** (visible / invisible ; activé / inactif ; focus ; …)

Nom état 2

Elément d'interface	**Description** (visible / invisible ; activé / inactif ; focus ; …)