BROSSARD Florian / MILLOTTE Fanny

Projet tuteuré de 3ème année en licence d'informatique

Logiciel d'aide à la réalisation d'algorithmes et génération de code LaTeX

Tuteur de projet : M. BOUQUET Fabrice

UNIVERSITÉ DE FRANCHE-COMTÉ

UFR ST

16 route de Gray
25000 Besançon

Remerciements

Nous remercions Monsieur BOUQUET pour ses précisions sur le sujet, ses conseils, ses idées d'améliorations et pour sa disponibilité depuis le début du projet. Nous le remercions également pour la rapidité et la clarté avec lesquelles il à su répondre aux questions que nous nous sommes posés tout au long de ce projet.

Table des matières

1 - Présentation

Dans cette partie nous allons faire une présentation du sujet du projet.

1.1 - Contexte

Ce projet a été réalisé durant notre troisième année en licence d'informatique à l'UFR ST de Besançon. Il fait partie des épreuves comptabilisées pour le semestre six de notre cursus et est à réaliser en binôme.

1.2 - Présentation du sujet

Le but de ce projet est de réaliser un logiciel aidant à la réalisation d'un diaporama représentant un algorithme. Pour cela le logiciel doit générer du code LaTeX. Le choix du langage de programmation étant libre.

Nous avions une représentation du rendu visuel d'une diapositive (voir Illustration 1 - Exemple d'une diapositive). Nous pouvons ainsi observer le pseudo code en haut à gauche, le tableau des variables en haut à droite, de même que le graphe représentant l'avancement de l'algorithme en bas.

Le code LaTeX généré par le logiciel doit être basé sur beamer pour la gestion des animations ainsi que Tikz pour la gestion du graphe.

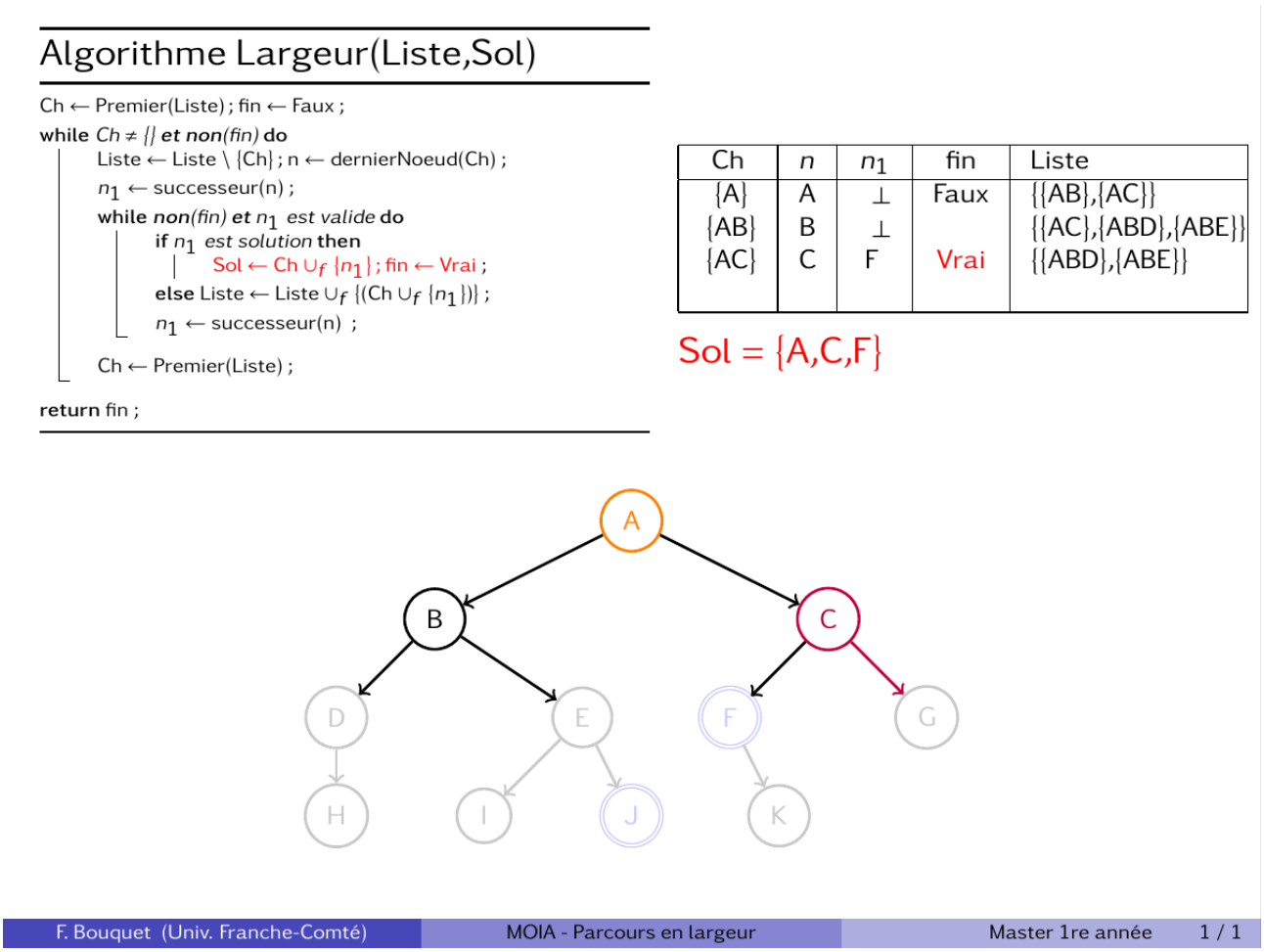

Illustration 1 - Exemple d'une diapositive

2 - Analyse des besoins

Une analyse des besoins est nécessaire à la compréhension du projet. Elle permet de définir ce qui est essentiel au projet et ce qui est optionnel. Ceci nous a aidé à définir dans quel ordre réaliser les différentes fonctionnalités de notre logiciel.

2.1 - Besoins de l'utilisateur

L'utilisateur a besoin de définir son algorithme par son pseudo code en montrant à quel déroulement de l'algorithme chaque diapositive correspond. Donc le logiciel doit disposer d'un éditeur pour le pseudo code ainsi qu'une mise en évidence sur ce dernier. Nous souhaitons faire ressembler cette partie à un éditeur de texte standard comprenant les numéros des lignes, chacun lié à une puce correspondant à la mise en évidence de la ligne.

Cette partie est visible en haut à gauche sur l'illustration 1. Une modification de ce pseudo code doit entraîner une mise à jour de celui ci sur l'ensemble des diapositives.

L'utilisateur a besoin de montrer clairement les valeurs contenues dans les variables définies dans le pseudo code. Il doit également pouvoir montrer à quel moment une variable change de valeur. Donc le logiciel doit contenir un tableau représentant les variables du pseudo code ainsi qu'une mise en évidence des variables au moment d'un changement de valeur. Le changement d'une valeur par rapport à une diapositive précédente doit entraîner la mise en évidence de cette valeur. L'utilisateur doit aussi pouvoir modifier le nombre de ligne et de colonne du tableau par une option permettant l'ajout ou la suppression de ces dernières.

L'utilisateur a besoin d'éditer un graphique représentant le déroulement de l'algorithme pas à pas ainsi qu'une mise en évidence du chemin parcouru à un moment précis. Le logiciel doit donc comporter une partie permettant l'édition d'un graphe ainsi qu'une mise en évidence d'une partie précise du graphe représentant le chemin qui est parcouru. Il est donc nécessaire d'ajouter des options permettant l'ajout d'un nœud et d'un lien ainsi que la suppression de ces deux éléments et la modification de la couleur d'un élément. Il est également utile de permettre à l'utilisateur de changer l'état d'un nœud, un nœud final étant représenté par un double cercle. Chaque nom d'un nœud ou d'un lien doit être modifiable. La seule chose qui n'est pas en commun sur chaque diapositive est la couleur des éléments.

L'utilisateur à également besoin de visualiser le rendu diapositive par diapositive. Il est donc impératif de pouvoir effectuer un changement de diapositive afin de visualiser l'avancement ou le rendu final d'un diaporama. Pour cela il doit être possible de passer à la diapositive suivante ou précédente, de donner le numéro de la diapositive souhaité ou encore de choisir de quelle partie changer la diapositive indépendamment des autres parties.

Pour finir le logiciel doit retranscrire les informations saisies par l'utilisateur sous la forme d'un code LaTeX. Il est donc utile d'ajouter une option permettant de choisir le fichier dans lequel on souhaite le générer.

En résumé, voilà l'ensemble des fonctionnalités nécessaire à l'utilisateur :

Editeur de pseudo code	Mise à jour des informations en cas de modification
	Mise en évidence
Tableau des variables	Mise à jour des noms des colonnes
	Ajout / suppression d'une ligne
	Ajout / suppression d'une colonne
	Mise en évidence lors d'un changement de variable
Graphique	Ajout d'un nœud
	Ajout d'un lien
	Modification de l'état d'un nœud
	Edition du label d'un nœud
	Edition du label d'un lien
	Mise à jour du graphe en cas de changement sur une diapositive
Changement de diapositive	Ajout / suppression d'une diapositive
	Changer de diapositive
	Sélectionner à quelle diapositive aller
	Sélectionner une ou plusieurs partie concerné par le changement de diapositive
Génération du code LaTeX	Générer un code LaTeX lié aux informations saisies
	Choisir le fichier dans lequel on souhaite générer le code

2.2 - Aspect technologique

Nous avions deux choix pour effectuer ce projet, dans un premier temps reprendre le travail réalisé durant l’année 2015 – 2016 par un autre binôme ou recommencer le projet du début.

2.2.1 - Première solution

Nous nous sommes d'abord tournés vers le premier choix, reprendre le travail laissé par un autre binôme. Le binôme précédent a choisi de réaliser le projet en C++ et Qt. Nous avons rencontrés de nombreuses difficultés tout d'abord avec la compréhension du code, nous n'avions fait que très peu de C++ et nous n'avions jamais utilisés Qt. L'absence de nombreux commentaires, de documentation et la non-organisation des fichiers ont grandement entravées notre travail de compréhension du code.

Le projet de ce binôme était presque entièrement terminé il ne manquais que les tests unitaires.

Le fait d'avoir essayé de reprendre ce projet nous à appris beaucoup de chose sur le code à laisser à d'éventuels personnes. Tout d'abord l'importance d'avoir un code clair et de ne pas oublier de commenter ce code tout au long du développement ainsi que l'importance de la documentation à apporter à ce code.

N'étant pas suffisamment à l'aise avec le langage C++ et n'ayant pas beaucoup avancé dans la compréhension du code pendant une période d'un mois nous nous sommes tourné vers la deuxième solution.

2.2.2 - Seconde solution

Nous avons finalement opté pour la seconde solution : recommencer le projet du début tout en intégrant ce que nous avons appris en essayant la première solution. Nous souhaitons éviter que notre programme soit trop difficile à reprendre si un éventuel binôme le fait.

Nous voulons choisir un langage que nous maîtrisons mieux, le C++ étant un langage que nous avons peu expérimenté.

Pour la réalisation de ce projet nous avons choisi d'utiliser JAVA. C'est le langage objet que nous avons le plus utilisé et nous avions également une matière lors de notre 5ème semestre nous apprenant comment utiliser la bibliothèque graphique swing de java. Cette matière à pu nous aider pour la réalisation de ce projet de même que le projet nous a aidé pour l'élaboration du projet de cette matière.

L'implémentation de tests unitaire nous permettra de déceler d'éventuels erreur durant le développement de notre logiciel et nous aidera à améliorer la structure de ce dernier. Pour cela nous avons décidé d'utiliser Junit, un framework open source pour le développement et l'exécution de tests unitaires automatisables.

Afin de suivre pas à pas le développement de notre logiciel nous avons choisi d’utiliser le logiciel de gestion des versions Git que nous n’avions jamais utilisé auparavant afin de pouvoir suivre l’évolution du projet pas à pas.

3 - Conception

Dans cette partie nous allons traiter l'aspect conception de notre logiciel.

3.1 - Structure du logiciel

Pour ce projet nous avons choisi d’utiliser l’architecture Modèle-Vue-Contrôleur pour l’implémentation des classes. Les diagrammes de classe correspondant aux différentes parties du projet sont présent en annexe.

3.2 - Conception des différents éléments

Nous pouvons voir dans le diagramme représentant le package Vue que la fenêtre principale est composé de plusieurs choses :

- Une barre de menu

- Une barre d'action

- Un panel comprenant le générateur de pseudo code où l'on peut retrouver un panel composé des numéros des lignes du pseudo code et des puces de mise en évidence

- Un panel comprenant le tableau des variables

- Un panel comprenant le graphe

- Une boite de dialogue qui permet de saisir les informations de la présentation

Nous pouvons voir dans le diagramme représentant le package Modele le lien qui existe entre les nœuds du graphe et les liens. Nous pouvons également voir les différents composant constituant la boite de dialogue correspondant aux informations de la présentation.

Nous pouvons voir dans le diagramme représentant le package Contrôleur les différents contrôleurs des composants inclus dans le logiciel, ainsi que les liens entre les différents contrôleurs du graphe.

Nous pouvons voir dans le diagramme représentant le package Utilities toutes les classes utilitaires utilisées pour la réalisation du projet : la génération du code LaTeX, le dessin d'un cercle, la gestion de la sauvegarde, etc.

3.3 - Logiciel -> LaTeX

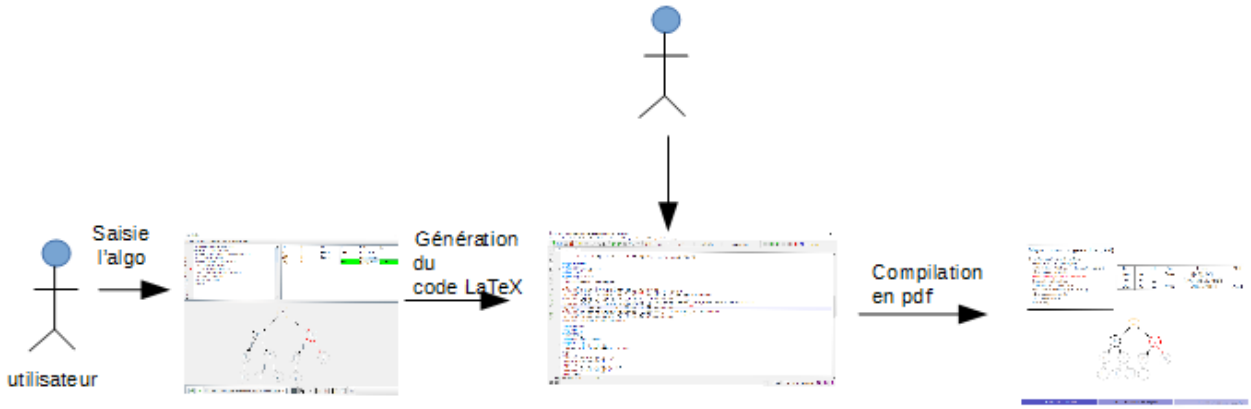

Illustration 2 - Principe du logiciel

L'utilisateur va saisir son algorithme dans le logiciel puis demander la génération du code LaTeX. Ceci va ouvrir le logiciel définis par le système pour ouvrir les fichiers .tex. Ensuite l'utilisateur devra utiliser ce logiciel pour compiler le fichier .tex en pdf, ou tout autre format souhaité.

3.4 - Synchronisation entre les diapositives

La synchronisation entre les diapositives est un mécanisme d'aide pour l'utilisateur.

Chaque panel à le même nombre de diapositive, l'ajout d'une diapositive sur un panel entraîne l'ajout d'une diapositive sur l'ensemble des panels aux positions affichées à l’écran. Certaines données sont communes à toutes les diapositives et sont immédiatement mise à jour en cas d'ajout d'une diapositive ou de modifications sur l'une d'entre elles. Ces données sont : le texte du pseudo code, la taille et le nom des colonnes du tableau des variables, le dessin du graphique ainsi que les labels des nœuds et des liens.

Lors de l'ajout d'une diapositive la puce présente sur l'éditeur de pseudo code passe à la ligne suivante par rapport à la diapositive précédente. De même que les données présentes dans le tableau des variables sont copiés sur celles présente dans la diapositive précédente. Les états des éléments du graphe sont tous définis Inactif par défaut puis chaque élément voyant sont état changé sur une diapositive voit ses états suivants changés à Parcourus s'ils sont définis comme Inacitf.

4 - Réalisation

Dans cette partie nous allons aborder l'aspect réalisation du logiciel. Nous passerons en revus les étapes du développement, les tests effectués puis nous verrons les plus grands problèmes rencontrés ainsi que les différentes améliorations effectuées

4.1 - Etapes du développement

Nous allons a présent parler de chaque étapes du développement du logiciel.

4.1.1 - Aspect visuel

Nous avons débutés notre projet en structurant notre fenêtre pour qu'elle ai un aspect le plus proche du rendu final attendu. Pour cela nous avons placé en haut à gauche un panel correspondant à l'éditeur de pseudo code, en haut à droite un panel correspondant au tableau des variables et en bas un panel correspondant au graphe.

4.1.2 - Editeur de pseudo code

Par la suite nous avons implémentés notre éditeur de pseudo code. Dans un premier temps nous avons ajouté une simple zone de texte et nous nous sommes penchés sur la problématique suivante : Comment visualiser la mise en évidence d'une ou de plusieurs lignes du texte ?

Nous avons d'abord pensés à ajouter des checkboxs au début de chaque ligne du texte, correspondant aux lignes devant être mises en évidence.

Mais nous avons finalement décidé de nous rapprocher le plus des outils informatique utilisés dans la vie courante. Nous avons donc ajoutés une colonne sur la gauche du texte où sont visible les numéros des lignes et où par un simple clic sur ce numéro, une puce indiquera que la ligne correspondante sera mise en évidence pour la diapositive courante. Un clic sur un numéro d'une ligne déjà mise en évidence entraîne la suppression de cette mise en évidence.

4.1.3 - Tableau des variables

Nous sommes ensuite passés à la réalisation de la partie comprenant le tableau des variables. Nous avons tout d'abord créés un tableau où il est possible de supprimer et d'ajouter des lignes et des colonnes.

Pour la mise en évidence nous avons également pensés à mettre des checkboxs correspondants aux lignes et colonnes du tableau.

Mais nous souhaitions faciliter le travail de l'utilisateur, nous avons donc ajoutés une fonctionnalité permettant de mettre en évidence une case modifiées durant la diapositive courante par rapport à la précédente. Le nom des colonnes n'étant pas pris en compte dans la mise en évidence.

4.1.4 - Graphe

Le graphe est constitué de nœuds et de liens entre deux nœuds. Chaque nœud est constitué du dessin d'un cercle simple ou double dans lequel se trouve un label (son nom). Si l'on souhaite créer un nœud depuis le menu le nœud, il sera positionné en (0,0). Si l'on utilise le menu contextuel sur la partie graphique, le nœud sera positionné à l'emplacement du clic.

Le menu contextuel présent sur chaque élément permet de changer l'état de cet élément ainsi que de le supprimer. Si l'élément est un nœud, le menu contextuel propose la suppression de ce nœud, entraînant la suppression des liens associés à ce nœud, ainsi que la création d'un lien entre ce nœud et un autre et pour finir le changement du type de ce nœud.

La couleur d'un élément est définie par son état :

```
public enum EtatParcours {
        Inactif, Actif, Parcourus, Solution, NonSolution, Erreur
    }
```

A noté que l'utilisateur ne peut pas choisir l'état ***Erreur*** **,** cet état étant utilisé par le logiciel pour signaler une erreur de traitement, en revanche sa couleur peut être définie.

La forme d'un nœud est définis par son type :

```
public enum TypeForme {
        Simple, Double, Initial
    }
```

Ainsi un nœud est par défaut simple, dessiné par un seul cercle, tandis qu'un nœud final correspond au type Double, dessiné par deux cercle de rayons différents. Un nœud de type Initial est dessiné comme un nœud simple sauf quand son état est Inactif ou Parcourus, auquel cas sa couleurs est spécialement définie.

4.1.5 - Changement de diapositive

Nous avons par la suite choisi de tester nos deux premières parties en réalisant les fonctions traitant le changement de diapositives. Pour cela nous avons liés le texte du pseudo code à toutes les diapositives, une modification apportée sur une diapositive est donc répercuté sur toutes les diapositives suivantes, de même pour les noms des colonnes du tableau mais sur toutes les diapositives. Nous considérons que chaque diapositive représente un pas dans l'algorithme, pour cela nous avons donc choisi de faire passer à la ligne suivante la puce du pseudo code lors de la création d'une nouvelle diapositive ou d'en créer une à la première ligne s'il n'y en a pas.

4.1.6 - Sauvegarde des données

Nous avons effectués la sauvegarde des données sous forme d'un fichier XML, qui permet de structurer les données dans le même format que les objets java, chaque informations stockées et utilisées dans la session en court sont lues et ajoutées à une chaîne de caractère qui seras écrite dans un fichier choisis par l'utilisateur.

Tout d'abord les informations relatives à la présentation : auteur, nom de l'algorithme, couleurs utilisées, taille des différentes minipages utilisées pour le code LaTeX, etc.

Puis les informations du panel comportant l'éditeur de pseudo code : texte, marqueur par diapositive. Par la suite, les informations contenu dans le tableau de variables.

Pour finir les informations du graphe. Tout d'abord les nœuds avec leurs positions chacun étant pourvu d'un identifiant unique. Dans chaque nœud on enregistre son nom ainsi que l'état dans lequel ils se trouvent pour chaque diapositive. Les liens sont enregistrés de la même manière mais sont liés aux identifiants correspondant aux nœuds auxquels ils sont rattachés.

4.1.7 - Génération du code LaTeX

Chaque information entrée dans le logiciel permet, lors de la génération du code LaTeX, de remplir un patron de code LaTeX déjà existant.

Nous avons choisi de réaliser ce code avec la notion d'une diapositive comprenant plusieurs pas. Cette méthode est plus simple à manipuler en cas de modification du code par rapport à la deuxième possibilité, qui consiste à créer plusieurs diapositive comprenant chacune un seul pas. Ainsi chaque modification sera appliquée à toutes les diapositives présentes sur le fichiers compilé.

4.2 - Test effectués

Nous avons implanté 8 catégories de test avec chacun un panel de jeu de données différents.

Quatre fonction de test on été créés pour la partie pseudo code correspondant à 14 tests :

- Test sur la récupération des lignes du pseudo code
- Test sur récupération du nombre de ligne du pseudo code
- Test sur la gestion des diapositives du pseudo code
- Test sur la gestion du marquage des lignes du pseudo code

Deux fonction de test on été créés pour la partie tableau correspondant à 12 tests :

- Test de la détection lors d'un changement de variable
- Test de la répercussion des modifications apportées au valeurs des variables

Deux fonction de test on été créés pour la partie graphe correspondant à 14 tests :

- Test sur la création d'un lien et la suppression d'un nœud associé
- Test sur la récupération des nœuds et des informations associées

4.3 - Problèmes rencontrés

Lors de la réalisation du projet nous avons rencontrés deux grands problèmes.

La sélection des liens du graphe étant un ovale dessiné à l'horizontale (car on ne peut pas lui appliquer directement un angle), la sélection des liens était uniquement effectuées sur le coté du premier nœud correspondant au lien. Ce problème à été résolue à l'aide d'équations impliquant la translation et la rotation du point du clic de souris vers l'ovale de détection afin qu'ils coïncides.

Lors de la génération du code LaTeX nous avons rencontrés plusieurs problème par exemple pour la disposition des éléments. Le tableau des variables se trouvait en dessous du pseudo code et non à côté, ce problème à été résolus : il s'agissait d'un simple saut de ligne en trop entre la partie pseudo code et la partie tableau des variables du code LaTeX.

4.4 - Améliorations effectués

Nous avons effectué quelques petites amélioration par rapport aux besoins initiaux de l'utilisateur.

Il est à présent possible de changer les couleurs par défaut des états des nœuds et des liens du graphe.

Il est également possible de changer la forme des liens du graphe par des liens simple ou flêchés.

5 - Bilan

Nous allons a présent effectuer un bilan sur les différents aspect de notre projet.

5.1 - Bilan pédagogique

Ce projet nous à permis d'utiliser les compétences que nous avons acquis durant notre cursus universitaire et de parfaire ces connaissances. Cela nous à notamment aidés lors des projets et examens des différentes matières liés à java swing pour l'interface graphique, et à XML pour la sauvegarde.

Nous avons également pu expérimenter l'aspect programmeur / client : comment répondre aux besoins d'un client, déterminer ce qui est essentiel à ce client pour la création du projet. Cela nous sera très utile pour notre futur stage en entreprise.

5.2 - Bilan technique

Les fonctionnalités que nous avons déterminés comme essentielles ont été implémentées.

Notre programme permet l'édition d'un pseudo code, d'un tableau correspondant aux variables ainsi qu'un graphe. La mise en évidence des données présentes dans ces trois parties est fonctionnelle. Il est possible de changer l'état de chaque nœud du graphe. Le logiciel peut créer et lire des sauvegardes en XML et le généré est compilable.

5.3 - Bilan général

Toute les fonctionnalités nécessaire à l'utilisateur du logiciel on été réalisées et sont fonctionnelles. Les difficultés rencontrées ont été résolues.

Ce projet nous a appris comment réaliser un projet demandé par un client, cela nous sera d'une grande aide pour nos futur projets à réaliser en entreprise.

5.4 - Améliorations possibles

Beaucoup d'améliorations sont possibles pour ce projet notamment implanter un arbre de syntaxe abstrait pour le traitement du pseudo code afin de pouvoir générer des blocs de code LaTeX tel que /While{condition}{instructions} en extrayant des mot clés dans le pseudo code, améliorer l'aspect visuel et pourquoi pas faire en sorte que ce logiciel puisse servir à tout ceux souhaitent générer du code LaTeX, par exemple pour réaliser un support de cours ou une présentation orale. De plus d'autres amélioration, plus techniques, sont possibles comme la réduction de la taille des fichiers de sauvegardes, pouvant figés quelques secondes le logiciel en cas de projet plus ou moins importants.

7 - Annexes

A - Diagrammes UML

Diagramme de la partie vue

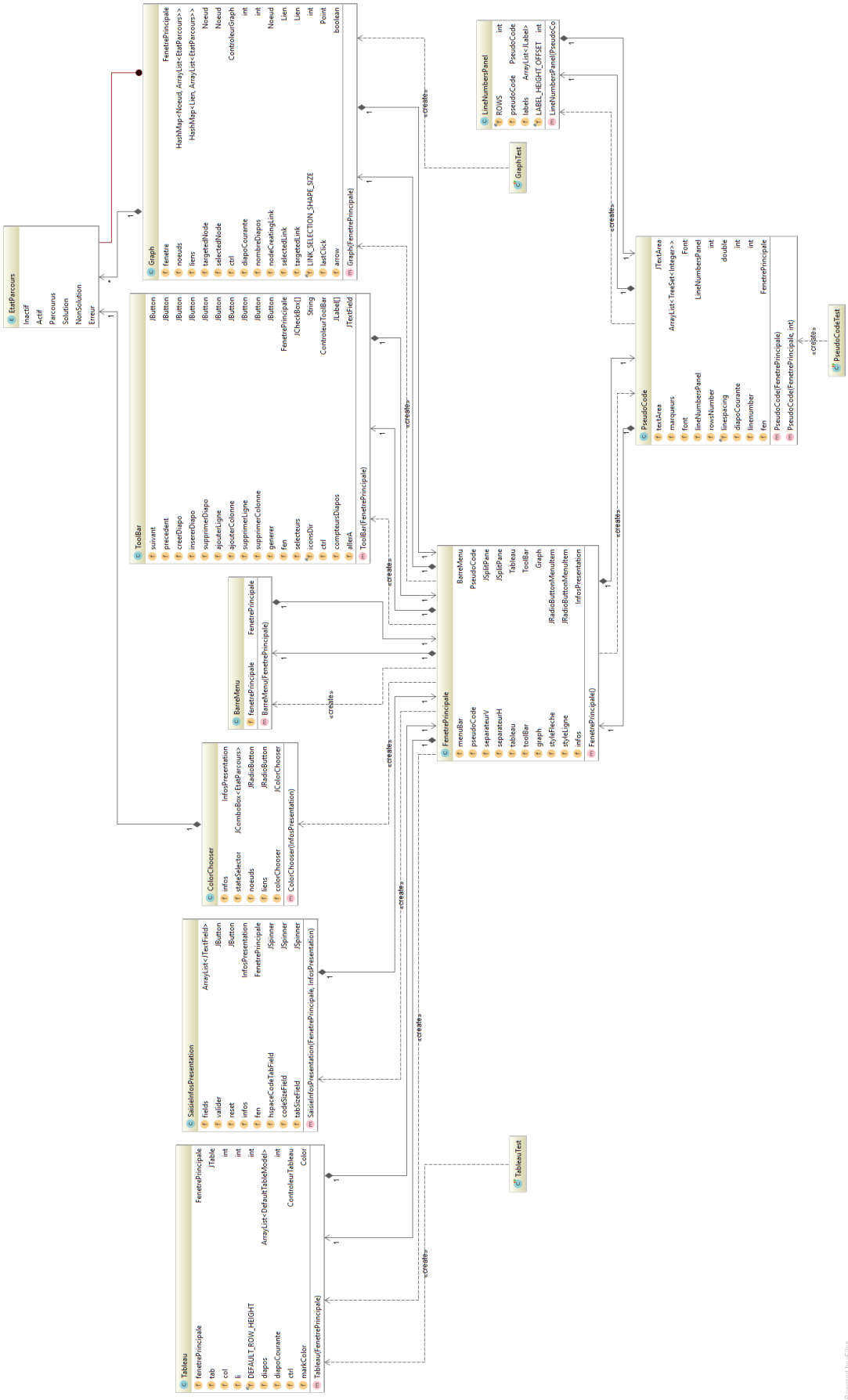

Diagramme de la partie modèle

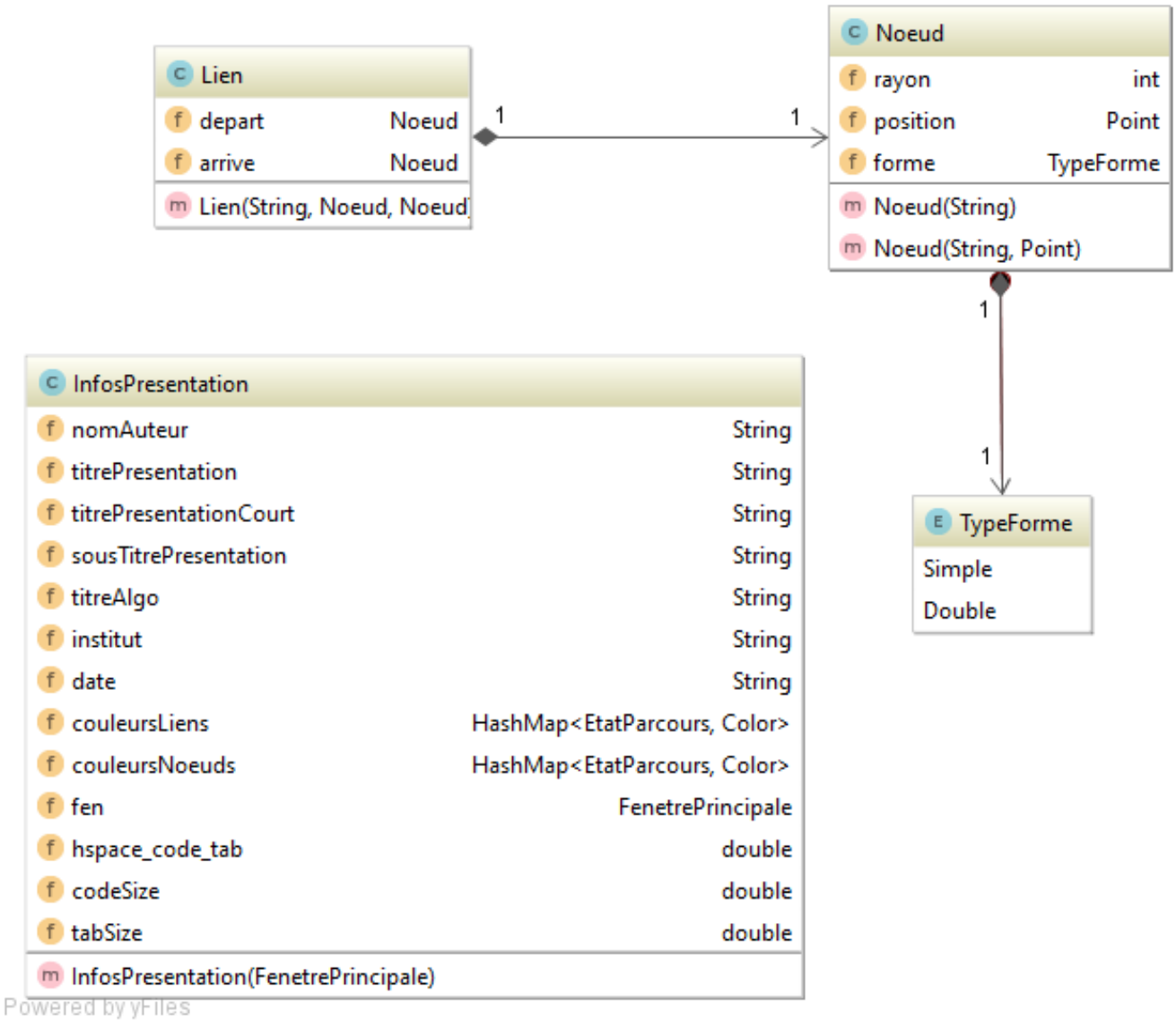

Diagramme de la partie contrôleurs

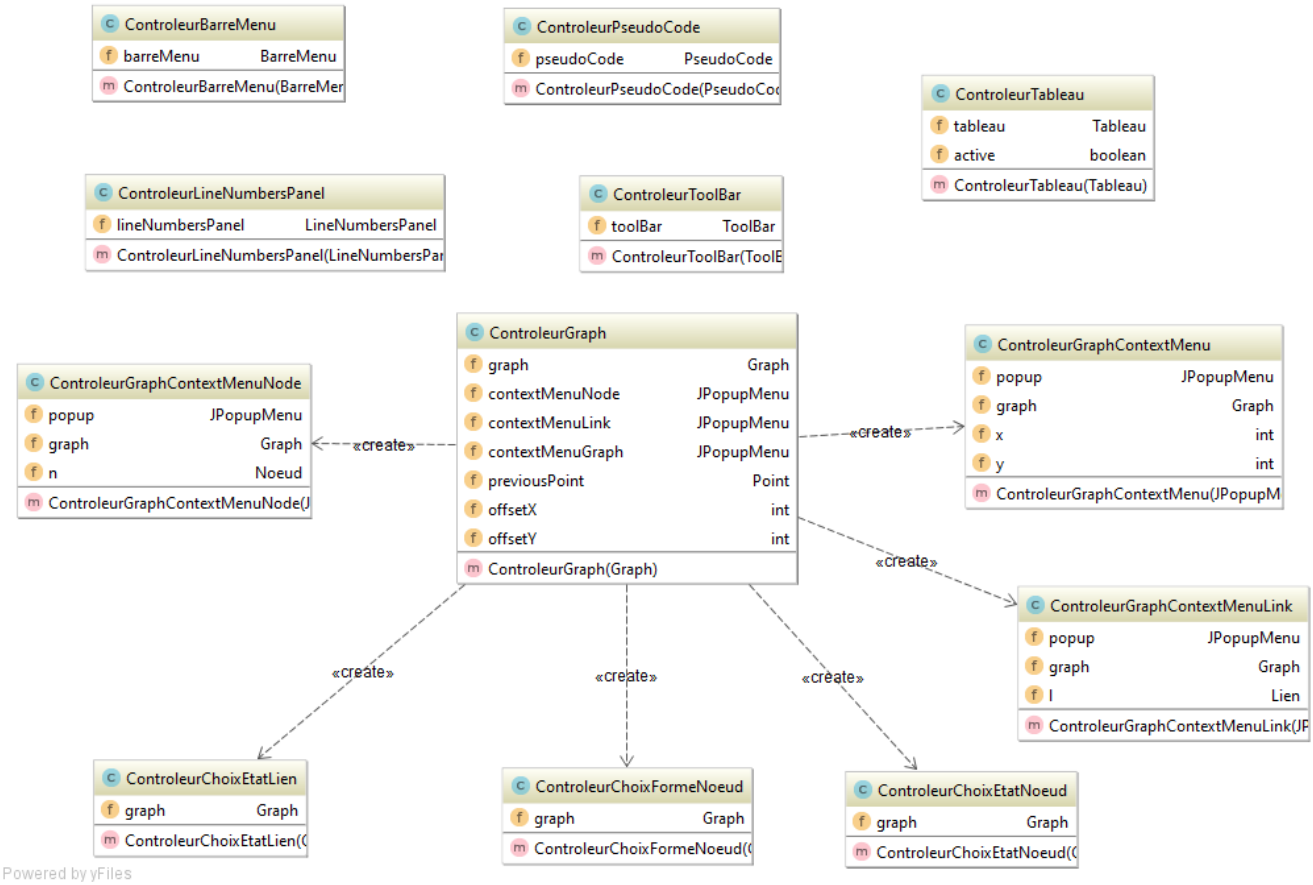

Diagramme des utilitaires