Les projets informatiques

Didier Rémy
2001 - 2002

http://cristal.inria.fr/~remy/mot/20/
http://www.enseignement.polytechnique.fr/profs/informatique/Didier.Remy/mot/20/

Calendrier des cours

Quand

- Les mercredi 14, 21, 28 novembre et 5 décembre
  le matin de 10h15 à 12h15.
- Soutenances des projets et les 17-19 décembre.

Présentation générale

Le projet est dédié à la mise en pratique du cours autour d'un problème de taille réaliste et raisonable. Il sert à se confronter à la pratique, mais aussi à confronter la théorie à la pratique.

Le projet

Dans tous les cas, votre implémentation devra être modulaire...

Deux des projets sont directement la mise en œuvre de la modularité : les modules et les classes de façon très poussée et complémentaire. L'un accentue l'approche par modules et foncteurs, l'autre l'appproche à objets.

Le troisième projet est une exploration les langages à objets de l'intérieur : l'implémentation d'un petit prototype.

Le cours accompagne le projet par des informations ciblées vers les projets retenus.

Travail d'implémentation

Dans les deux types de projets qui consistent en l'écriture d'une librairie et d'une ou plusieurs petites applications utilisant cette librairie, vous jouerez trois le rôle : celui d'implémenteur (écriture de la librairie) et d'utilisateur expert (adaptation de la librairie à une application particulière) et d'utilisateur final (écriture de l'application).

Le travail principal est dans l'écriture de la librairie. Celle-ci doit être modulaire dans son implémentation et simultanément permettre une utilisation modulaire. Les deux types de modularité sont différents et il est important de bien les dissocier dans la conception de la librairie.

En particulier, le rôle de l'utilisateur expert permet de démontrer que la librairie reste extensible.

Les applications de la librairie servent à montrer l'utilisation de la librairie (sorte de mini-tutorial) en insistant sur ses fonctionnalités et son extensibilité. Cette partie est a priori un investissement moindre que la partie précédente, mais on peut très bien envisagé une application importante aussi.

L'efficacité n'est pas un critère prioritaire. D'ailleurs l'implémentation modulaire doit permettre de remplacer ultérieurement une implémentation par une autre plus efficace. Il est seulement important de ne pas faire des choix qui seraient une source d'inefficacité structurelle, *i.e.* qui ne pourrait pas être corrigé par la suite.

Le plus important, et le plus difficile sans doute, est la conception de l'architecture. Une étude comparative de différentes structures, qui ne figurera pas dans le code final, peut toutefois faire partie intégrante du projet et être présentée.

Librairie de calcul formel

Il s'agit d'écrire une librairie extensible pour le calcul formel qui devra permettre le calcul dans les groupes, anneaux, corps, espaces vectoriels, polynômes, etc. construits à partir des entiers, des flottants, des nombres en précision arbitraire, etc.

La librairie devra être modulaire dans son écriture et dans son utilisation. En particulier elle doit pouvoir être étendue par l'utilisateur avec de nouvelles structures mathématiques en s'appuyant sur les structures existantes.

La librairie devra également présenter suffisamment de sécurité, par exemple empêcher d'effectuer des opérations non permises par la structure mathématique.

Elle devra aussi pouvoir être spécialisée (par dans des structures plus riches certaines opérations implémentées par défaut possèdent une implémentation plus efficace)

Librairie de calcul formel (application)

On utilisera la librairie sur de petits calculs pour illustrer ses structures les plus riches et sa flexibilité.

La modularité de l'implémentation et de l'utilisation est l'aspect essentiel. L'efficacité des algorithmes implantés est secondaire, mais il faut éviter une inefficacité structurelle. C'est-à-dire, il devra être possible de remplacer les implémentations inefficaces par des algorithmes efficaces sans devoir changer la structure.

Le plus important, et le plus difficile sans doute, est la conception de l'architecture. Une étude comparative de différente structures, qui ne figurera pas dans le code final, peut toutefois faire partie intégrante du projet et être présentée.

Librairie de calcul formel (application)

Groupes

Montrer que les entiers qui sont la somme de 8 carrés forment un groupe multiplicatif.

Calcul sur les polynômes

Géométrie

Vérifier que l'orthocentre d'un triangle est bien défini.

Calcul sur les polynômes fractionnels

Librairie de composants graphiques

Le projet consiste à écrire une librairie de composants graphiques permettant ensuite à l'utilisateur de se programmer rapidement de petites applications graphiques.

La librairie

Écrire des composantes graphiques élémentaires : buttons, menus, images, texte ainsi que quelques composants d'assemblage (vertical, horizontal), etc. permettant d'écrire de petites applications graphiques.

La choix des composants est assez libre, ce qui peut permettre d'orienté le travail dans des directions assez différentes, allant par exemple d'un petit gestionnaire de fenêtres à un éditeur de dessins.

Il existe plusieurs protocoles intéressants de communication entre les widgets. Il n'y a donc pas de solution unique, loin de là, et la variété des modèles rend l'exercice plus intéressant. Il est cependant important de bien définir le modèle et de le respecter dans l'ensemble de la librairie.

Pour le dessin, on se contentera de la fenêtre (et des primitives) graphique de base.

On illustrera les fonctionnalités de la librairie et sa flexibilité sur quelques petits exemples.

Finder

Cette application peut être vu comme un composant "finder" ou comme un gestionnaire de fenêtres à la macintosh. Les fichiers et dossiers sont représentés par des icônes. Les dossiers peuvent être ouverts pour montrer leur contenu. Un fichiers ou un dossier fermé peut être retiré à la souris du dossier qui le contient et glissé dans un autre dossier ouvert ou sur sont icône.

On peut réarranger la fenêtre en déplaçant les icônes, renommer un fichier, créer de nouveaux fichiers et dossiers vides, etc.

Dans l'approche gestionnaire de fenêtre, on peut aussi glisser un fichier sur un fichier exécutable pour lancer l'application correspondante avec le premier fichier comme paramètre.

Gestionnaire de fenêtres

C'est le même projet mais on s'intéresse plus au gestionnaire des composants qu'aux composants eux-mêmes, donc à la façon de gerer de nouveux composant plustôt qu'à la façon de les assembler. Cette fois-ci, le gestionnaire plutôt que les composants devra pourvoir être étendu et personalisé.

Le composant d'affichage d'arbres à tiroir

Cette application peut-être vue elle-même comme un sous-projet avec les mêmes soucis de modularité : écrire une librairie d'affichage d'arbre à tiroir (on montre et on cache les sous-arbres en cliquant sur les nœuds).

À nouveau, ce sous-projet doit être écrit de façon modulaire, mais aussi et surtout permette à l'utilisateur de personnaliser la librairie de base par rapport à ses besoins. Cette personnalisation est à deux niveaux : paramétrer l'affichage des noeuds, offrir d'autres actions propres aux nœuds, etc., mais aussi définir de nouveaux comportement d'affichage, par exemple la fermeture récursive de tous les sous-tiroirs. On illustrera les fonctionnalités de la librairie et sa flexibilité sur quelques petits exemples.

Notez que les composants d'affichage pourront ensuite être intégrés à la librairie graphique (on peut fournit ainsi une autre version du composant graphique "finder" avec une présentation arborescente)...

Implémentation d'un mini-langage à objets

Réservé à ceux ayant suivi la majeur langages de programmations.

Il s'agit :

- de définir le langage (sa sémantique et son système de typage)
- d'implémenter un typeur et un interprète pour ce langage.

Des indications notemment sur le typage seront fournis au besoin.

Le langage pour être un mini Ocaml (types structurels, polymorphies, inférés, ou un mini-java (types par nom, monomorphes, déclarés).

Pratiquement

Il vous sera demandé un bref rapport écrit de quelques pages (à préciser) ultérieurement à remettre avant la soutenance.

Les projets par binômes sont encouragés, mais les soutenances seront individuelles. Les projets couplés, *i.e.* deux projets complémentaires pouvant être ensuite fusionnés, sont aussi intéressants, notamment pour la librairie de calcul formel. À voir vers la mi-novembre.

Notes

Projet graphique

Les projets graphiques devront être effectués avec la librairie graphique de base.

En unix la librairie graphique a parfois un comportement anormal : après une attente sur les événements [Mouse_motion; Button_down], une attente sur l'événement [Button_up] répond à tord à un déplacement de la souris. Cela se contourne facilement en définissant :

```
let wait_button_up () =
  if Graphics.button_down() then
    while (Graphics.wait_next_event [ Button_up]).button do () done;;
let wait_event_button_up () =
  let e = ref (Graphics.wait_next_event [ Poll ]) in
  while !e.button do e := Graphics.wait_next_event [ Button_up ] done;
  !e;;
```

La première fonction ne retourne pas l'événement attendu, la seconde le retourne.

De façon général, il sera utile de compléter la librairie graphique par un module auxiliaire définissant par exemple d'autres couleurs de base, ou quelques fonctions supplémentaires, telles que la l'affichage, la sauvergarde et la restauration de rectangles, *etc.*

Les performances de la librairie graphiques sont suffisemment bonnes pour le projet. La seule précaution est de ne redessiner le contenu de la fenêtre que lorsque c'est nécessaire (en particulier, c'est inutile de la redessiner plusieurs fois avant de se mettre en attente sur un événement clavier ou souris) afin d'éviter son clignotement.

Projet de calcul formel

Pour ce projet, on pourra utiliser la libraire `num` pour permettre des calculs en précision arbitraire.

Soutenance

À la fin du cours d'EA vous devez passer une soutenance orale, que vous fassiez ou non le projet dans cet EA.

Il y a donc deux sortes de soutenances avec et sans projet qui se déroulement, *a priori*, de la façon suivante (mais l'examinateur se réserve le droit *de modifier entièrement le déroulement de l'oral.*)

EA seul

La soutenace d'un EA sans projet durera 1 heure, mais avec 30 minutes de préparation (élève seul) et 30 minutes de présentation orale.

L'examen comportera un exercice de programmation (à préparer en 30 minutes). La présentation oral consitera en la présentation de cet exercice (environ 15 minutes) suivie ou accompagnée de questions de l'examinateur (sur l'exercice de programmation ou sur le cours).

EA avec projet

La soutenance d'un EA avec projet durera 45 minutes, constitué entièrement d'une présentation, mais l'élève devra se présenter environ 15 minutes avant pour mettre en place sa démonstration. L'oral est individuel, même si vous êtes binomé pour votre projet (dans ce cas, vous devrez décrire la répartition du travail.)

La présentation du projet devra se faire en 25 minutes maximum, démonstration incluse. Le temps restant sera consacré à des questions sur le projet ou sur le cours.

De plus vous devez m'envoyer une description sommaire de votre projet (une centaine de lignes, en ASCII, par courrier électronique pour la mardi 19 au soir, au plus tard (n'envoyez pas le code).

Remarques

N'oublier pas que le sujet est sur la modularité. La présentation de votre projet doit donc montrer clairement la structure du programme, la modularité obtenue, à la fois dans l'écriture du logiciel et dans son utilisation en tant qu'expert ou en tant qu'utilisateur final.

Votre librairie est probablement écrire en plusieurs fichiers. Faites en une vrai librairie (voir Batch compilation). Faites aussi un `Makefile`, etc.

Une discussion des problèmes rencontrés dans l'organisation du code, la façon de les surmonter, ou au contraire les restrictions résultant de ces contraints sont aussi bienvenus.

L'utilisation de transparents est recommandée.