# Projet de programmation*
# Vers une amélioration de JasTEX

Stéphane Glondu

Jeudi 26 mai 2005

Ce document comporte 6 pages.

## 1 Introduction

JasTEX est une interface graphique — écrite en Java — pour GasTEX, un ensemble de macros LATEX — développées par Paul Gastin — pour dessiner notamment des graphes et des automates. La première version (à ma connaissance) a été développée[1] en 2002 dans le cadre d'un Travail d'Étude et de Recherche (TER) de maîtrise d'informatique à l'université Denis Diderot (Paris 7), par Michel Duong, Grégory Kokanosky et Yannick Ozouf.

Cette version souffrait de quelques défauts. Je me suis intéressé particulièrement aux suivants :

- la représentation interne des différentes longueurs intervenant dans les objets (position, taille d'un nœud, ...) était douteuse : parfois en pixels, parfois en unités naturelles, avec de nombreuses opérations de conversions, ce qui compliquait le zoom et la génération du code LATEX ;
- l'organisation du code rend difficile toute extension, malgré les commentaires : j'ai trouvé les classes trop emmêlées et des bouts de code pas à leur place (bien sûr, c'est une affaire de goût...).

Ces deux points m'ont obligé à récrire complètement un certain nombre de classes incompatibles avec l'ancienne architecture logicielle. En trois mois, je n'ai pas eu le temps de récrire une version qui a les mêmes fonctionnalités que l'ancienne, mais j'ai privilégié certains aspects, jugeant que les autres avaient un intérêt limité à ce stade. Ce projet pourrait aboutir dans le futur

*magistère STIC, second semestre, École Normale Supérieure de Cachan

[1]sous licence GPL

à un logiciel complètement différent. Ce que j'ai fait pour l'instant[2] englobe les fonctionnalités suivantes :

- le zoom ;
- l'édition à la souris d'un nœud standard et d'une arête simple ;
- l'édition de quelques propriétés dans une boîte de dialogue ;
- la génération du code LaTeX est partiellement implémentée, mais pas visible dans la démonstration.

Il reste encore beaucoup de choses à faire, les plus importantes à mes yeux sont :

- la gestion des étiquettes, utilisation de LaTeX pour générer un aperçu de celles-ci ;
- l'interfaçage avec le parseur n'est pas faite, cette version ne permet donc de travailler que sur un dessin statique ;
- la possibilité de modifier un dessin dans un fichier LaTeX sans modifier le reste du fichier.

# 2 Organisation des fichiers du projet

## 2.1 Archive tar

Le répertoire `src` contient les sources, le répertoire `rapport` contient les sources LaTeX de ce rapport, le répertoire `Jastex` contient le projet Netbeans (il est possible qu'il ne marche pas partout ; dans ce cas, créer tout simplement un projet à partir de sources existantes), et l'invocation du Makefile du répertoire `src` crée un répertoire `build` qui contient une version distribuable incluant l'ancienne version et la mienne. A priori, le répertoire `build` peut être archivé en jar. Il contient des exécutables `Jastex` et `devel` qui démarrent respectivement l'ancienne version, et une petite démonstration de ce que j'ai fait.

## 2.2 Sources

Initialement, toutes les classes étaient dans le package par défaut. J'ai choisi de regrouper les classes par paquets :

- `jastex` : classes générales, de lancement ;
- `jastex.gui` : interface graphique ;
- `jastex.obj` : objets (au sens des macros GasTeX) ;
- `jastex.obj.graph` : objets spécialisés dans les graphes ;
- `jastex.obj.tabs` : onglets d'éditions des objets ;

[2] ce rapport se réfère à la version 1.99.1 du 26 mai 2005

– `jastex.parsing` : parseur ;
– `jastex.test` : utilisé pour le développement ;
– `jastex.old.gui` : ancienne interface graphique ;
– `jastex.old.gui.tabs` : ancienne interface graphique spécialisée dans l'édition des propriétés des objets ;
– `jastex.old.obj` : objets avec l'ancienne représentation interne.

Il y a aussi, dans `src`, les ressources, comme le répertoire `images` utilisé par l'interface graphique, ou `lang` utilisé pour l'internationalisation. Ces fichiers sont recopiés tels quels dans la version distribuable. De plus, la distribution originale de JasTEX incluait les sources de JLex et de Java Cup, je n'y ai rien changé.

# 3 Solutions proposées

## 3.1 Interfaçage avec les objets GasTEX

Précédemment, ils héritaient tous de la classe abstraite `JasObject`, qui contient le code relatif à l'affichage et à la génération du code LATEX, mais aussi beaucoup de propriétés qui ne sont pas utilisées par tous les objets (comme la couleur de remplissage). En outre, certaines classes qui n'ont rien à voir avec les objets GasTEX descendaient aussi de `JasObject`. Chaque `JasObject` possède un champ `ggp` de type `GastexGlobalProperties` qui contient la configuration GasTEX en cours.

J'ai choisi de définir une interface `IJasObject` que doivent implémenter tous les objets, ainsi qu'une classe abstraite `AbstractJasObject` qui ne gère que le champ `ggp`. Pour pouvoir organiser (plus tard) un dessin en arborescence (avec partage de propriétés), j'ai aussi définit une classe `JasVector` qui est un vecteur de `IJasObject` et implémente lui-même `IJasObject`, et qui se contente de déléguer toutes ses actions à ses éléments. Ainsi, l'interface graphique n'a qu'à gérer l'affichage et la génération du code LATEX d'un seul objet.

Tous les « vrais » objets implémentés ici sont descendants de la classe `AbstractJasObject`. Comme `IJasObject` est une interface, cette relation de descendance n'est pas nécessaire (cela pourrait être une voie vers une génération automatique de code[3]). En utilisant des interfaces, on compartimente le code en rendant indépendantes l'interface graphique et la représentation interne des objets (modularité dont manquait cruellement l'ancienne version).

[3]On pourrait envisager ainsi de décrire dans un langage restant à concevoir les objets GasTEX, puis de générer automatiquement les packages LATEX et les classes Java adaptées pour s'en servir.

## 3.2 Objets GasTEX

Je n'ai implémenté[4] que les nœuds standards (`StdNode`) et les arêtes incurvées simples (`StdEdge`). Pour cela, je suis passé par plusieurs classes intermédiaires (cela devrait simplifier l'ajout de nouveaux objets) :

- interface `ILine` et classe `AbstractLine` : regroupent les propriétés communes aux lignes simples (et uniquement à celles-ci) ;
- interface `IClosedLine` et classe `AbstractClosedLine` : regroupent les propriétés communes aux lignes fermées ;
- interface `IEdge` et classe `AbstractEdge` : regroupent les propriétés communes aux arêtes ;
- classe `AbstractBezierEdge` : gère les arêtes qui ont la forme d'une courbe de Bézier cubique, se charge de leur affichage ;
- interface `INode` et classe `AbstractNode` : regroupent les propriétés commune aux nœuds ;
- classe `RMark` : gère les marqueurs de répétition (dédoublement de la bordure d'un nœud) ;
- classe `FMark` : gère les marqueurs initiaux ou finaux (flèche entrante ou sortante dans les automates).

Ces classes constituent les packages `jastex.obj` et `jastex.obj.graph` et sont organisées ainsi :

```
IJasObject
    JasVector
    AbstractJasObject
        AbstractLine (ILine)
            AbstractEdge (IEdge)
                AbstractBezierEdge
                    StdEdge
                FMark
            AbstractClosedLine (IClosedLine)
                AbstractNode (INode)
                    StdNode
            RMark
```

Le problème des longueurs évoqué précédemment est réglé : elles sont toutes stockées en unités naturelles (celles du code LaTeX), et la conversion n'est faite qu'à l'affichage, de manière transparente par l'objet `Graphics2D` de la bibliothèque Java. Le zoom se fait alors tout seul.

[4] j'ai repris beaucoup de code de l'ancienne version

### 3.3 Interface graphique

Les objets qui sont modifiables dans l'interface graphique doivent implémenter l'interface `IEditable`, qui définit des méthodes relatives à l'édition à la souris et par boîte de dialogue :

- `contains` : utilisée pour déterminer si le pointeur de souris survole un objet ;
- `drawSketch` : dessin spécial (pour l'édition) de l'objet ;
- `{get,update}MouseablePoints` : utilisées pour l'édition à la souris ;
- `updateToolBox` : utilisée pour l'édition par boîte de dialogue.

Ces méthodes sont toutes appelées par le composant `JasDrawingArea`. Ce composant contient d'une part un champ de type `JasVector` représentant le dessin global, et d'autre part un champ de type `IEditable` contenant l'objet en cours d'édition. La procédure pour changer ce champ est actuellement un vecteur contenant toutes les références vers les objets implémentant l'interface `IEditable`, qui est parcouru à chaque événement de souris.

L'édition à la souris se fait par l'intermédiaire d'objets descendant de `MouseablePoint`. Ils correspondent aux petits carrés que l'on déplace pour déplacer ou redimensionner des objets (`MiniBox` dans l'ancienne version). Chacun d'entre eux a notamment la méthode `onDrag` qui est systématiquement surchargée afin de modifier directement l'objet auquel ils se rapportent. Ce sont ces méthodes qui seront appelées par les gestionnaires de souris. La surcharge se fait directement dans le constructeur des objets éditables via des classes anonymes. La classe `MouseablePoint` prévoit aussi des champs pour l'image du curseur de souris et la granularité.

L'édition par boîte de dialogue se fait par l'intermédiaire de la classe `JasObjectToolBox`, qui est un conteneur de `JasTab`. Contrairement à la solution déjà implémentée, ici, c'est l'objet qui veut se faire éditer qui modifie la boîte de dialogue. Cette solution n'est pas encore tout à fait au point.

## 4 Démonstration

À partir de la racine de la distribution[5] compilée, exécuter[6] la classe `jastex.test.JFrameTest`. Cela démarre une fenêtre avec deux sommets, dont un avec un marqueur initial et de répétition, et une arête les reliant. Toute l'édition à la souris devrait fonctionner, et il est possible de modifier la couleur et l'épaisseur des lignes avec une boîte de dialogue. Il est aussi possible de changer le facteur de zoom.

[5] l'ancienne version est accessible via `jastex.Jastex`

[6] cela fonctionne au moins avec la version 1.5 de la JRE

# 5 Conclusion

Ce projet a été une bonne occasion de se familiariser avec Java. Il reste encore beaucoup de choses à faire, comme le suggère ce rapport, ainsi que les balises `TODO` et `FIXME` disséminées un peu partout dans le code, mais je pense avoir apporté des solutions claires à certaines problèmes, notamment l'extensibilité (ajouter d'autres types de nœuds, d'arêtes, ou d'objets quelconques, ne devrait plus poser de problèmes).