



Rapport de projet de Génie Logiciel 2^{ème} phase
Master 2 Pro. Ingénierie de la Communication Personne-Systèmes

ANIMATHS

UN ENVIRONNEMENT WEB INTERACTIF POUR LA MANIPULATION DIRECTE ET L'ANIMATION DE TRANSFORMATIONS D'OBJETS MATHÉMATIQUES

Maxime Lefrançois & Édouard Lopez

UPMF IC2A – Master 2 Pro. Ingénierie de la Communication Personne-Système

Date de soutenance : 26 février 2010

Projet encadré par Mohamed El Methni

Enseignant Chercheur – Chargé de mission TICE de l'Université Pierre-Mendès-France.

TABLE DES MATIERES

Contexte et Rappel.....	4
Introduction	4
Cadre du projet.....	4
La représentation arborescente des objets mathématiques	5
Notations	5
Diagramme syntaxique (ou <i>Railroad</i>).....	6
MOEquation (MOE)	6
MONumber (MON)	7
MOIdentifieur (MOI)	7
MOSignedElement (MOSE)	7
MOAddContainer (MOAC).....	8
MOMultiplyElement (MOME).....	8
MOMultiplyContainer (MOMC).....	9
L'interface utilisateur de l'application	10
Description	10
Fil d'Ariane	10
Zone d'action(s) rapide(s)	10
Consigne de l'exercice	11
Zone passive de manipulations	11
Zone active de manipulations.....	11
Manipulation d'équation.....	11
Signalétique des objets MathObjects	11
Etat primaires pour les MathObjects de la zone de manipulation	11
États secondaires pour le MathObject copié attaché au curseur	13
Description des zones d'interaction.....	14
Les boîtes de dialogue : choix ou nécessité ?	14
Manipulations implémentées	17
Addition	17
Simplification de signe	17
Multiplication.....	18

Équation.....	18
Phase de développement.....	20
Résumé des choix retenus lors de la première phase.....	20
Architecture globale du projet	20
Le package client.mvp	21
Implémentation des éléments MathML	22
Les objets MathObject.....	23
POO et critique du MVP	23
exigences sur l'implémentation des objets MathObject	23
Le package client.interaction	24
Le package client.event	24
Description d'un processus	25
Stockage des données sur le serveur.....	27
Communication client-serveur (RPC).....	27
Exercices & Tutoriaux	29
Structure des fichiers.....	29
Analyse	30
Tutoriel.....	31
Bilans.....	31
Retour sur les objectifs.....	31
Problèmes rencontrés	32
MathML, Firefox, GWT et le HTML.....	32
GWT	32
Eclipse	32
UiBinder.....	33
Bilan personnel.....	34
Maxime.....	34
Édouard	34
Références.....	35

Introduction

L'idée du projet AniMaths est de proposer une interface web où un utilisateur peut manipuler directement les objets mathématiques. L'apprenant se retrouverait alors face à un genre de casse-tête (puzzle mathématique) qu'il devrait manipuler pour trouver la solution au problème. L'apprenant pourrait alors comprendre par l'exemple quelles manipulations les mathématiques l'autorisent à effectuer sur une expression ou une équation.

Dans ce projet, on propose une représentation arborescente des manipulations sur un objet mathématique, où l'ensemble des nœuds représente l'ensemble des expressions égales (ou des équations équivalentes), et où chaque arc représente l'utilisation d'un axiome, ou d'une opération élémentaire.

Dans le rapport de la première phase de ce projet, et lors de sa soutenance le 11 janvier 2010, nous avons présenté en détails les motivations de ce projet, les objectifs que nous nous proposons d'atteindre ainsi que la philosophie adoptée. Nous avons également présenté le détail des exigences sur le programme AniMaths, la recherche de solutions techniques que nous avons menée, les premiers choix de conception ainsi que le début de la phase de développement.

Nous présentons dans ce rapport le travail d'un mois et demi. Nous sommes très heureux des avancées de notre application dans ce laps de temps relativement court, et nous pouvons dès à présent affirmer que nous serions très enthousiastes à l'idée qu'il soit poursuivi. Nous pensons également qu'une semaine supplémentaire ou deux suffiraient pour aboutir à une version utilisable en situation réelle.

Cadre du projet

Ce projet de Génie Logiciel prend place dans le cadre du Master 2 Ingénierie de la Cognition, de la Création, et des Apprentissages (IC2A) spécialité Ingénierie de la Communication Personne-Système (ICPS) de l'Université Pierre-Mendès France (UPMF) de Grenoble.

Il est encadré par Mohamed El Methni et prend place dans le service Technologies de l'Information et de la Communication adaptées à l'Education (TICE) et de l'Enseignement à Distance (EAD) de l'UPMF.¹

Ce service est chargé de diffuser la politique TICE de l'établissement au sein des composantes et de coordonner l'enseignement à distance à l'UPMF. Il pilote les projets interuniversitaires TICE du site (GU), de la Région Rhône-Alpes et nationaux (FIED, SDTICE MESR).

¹ Organisation TICE-EAD : http://tice.upmf-grenoble.fr/75800044/0/fiche___pagelibre/&RH=TICEFR_1&RF=TICEFR1_1

LA REPRESENTATION ARBORESCENTE DES OBJETS MATHÉMATIQUES

Lors de la première phase, nous avons choisi le *MathML* pour l'affichage des équation mathématiques sur le navigateur. Le *MathML* est bien adapté pour la visualisation, mais est vide de sens mathématique. Pour faciliter les manipulations sur les objets mathématiques, nous avons proposé une vision arborescente univoque des objets mathématiques : Les *MathObjects*. Cette partie décrit avec précision les arborescences *MathObjects* possibles.

Notations

Dans ce rapport, nous désignerons les objets mathématiques *MathObject* avec un préfixe *MathObject*, MO, indifféremment. Dans le programme, les classes en rapport avec *MathObjects* sont préfixées MO.

Les *MathObject* sont suffixés par le type d'éléments qu'ils représentent (p.ex. MOEquation, MOIdentifier, etc.). Nous utiliserons également une notation abrégée : MO, suffixée par les initiales de l'élément représenté (p.ex. : MOE, MOI, etc.).

Designation dans le programme	Code de l'élément MathObject
MOEquation	MOE
MONumber	MON
MOIdentifier	MOI
MOSignedElement	MOSE
MOAddContainer	MOAC
MOMultiplyElement	MOME
MOMultiplyContainer	MOMC

Tableau 1 : notations des objets mathématiques *MathObjects*.

Pour les objets mathématiques *MathML*, nous les préfixerons par MML ou ML pour la forme abrégée. Ils seront également suffixés par l'élément qu'il représente, ou pour faire plus court, par des initiales.

Designation dans le programme	Code de l'élément
MMLMath	MLMath
MMLIdentifier	MLI
MMLNumber	MLN
MMLOperator.equality()	MLEgal
MMLOperator("+")	MLPlus
MMLOperator("-")	MLMoins
MMLOperator.times()	MLMult
MMLFrac	MLFrac
MMLRow	MLRow
MMLOperator.lFence()	MLPar
MMLOperator.rFence()	

Tableau 2 : notations des objets mathématiques *MathML*

Diagramme syntaxique (ou *Railroad*)²

Pour décrire l'imbrication des différents objets représentant une équation, nous utiliserons des diagrammes similaires aux diagrammes syntaxiques ou *railroad*. Les règles de lecture sont les suivantes :

1. Commencer par la gauche et suivre le chemin vers la droite.
2. Lors du parcours, on rencontre :
 - des littéraux représentés par des ovales,
 - des règles ou descriptions représentées par des rectangles arrondi.
3. Tout chemin qui peut être effectué en suivant les arcs orientés est valide.
4. Tout chemin qui ne peut pas être effectué en suivant les arcs orientés est invalide.

The rules for interpreting these diagrams are simple:

You start on the left edge and follow the tracks to the right edge.

As you go, you will encounter literals in ovals, and rules or descriptions in [rounded] rectangles.

Any sequence that can be made by following the tracks is legal.

Any sequence that cannot be made by following the tracks is not legal.

Railroad diagrams with one bar at each end allow whitespace to be inserted between any pair of tokens. Railroad diagrams with two bars at each end do not.

JavaScript: The Good Parts by Douglas Crockford pp. 23

MOEquation (MOE)

C'est l'objet racine d'une équation ou d'une inéquation. Pour le moment, nous nous sommes limités aux égalités. De ce fait, on décrit une équation comme étant composée de deux membres de type générique MOElement séparés par un signe égal de type MMLOperator.equality(). Nous verrons qu'il sera possible d'étendre le programme très simplement avec des classes pour décrire les interactions spécifiques aux inégalités.

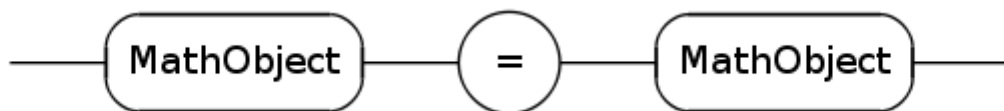


Figure 1 : Objet MOEquation

² Diagramme syntaxique : http://en.wikipedia.org/wiki/Syntax_diagram

MONumber (MON)

Élément de type terminal représentant une valeur numérique positive. Cet objet est une feuille dans l'arbre d'une équation, il correspond à l'élément MathML $\langle mn \rangle^3$.

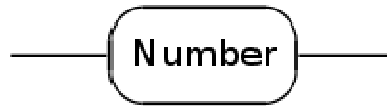


Figure 2 : Objet MONumber

MOIdentifier (MOI)

Élément de type terminal représentant une variable, par exemple x . Cet objet est une feuille dans l'arbre d'une équation, il correspond à l'élément MathML $\langle mi \rangle^4$.



Figure 3 : Objet MOIdentifier

MOSignedElement (MOSE)

Cet objet représente un pseudo-conteneur qui permet d'indiquer le signe de l'élément fils, grâce à la propriété *isMinus*. Par défaut, l'enfant est considéré comme étant positif, i.e. *isMinus* vaut *false*. L'élément fils de cet objet peut être de nature diverse, toujours dérivé du type MOElement.

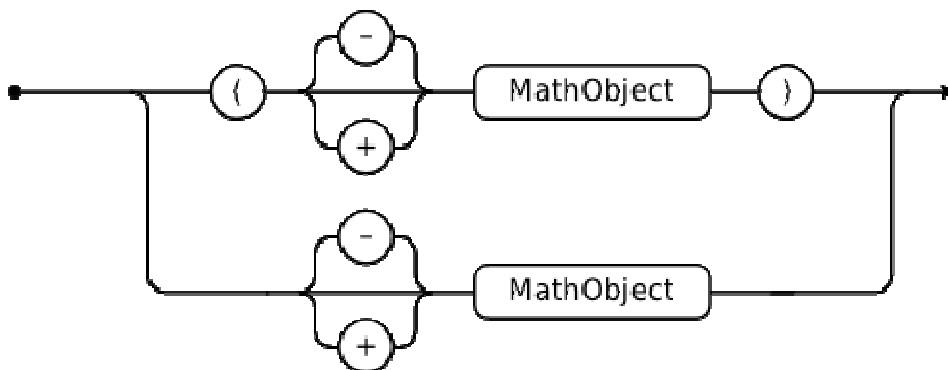


Figure 4 : Objet MOSignedElement

Les règles de priorité mathématiques impliquent que cet objet peut avoir des parenthèses. La présence de celle-ci est déterminée en fonction du parent ; un MOSignedElement aura des parenthèses si son parent est de l'un des types suivants :

- MOSignedElement
- MOMultiplyElement
- MOMultiplyContainer

³ MN : <http://www.w3.org/TR/MathML2/chapter3.html#presm.mn>

⁴ MI : <http://www.w3.org/TR/MathML2/chapter3.html#presm.mi>

Le signe d'un `MOSignedElement` peut-être affiché, ou non. Par défaut, il est affiché. Le seul cas où l'on rencontrera où le signe ne sera pas affiché est quand l'objet est le premier objet d'une addition de plusieurs `MOSignedElement`, et que son signe est positif.

MOAddContainer (MOAC)

Cet objet est un conteneur pour les additions et les soustractions. Il ne contient que des enfants signés de type `MOSignedElement`.

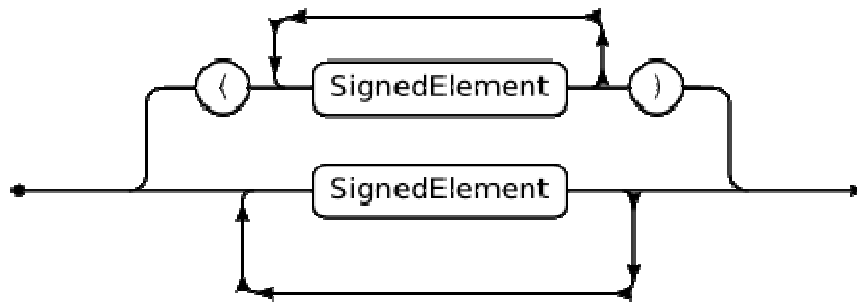


Figure 5 : Objet MOAddContainer

Ici encore, par sa nature de conteneur, cet élément peut avoir des parenthèses si son parent est de l'un des types suivants :

- `MOSignedElement`
- `MOMultiplyElement`
- `MOPower`

MOMultiplyElement (MOME)

Cet objet représente un pseudo-conteneur indiquant la position de l'élément fils dans une multiplication, grâce à la propriété `isDivided`. Par défaut l'enfant est considéré comme étant un numérateur, i.e. `isDivided` vaut `false`. Dans le cas où `isDivided` vaut `true`, l'élément est considéré comme un numérateur provoquant le rendu d'une fraction et non plus d'une multiplication.

L'élément fils de cet objet peut être de nature diverse, toujours dérivé du type `MathObjectElement`.

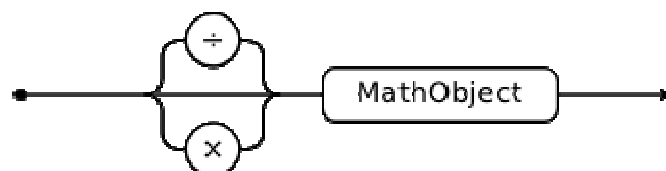


Figure 6 : Objet MOMultiplyElement.

MOMultiplyContainer (MOMC)

Cet objet est un conteneur pour l'opération de multiplication ou de division ; il peut contenir un nombre illimité d'enfants de type MOMultiplyElement. En fonction des éléments fils, le rendu sera différent :

- dans le cas où tous les objets fils ont la propriété *isDivided* a *false*, alors l'objet rendu sera une multiplication.
- dans le cas où au moins un objet fils à sa propriété *isDivided* a *true*, alors on utilise un objet MMLFrac (Fraction) avec deux objets MMLRow, un pour le numérateur et un autre pour le dénominateur.

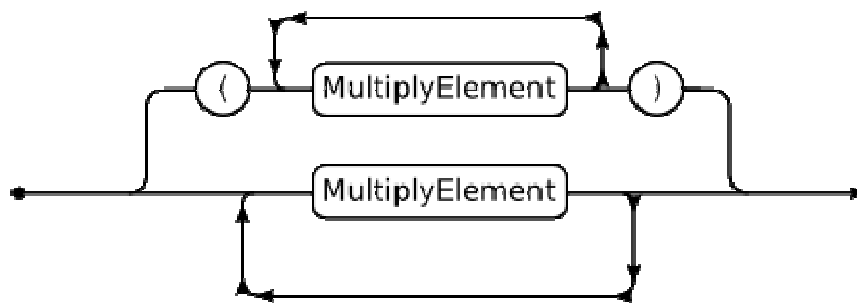


Figure 7 : Objet MOMultiplyContainer.

Les objets MOMC peuvent eux aussi avoir des parenthèses si le parent est lui même de type MOMultiplyElement.

L'INTERFACE UTILISATEUR DE L'APPLICATION

Description

L'interface à laquelle l'utilisateur sera confronté est simple est épurée, elle est composée de 5 parties que nous illustrons ci-dessous.

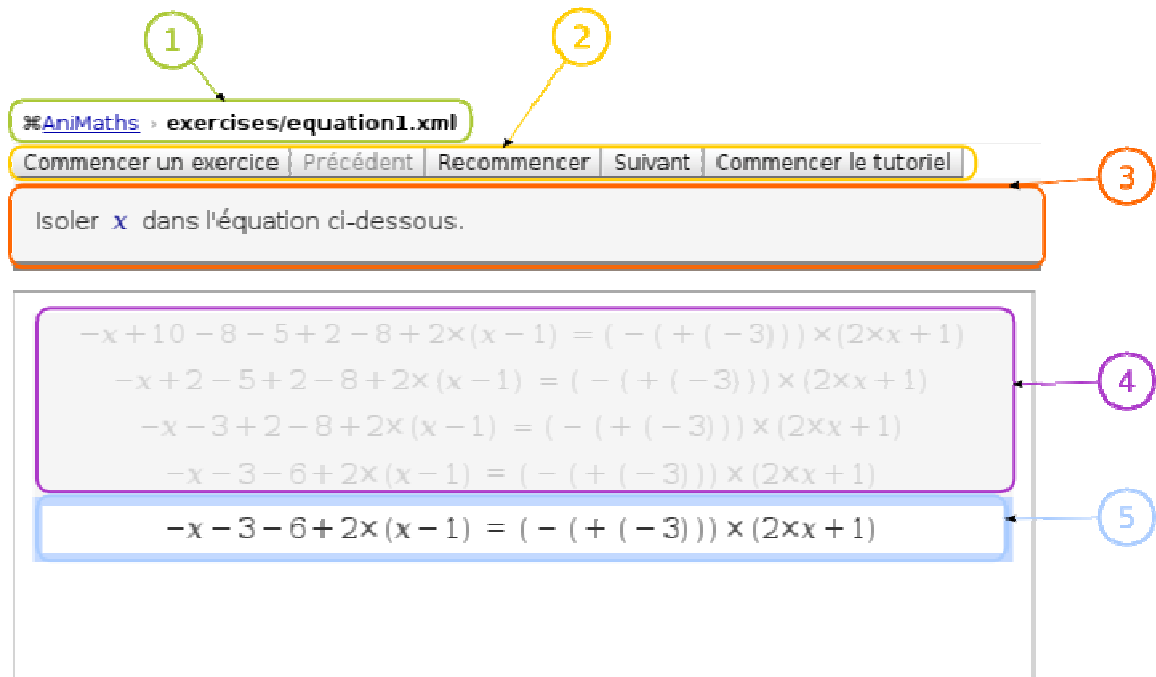


Figure 8: ① Fil d'ariane ; ② zone de d'action(s) rapide(s) ; ③ Consigne de l'exercice ;
④ zone passive de manipulations ; ⑤ zone active de manipulations

Fil d'Ariane

Cette zone n'a pas à l'heure actuelle un rôle très important car notre application n'est composée que d'une seule vue, commune à tous les utilisateurs. À terme, elle a pour but de faciliter la navigation de l'utilisation vers des pages parentes ou précédemment vue.

On notera cependant que dans la dernière version de notre programme, le fil d'Ariane permet de distinguer la situation d'exercice, de la situation de tutoriel ; il permet également de connaître respectivement l'exercice en cours ou l'étape du tutoriel.

Zone d'action(s) rapide(s)

Cette zone contient 5 boutons d'action rapide :

- **[Commencer le tutoriel]** Commence un tutoriel pour aider l'utilisateur à comprendre comment on peut manipuler les objets mathématiques avec AniMaths.
- **[Commencer un exercice]** Permet de passer directement à la résolution de la série d'exercices. Si on est dans le tutoriel, il est arrêté mais pourra être recommencé plus tard.
- **[Précédent]** Permet de revenir à la page précédente du tutoriel. Lors de la résolution d'un exercice, cliquer sur ce bouton arrête l'exercice en cours pour revenir à l'exercice précédent.

- **[Suivant]** Permet de passer directement à la page suivante du tutoriel. Lors de la résolution d'un exercice, cliquer sur ce bouton arrête l'exercice en cours pour passer directement à l'exercice suivant.
- **[Recommencer]** Permet de réinitialiser les lignes d'équations pour recommencer la résolution de l'exercice.

Consigne de l'exercice

Cette zone énonce le problème de façon verbale. Elle peut déjà contenir du HTML et des formules mathématiques, à terme il serait bon qu'elle puisse également contenir des illustrations. Cette zone est statique et ne permet donc aucune manipulation, son rôle est purement ostentatoire.

Zone passive de manipulations

Cette zone évolue au fur et à mesure que l'utilisateur effectue des manipulations sur l'équation. En effet, lorsque celui-ci termine une manipulation, la ligne courante est figée sur l'état *avant* la manipulation, et une nouvelle ligne est également créée, contenant l'équation *après* manipulation.

Zone active de manipulations

Cette zone correspond à l'espace de travail et d'interaction principale de l'utilisateur ; elle correspond à la dernière ligne. Cette ligne est la seule à être interactive, des événements sont déclenchés lors du survol des différents éléments. La partie suivante de ce document décrit les différentes interactions possibles avec la zone active de manipulation et les états visuels associés.

MANIPULATION D'EQUATION

L'objectif de l'application AniMaths étant que les apprenants manipule des objets mathématiques sans trop d'apprentissage, nous avons attaché beaucoup d'importance au caractère intuitif des interactions, ainsi qu'aux retours informatifs. Nous avons ainsi adopté choisi une signalétique adaptée pour indiquer les différents états des objets manipulables ou manipulés : curseurs différents, code couleur, objet qui suit le curseur pendant le déplacement.

Signalétique des objets MathObjects

Etat primaires pour les MathObjects de la zone de manipulation

Défaut

Ce style correspond à l'aspect par défaut des éléments d'une équation, c.-à-d. que les objets ne sont ni survolés, ni sélectionnés ou autre. Lorsqu'une équation est dans cet état, ses éléments sont à priori manipulables, i.e. on peut sélectionner un élément pour le déplacer. Le curseur de la souris est une main avec l'index levé.

$$(- (+ (- 3))) \times (2 \times x + 1)$$

Figure 9 : style par défaut des objets en zone active.

Lorsque l'équation bascule dans la zone passive de manipulation, elle est grisée pour signifier que ce n'est plus manipulable. Dans cette configuration, plus aucune manipulation n'est possible sur les éléments de l'équation. Les lignes de ce type sont conservées pour que l'utilisateur ait une *trace* de son cheminement.

$$(- (+ (- 3))) \times (2 \times x + 1)$$

Figure 10 : style des équations en zone passive.

ATTENTION : LES ÉTATS DÉCRIT PAR LA SUITE SERONT POSSIBLE UNIQUEMENT DANS LA ZONE ACTIVE DE MANIPULATION.

Survol

Les éléments de la zone active survolés sont marqués par un fond de couleur bleu clair. La région bleuté indique l'objet que l'on pourra sélectionner. En se déplaçant de droite à gauche on constatera que l'on peut manipuler des objets plus ou moins complexe. Dans l'exemple ci-dessous on survole un élément de type MOMultiplyElement.

$$(- (+ (- 3))) \times (2 \times x + 1)$$

Figure 11 : Style d'un objet survolé.

Sélection

Une fois que l'utilisateur à choisi l'objet qu'il souhaite déplacer, il sélectionne celui-ci en cliquant dessus. Le clic marque que l'objet est maintenant manipulable, cet état est notifié par un fond de couleur bleu foncé. Le curseur est une main ouverte.

$$(- (+ (- 3))) \times (2 \times x + 1)$$

Figure 12 : Style d'un objet sélectionné.

Déplacement & origine

Lorsqu'on commence à faire glisser l'objet sélectionné, une copie de celui-ci est « attachée » à la souris. L'objet d'origine est de couleur grise quand on essaie de reposer le clone dessus. Le curseur est une main fermée. Par la suite la couleur et le statut de l'objet déplacé changera en fonction de la destination choisie. La section suivante décrit les différents états possibles.

$$+ 2 \times (x - 1)$$
$$+ 2 \times (x - 1)$$

Figure 13 : Style d'un objet sélectionné.

États secondaires pour le MathObject copié attaché au curseur

En fonction de l'objet en cours de déplacement et de la position à laquelle on souhaite le déposer, la copie de l'objet prendra des états différents. On distingue les cas suivants :

Pas d'interaction possible

Cette configuration se rencontre lorsqu'aucune interaction n'est prévue à cet endroit. Par exemple, lorsqu'on déplace un MathObject ailleurs que sur la zone active de manipulation. Un sens interdit est attaché au curseur.

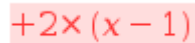

$$+2x(x-1)$$

Figure 14 : Style d'une manipulation non permise.

Interaction automatique possible

Quand l'objet en cours de déplacement et celui de destination forme une manipulation valide alors l'objet déplacé est en vert sur fond vert. Pour le moment, seules la commutation de l'addition et de la multiplication sont des interactions automatiques possibles. Une flèche courbe attachée au curseur.

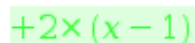

$$+2x(x-1)$$

Figure 15 : Style d'une manipulation automatique possible.

Interaction soumise à question

Ce style correspond à des manipulations pour lesquelles une question sera posée à l'utilisateur avant que l'interaction soit validée. On utilise une boîte de dialogue pour poser cette question. Une flèche courbe attachée au curseur.

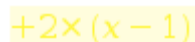

$$+2x(x-1)$$

Figure 16 : Style d'une manipulation soumise à question.

Les boîtes de dialogues proposent deux types de d'interaction :

- l'utilisateur entre un résultat au clavier (ex : addition ou multiplication de deux termes)
- l'utilisateur choisit un résultat parmi plusieurs proposés (ex : passer un terme dans l'autre membre).

Les différentes manipulations et questions associées sont abordées au chapitre suivant.

Description des zones d'interaction

Où a-t-on le droit de lâcher un objet MathObject ? Dans une zone qui va bien ! Avant de chercher à savoir dans quelle zone d'un objet mathématique le curseur se situe, nous devons définir ces zones ! Nous proposons ici un méthode de quadrillage des objets MathObject, expliquée sur le dessin suivant :

Les petits traits sont les deux bords et le milieu. La distance entre les petits traits et les grands traits les plus proches est de 10 pixels. on définit les zones :

- ZONE_OOO = -3
- ZONE_OO = -2
- ZONE_O = -1
- ZONE_C = 0
- ZONE_E = 1
- ZONE_EE = 2
- ZONE_EEE = 3

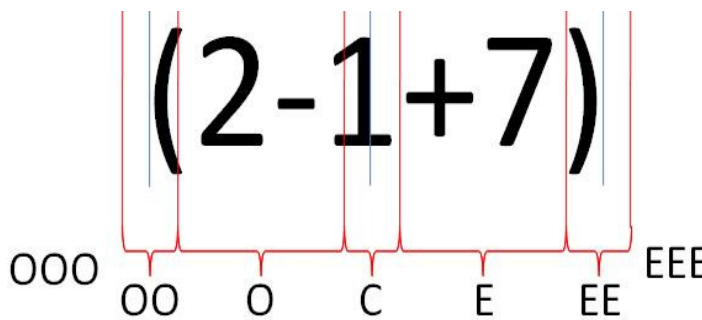


Figure 17 : découpage d'un objet MathObject en zones : horizontale.

Ces zones et leur code numérique nous permettent de déterminer rapidement si on est sur le bord droit ou le bord gauche par exemple. Le même découpage est fait suivant la verticale, ce qui permet de définir des interactions assez complexes.

Deux exceptions à cette façon de découper en zones :

- Dans le cas des équations, nous centrons la petite ligne verticale du milieu non pas sur le milieu de l'objet, mais sur le milieu du signe égal.
- Dans le cas des MOMultiplyContainer, si il est représenté par une fraction, on centre la ligne horizontale du milieu sur la barre de fraction.

Les boîtes de dialogue : choix ou nécessité ?

Comme nous le disions, nous avons attaché beaucoup d'importance au caractère intuitif des interactions et aux retours informatifs. Nous pensons que les boîtes de dialogue sont très utiles :

- Dans le cas des questions, l'utilisateur peut s'assurer qu'il obtiendra bien ce qu'il souhaite à l'issue de la manipulation
 - Si il donne une réponse correcte, on lui dit, et on lui apporte quelques informations supplémentaires.
 - Si il donne une réponse incorrecte, on le prévient, et dans la mesure du possible on lui donne des pistes pour ne pas se tromper la prochaine fois.
- Lors d'un long temps de traitement, il est bon d'informer l'utilisateur pour qu'il ne s'impatiente pas.

A chaque fois qu'il y a une question où l'utilisateur doit choisir parmi un ensemble de réponses, on affiche les réponses dans un ordre aléatoire. Cela permet d'une part d'éviter que la bonne réponse soit toujours à la même place, et d'autre part cela empêche l'utilisateur d'utiliser la technique de la recherche systématique de la bonne réponse, et si l'utilisateur se trompe souvent, il sera obligé de se concentrer pour mémoriser les réponses qui sont fausses.

Voici quelques exemples de boîtes de dialogue dans notre application :

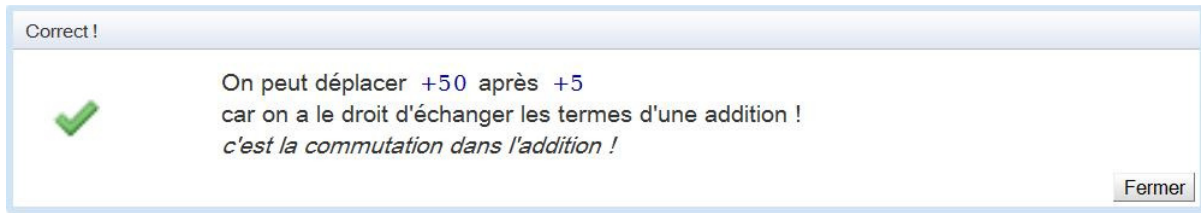


Figure 18 : Lors de la commutation dans l'addition



Figure 19 : $2 \times (-2) = 12$?



Figure 20 : $2 \times (-2) = 4$?

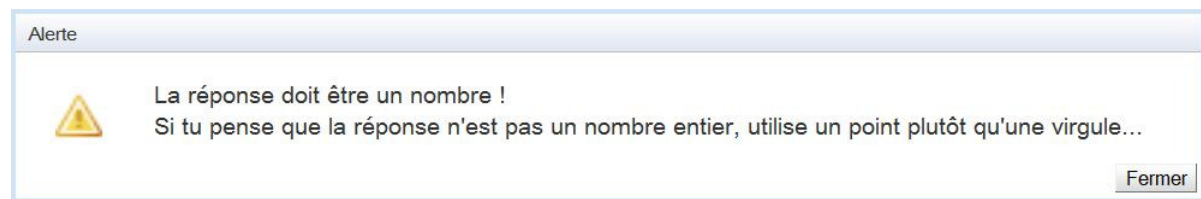


Figure 21 : $2 \times (-2) = sq654kml3254fh$?

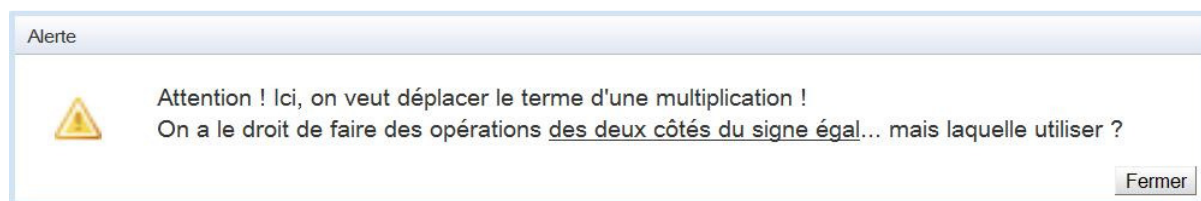


Figure 22 : L'utilisateur s'est trompé !

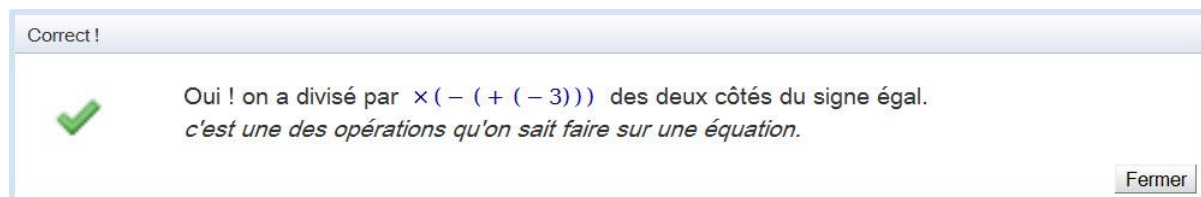


Figure 23 : Même cas de figure, l'utilisateur ne s'est pas trompé !

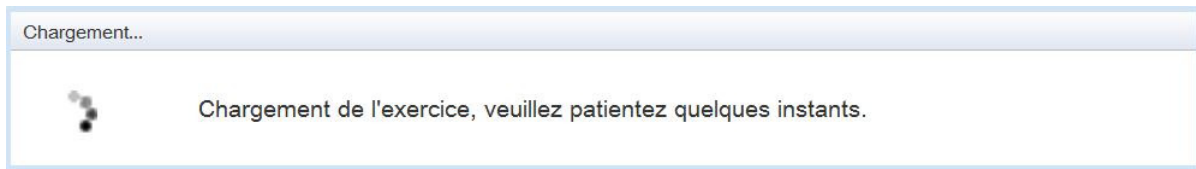


Figure 24 : Dans le cas d'un chargement qui risque d'être long.

Lorsque c'est utile, nous avons utilisé un compte à rebours pour limiter la durée d'apparition des boîtes de dialogue. Dans ce cas, nous avons également ajouté un écouteur DOM pour fermer la fenêtre si l'utilisateur appuie sur la touche entrée.

Alors ? Les boîtes de dialogues, choix ou nécessité ?

On pense que ces boîtes de dialogues sont un très bon choix à la fois d'un point de vue de l'interaction personne-systèmes, et à la fois d'un point de vue pédagogique. Mais aurait-on pu éviter de les utiliser ? rien n'est moins sûr ! En effet, nous avons fait le choix initial de modifier le MathObject copié attaché au curseur *pendant son déplacement*, c'est à dire : quand on passe un terme x_5 de l'autre côté du signe égal, on voit visuellement qu'il devient $x(1/5)$... C'eût été beau !

Nous avons donc implémenté un certain nombre de telles interactions. Dans ce cas trivial c'est simple, seulement voilà : admettons qu'on ait une fraction sur une fraction sur une fraction... comment se comporte le x_5 du dénominateur du dénominateur du dénominateur si on cherche à le placer de l'autre côté du signe égal ?? Pas si simple ! Et pourquoi le x_5 se change une fois sur deux alors ? C'est pour éviter ce genre de frustration et de questionnement, que nous avons abandonné cette idée et que désormais, le MathObject copié attaché au curseur reste exactement égal à l'objet sélectionné pendant son déplacement.

MANIPULATIONS IMPLEMENTEES

Nous avons implémenté un certain nombre d'interactions, et la possibilité très importante de ne sélectionner que certaines d'entre elles pour la résolution, à l'aide d'un paramètre numérique noté *level*. Ainsi, plus le level est haut, plus il y aura d'interactions possibles. On peut aussi imaginer qu'à partir d'un certain niveau, une interaction vienne en remplacer une autre (processus automatique plutôt que par questionnement par exemple).

Nous avons également codé la possibilité de n'interagir qu'avec le membre de droite (pour la simplification d'une expression par exemple. Ces deux paramètres pourront être décrits dans chaque fichier exercice, ou adaptés au niveau de l'élève...

Nous présentons dans les sections suivantes les différentes manipulations que l'on peut faire à l'heure actuelle.

Addition

Commutation

Il est possible de commuter les opérands d'une addition, par exemple passer de $x + y$ à $y + x$.

Calcul

Lorsque l'on fait glisser-déposer un opérande d'addition sur un autre, alors un pop-up apparaît pour demander le résultat de l'addition de ces éléments.



Figure 25 : Pop-up apparaissant lorsque l'on fait glisser le 2 sur le -5 ou réciproquement.

Simplification de signe

Il peut arriver que certaine équation ou partie d'équation soit écrite de façon très verbeuse et où l'indication du signe est superflu comme par exemple.

$$(-(+(-3)))$$

Si on déplace un `MOSignedElement` sur un autre `MOSignedElement` qui est son parent ou son enfant, une boîte de dialogue propose de simplifier le résultat :

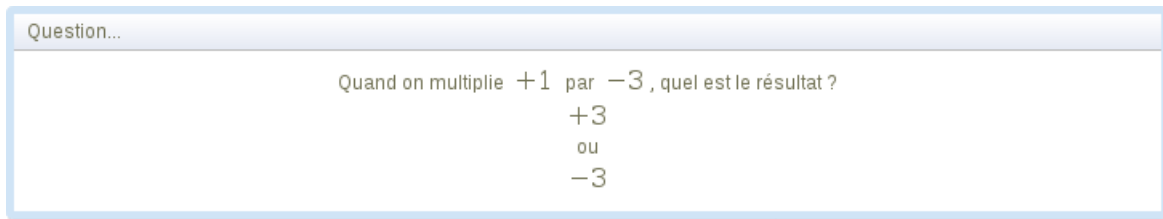


Figure 26: Simplification de signe.

Multiplication

Commutation

Dans la version actuelle du programme, il est possible de commuter les opérands d'une multiplication, tant qu'il sont tous deux du même côté de la barre de fraction.

Calcul

Comme pour l'addition, glisser-déposer un opérande de multiplication sur un autre fait apparaître une boîte de dialogue pour demander le résultat de ce produit :

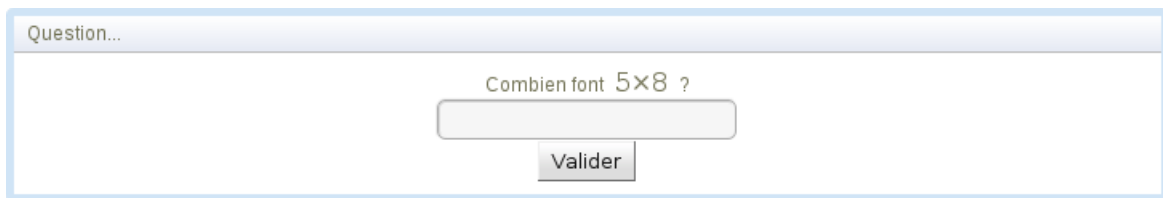


Figure 27 : Pop-up apparaissant lorsque l'on fait glisser le 8 sur le 5 ou réciproquement.

Équation

Terme additif ou multiplicatif

Dans quelques cas, l'utilisateur peut passer des termes d'un membre à l'autre de l'équation. Là encore une boîte de dialogue lui apparaît et lui demande de choisir le résultat de son action parmi plusieurs.

Question...

Après avoir déplacé $+2 \times (x - 1)$ de l'autre côté du signe égal, quel est le résultat ?

$$-x + 10 - 5 + 2 - 8 - 8 = \frac{(- (+ (-3))) \times (2 \times x + 1)}{(2 \times (x - 1))}$$

ou

$$-x + 10 - 5 + 2 - 8 - 8 = 2 \times (x - 1) + (- (+ (-3))) \times (2 \times x + 1)$$

ou

$$-x + 10 - 5 + 2 - 8 - 8 = -2 \times (x - 1) + (- (+ (-3))) \times (2 \times x + 1)$$

ou

$$-x + 10 - 5 + 2 - 8 - 8 = (2 \times (x - 1)) \times (- (+ (-3))) \times (2 \times x + 1)$$

Figure 28 : Passage de $2x(x+1)$ du membre de droite à celui de gauche, l'utilisateur doit choisir le bon résultat pour que l'opération soit validée.

Terme orphelin

Déplacer un terme orphelin d'un membre à l'autre est la manipulation qui laisse le plus de possibilités à l'utilisateur :

Question...

Après avoir déplacé -11 de l'autre côté du signe égal, quel est le résultat ?

$$x - 31 - 11 = 1$$

ou

$$\frac{(x - 31)}{(-11)} = 1$$

ou

$$x - 31 + 11 = 1$$

ou

$$x - 31 + 11 = 0$$

ou

$$(x - 31) \times (-11) = 0$$

ou

$$\frac{(x - 31)}{(-11)} = 0$$

ou

$$(x - 31) \times (-11) = 1$$

ou

$$x - 31 - 11 = 0$$

Figure 29 : Passage de -11 du membre de droite à celui de gauche, l'utilisateur peut choisir la n°2 ou la n°8

Résumé des choix retenus lors de la première phase

Nous avons choisi de développer une application web, le navigateur Mozilla Firefox s'est imposé car c'est le seul à gérer nativement les objets MathML, ce qui est primordial pour ce projet.

Pour les technologies de manipulation, nous avons choisi de développer notre application en Java, avec la boîte à outils Google Web Toolkit (GWT) dans sa version 2.0 de décembre 2009, qui propose un compilateur puissant du Java vers le JavaScript, un plugin pour Mozilla Firefox qui permet d'exécuter l'application sans nécessiter de compilation, une bibliothèque d'émulation JRE bien utile, ainsi qu'une bibliothèque de composants graphiques que nous avons étendus pour les objets MathML.

Nous avons développé notre application sous Eclipse, car nous connaissons bien cet environnement de développement intégré de travail, et que GWT propose un bon plugin pour Eclipse.

Pour collaborer au mieux, nous utilisons le gestionnaire de fichier SVN, par le biais du plugin Subclipse d'Eclipse. Nous avons déposé notre programme sur un compte Google Code à l'adresse suivante :

<http://code.google.com/p/animaths/>.

Le projet est placé sous licence Apache 2.0, conformément à notre souhait de permettre à qui le souhaite de réutiliser notre framework.

Nous avons choisi d'adopter au mieux les bonnes pratiques de développement d'applications GWT proposées lors de la conférence annuelle des développeurs Google (Google I/O '09), en particulier :

- nous utilisons un bus d'évènements (EventBus) pour assurer un couplage très faible entre les différents composants de l'application et ainsi rendre le code "propre" ;
- nous essayons de respecter le paradigme de conception d'interface Modèle-Vue-Animateur (Model-View-Presenter ou MVP) ;
- Nous utilisons la librairie GIN (GWT INjection) d'injection automatique de dépendance pour alléger le code d'initialisation des classes.

Par le respect de ces bonnes pratiques, nous espérons accroître la maintenabilité du code, et sa reprise potentielle par d'autres développeurs.

Pour le détail de tous ces choix, nous proposons au lecteur de se référer à notre rapport de la première phase de ce projet de génie logiciel.

Architecture globale du projet

Le programme comporte un ensemble de ressources statiques dans le dossier **war**, dont un fichier html, coquille vide de l'application, des fichiers css de feuilles de style en cascade, les ressources graphiques et un ensemble de fichiers XML dans les dossiers **war/exercices** et **war/tutorials** qui contiennent une description XML de l'équation MOEquation à résoudre, et un énoncé (html + description XML des MathObjects imbriqués). Nous reviendrons dans une prochaine partie sur ces fichiers XML, leur utilité, leur structure, et leur utilisation.

Le package principal de notre application GWT est **fr.upmf.animaths**.

serveur comporte les servlets qui sont utilisées pour la communication AJAX avec le client, que nous détaillerons dans une des parties suivantes.

client comporte toutes les classes qui seront transformées en JavaScript par le compilateur de GWT. On a principalement **client.gin** pour l'injection automatique de dépendance, **client.mvp** pour la définition des animateurs et des vues (Widgets), **client.event** pour les événements de l'application gérés par l'EventBus, qui assurent un couplage faible entre les composants de l'application, et enfin **client.interaction** pour les classes métiers de gestion des interactions.

Le package client.mvp

client.mvp contient un ensemble de couple classe animateur - classe vue.

- AniMathsPresenter et AniMathsDisplay : classes principales de l'application et point d'entrée. Elles décrivent l'interface utilisateur, les boutons de navigation rapide ; la communication avec le serveur ; la gestion des autres animateurs.
- AniMathsMessageBox : boîte de dialogue pour les retours informatifs : informations ; chargements ; bonnes réponses ; avertissements ; erreurs.
- MOAbstractPresenter : classe générique qui affiche les équations dans un élément div.
- MOBasicPresenter et MOBasicDisplay : utilisés pour afficher des équations passives.
- MODynamicPresenter et MODynamicDisplay : pour la zone active de manipulation. Cet animateur écoute les événements DOM, les traduit en événements bas-niveau propres à l'application qu'il transmet alors à l'EventBus. Ces événements comportent les informations nécessaires et suffisantes pour initier les manipulations.
- MODragPresenter (couple avec MOBasicDisplay): affiche son équation passive dans l'élément <div id="drag"> de l'application, et gère le positionnement absolu de cet élément.
- MOFocusWidget : est utilisé dans les boîtes de dialogue. Il contient un MOBasicPresenter qui peut obtenir le focus et sur lequel on peut cliquer.
- MOWordingWidget : est utilisé pour les énoncés où sont imbriqués du HTML et des MathObject.

Ce package contient également deux classes que nous avons mentionnées lors de la soutenance de la première phase, que nous n'utilisons pas dans cette version du programme, mais qui seront bien utiles :

- La classe MOProcessWidget complètera la trace des manipulations par un énoncé (MOWordingWidget) décrivant le passage entre deux étapes, et un bouton "rejouer" pour animer les manipulations à partir de cette étape.
- La classe MOEquationGenerator dont nous avons fait la démonstration, permet de générer aléatoirement une équation.

Implémentation des éléments MathML

Le package `client.mvp.MathML` contient le sous-ensemble des éléments de la spécification MathML décrit dans le chapitre 2 et adaptés à nos besoins en tant que Widgets.

Nous avons rencontré quelques difficultés concernant l'implémentation des éléments MathML. Ces difficultés sont imputables au fait que nous travaillons avec des technologies de pointe, encore jeunes et instables :

Tout d'abord un obstacle très limitant lors de la première phase du projet : afficher correctement des éléments MathML ajoutés au DOM par du JavaScript. Après bon nombre de recherches et d'essais, on s'est rendu compte que la difficulté serait levée si on pouvait utiliser un doctype (X)HTML 4.0 ou 5.0, car Mozilla Firefox gère nativement l'affichage des éléments MathML dans ces formats. Malheureusement, par souci de portabilité les développeurs de GWT ont décidé d'imposer le HTML. Nous avons dû trouver le moyen d'accéder aux différentes fonctionnalités natives de gestion DOM, en particulier pour la déclaration d'un espace de nom. Finalement, notre implémentation fonctionne pour Mozilla Firefox, mais elle sera évidemment différente pour Internet Explorer. Nous avons néanmoins proposé une implémentation pour ce navigateur, que nous n'avons pas pu tester. Toutefois, lorsqu'il sera capable d'afficher nativement du MathML, la modification du code que nous avons proposé sera réduite ; voire inexistante.

Autre difficulté, et pas des moindres : L'implémentation des éléments MathML dans MozillaFirefox est encore incomplète et hasardeuse !

- En particulier, il est impossible d'accéder simplement à l'attribut `id`, ni à la classe, ni à l'objet JavaScript *style*. Ces lacunes nous ont empêché d'utiliser les émulations des fonctions JavaScript en Java proposées par GWT. Nous avons donc dû gérer nous-même les attributs des éléments MathML, en particulier les attributs de styles.
- Il est également impossible d'accéder directement aux fonctions `getClientBounding...` pour connaître la position des éléments dans la page. Pour obtenir ces informations, nous avons utilisé des fonctions dites natives, qui décrivent du code JavaScript, qui sera traité différemment par le compilateur de GWT.
- Enfin, les éléments MathML dans leur version actuelle ne peuvent pas gérer les événements DOM ! Nous ne pouvons donc pas leur attacher d'écouteurs. Nous avons contourné les obstacles induits en ajoutant les écouteurs au `MODynamicPresenter` et en y maintenant une `Map` qui rassemble les couples `ElementDOM - MathObject`. Ainsi, lorsqu'un événement DOM est lancé, il est possible de savoir au-dessus de quel élément du DOM le curseur se situe, et de remonter à l'élément `MathObject` possesseur de cet élément.

La plupart des classes du package `client.mvp.MathML` dérivent de la classe `MMLElement`. Cette classe possède les constructeurs et fonctions pour contourner les difficultés décrites ci-dessus. Elle possède en particulier la gestion des classes et du style, et des fonctions natives pour obtenir les positions des éléments MathML.

Les objets MathObject

Le package `client.mvp.MathObject` contient les objets `MathObject` décrits dans le chapitre 2. La fonction principale chacune des classes de ce package est la fonction `pack`, qui permet de créer récursivement les widgets `MMLElement` et de les ajouter au bon élément du DOM.

POO et critique du MVP

La classe `MOElement` est générique. Elle contient une fonction très importante : `needsFence`, qui évalue si un `MathObject` doit être entre parenthèse en fonction de la priorité des opérateurs mathématiques, et on a défini un certain nombre d'interfaces qu'implémentent les objets `MathObject` :

- `IMOHasStyleClass` regroupe les différentes constantes pour gérer l'attribut `class` des objets, et ainsi gérer leur apparence.
- `IMOHasSign` pour les `MOSignedElement` et `MOMultiplyElement` ;
- `IMOHasFence` pour les objets qui peuvent être être parenthèses ;
- `IMOHasOneChild` pour `MOSignedElement` et `MOMultiplyElement` ;
- `IMOHasSeveralChild` pour `MOAddContainer` et `MOMultiplyContainer` ;
- `IMOHasValue` pour `MONumber` et `MOIdentifier` ;
- `IMOHasZone` pour définir les zones d'interaction sur un élément ;
- `IMOSelection` définit des fonctions pour naviguer parmi les objets sélectionnables. Nous avons fait un certain nombre de choix de conception que nous ne détaillerons pas ici. Le mieux est d'essayer directement sur le programme !

Chaque objet `MathObject` est scindé en deux : un animateur (ex `MOEquation`) et une vue (`MOEquationDisplay`). La vue possède les widgets `MathML`. Nous avons fait ici un exercice de style mais nous pensons que compte tenu de la simplicité des classes vue, le respect du paradigme MVP est discutable dans le cas de ces classes...

exigences sur l'implémentation des objets MathObject

Dans notre programme, nous avons souhaité limiter l'utilisation des `MathObjects` lorsqu'ils ne sont pas nécessaires.

- Ainsi, si un `MOSignedElement` positif est fils d'une équation (`MOEquation`), le signe plus est implicite, donc inutile. Dans ce cas, `MOSignedElement` laissera sa place à son élément fils.
- Un `MOMultiplyElement` n'a le droit d'exister qu'au sein d'un `MOMultiplyContainer`.
- De même, si un `MOAddContainer` n'a qu'un fils, il laissera sa place à ce fils.
- Dans le cas du `MOMultiplyContainer`, c'est un peu plus compliqué. Si il possède un `MOMultiplyElement`, qui est au numérateur, alors il pourra lui laisser sa place. Par contre, si il possède un ou plusieurs `MOMultiplyElement` au dénominateur, et aucun au numérateur, alors on rajoutera une objet de type `MOMultiplyElement`, de propriété `isDivided` à `false`, et qui contiendra un `MONumber` qui vaudra 1

Pour le moment, par souci de temps, nous avons légué aux différentes interactions décrites ci-dessous le soin de vérifier qu'aucun objet `MathObject` n'est inutile. Il serait tout-à-fait envisageable, et même souhaitable, de factoriser ces bouts de code et de laisser aux `MathObjects` le soin de se supprimer lorsqu'ils sont inutiles.

Le package client.interaction

Le package **client.interaction** est le coeur de l'application qui apporte l'interactivité à l'objet MODynamicPresenter. Son cœur dur contient les classes principales suivantes :

- AniMathsCoreInteraction : classe principale pour l'interaction. Elle écoute les évènements générés par la classe MODynamicPresenter, elle gère la sélection, et elle agit en tant que médiateur entre les processus intéressés par une interaction pour ne donner la main qu'à un seul d'entre eux.
- AniMathsAbstractProcess : classe abstraite dont dérivent tous les classes processus. Cette classe factorise beaucoup de code de sorte que les classes processus n'aient quasiment plus à dialoguer avec AniMathsCoreInteraction. Cette classe est faite pour faciliter la maintenabilité du code et l'ajout de processus d'interaction.
- AniMathsQuestionButton et AniMathsQuestionTextBox sont deux classes qui définissent des boîtes de dialogues, pour poser une question à l'utilisateur avant de procéder à la manipulation.
 - AniMathsQuestionButton expose un certain nombre de propositions à l'utilisateur, qui doit choisir celle qui lui semble juste
 - AniMathsQuestionTextBox demande à l'utilisateur de rentrer une réponse au clavier.

En plus de ces classes, nous avons déjà implémenté quelques classes processus :

- SEs_N_Add : additionner deux nombres entre eux. Ces nombres peuvent être positifs ou négatifs.
- MEs_N_Multiply : multiplier deux nombres entre eux. Ces nombres peuvent être positifs ou négatifs.
- SEs_AC_Commutation : commuter les termes d'une addition.
- MEs_MC_Commutation : commuter les termes d'une multiplication.
- SEs_SEs_ChangeSign : simplifier les signes.
- Ns_Is_E_ChangeHandSide : passer un terme orphelin de l'autre côté du signe égal.
- SEs_AC_E_ChangeHandSide : passer un terme additif de l'autre côté du signe égal (incomplet)
- MEs_MC_E_ChangeHandSide : passer un terme multiplicatif de l'autre côté du signe égal (incomplet)

Le package client.event

L'ensemble des évènements de l'application sont définis dans le package **client.event**. Ces évènements Event et leurs interfaces Handler associées sont générés à différent niveaux.

Certains sont directement traduits à partir des évènements DOM ; ils sont donc générés très fréquemment:

- FlyOver : correspond à un survol d'un élément MathObject.
- Selection : correspond au click sur un élément MathObject.
- SelectionChange : appui sur l'une des touches q;s;d;z pour changer la sélection.
- Grab : bouton enfoncé sur un élément
- Drag : élément déplacé avec le bouton enfoncé
- Drop : élément relâché

D'autres sont générés lors du dialogue entre le médiateur et les différents processus ; Il sont générés beaucoup moins fréquemment.

- GrabSelected : bouton enfoncé sur l'élément qui était sélectionné

- **ProcessInterested** : un des processus est concerné par les manipulations qui peuvent être faites sur cet élément.
- **DragSelected** : l'élément qui était sélectionné est déplacé
- **TagDeclaration** : un processus se déclare intéressé ou non par le lâché du MathObject déplacé, si il est intéressé ça veut dire qu'une manipulation est possible si on relâche à cet endroit (le tag en question, c'est TAG_OK, TAG_CAUTION ou TAG_NO)
- **DropSelected** : l'élément déplacé est relâché.
- **ProcessLaunch** : un processus est lancé.
- **ProcessDone** : un processus vient de terminer (pas forcément validé...).

Finalement, ces deux derniers évènements sont écoutés par AniMathsPresenter, et sont plutôt rare en comparaison aux deux précédentes catégories.

- **NewLine** : Une manipulation vient d'être validée, ajouter une ligne aux équations statiques...
- **ExerciseSolved** : L'exercice est résolu, ouvrir une boîte de dialogue pour demander à l'utilisateur quoi faire...

Description d'un processus

On présente dans le tableau ci dessous un extrait du dialogue entre le médiateur et un processus, en insistant sur les parties de la communication et du processus qui sont gérées par la classe abstraite AniMathsAbstractProcess.

A la lecture de ce tableau, on se rend bien compte qu'il reste très peu de choses à gérer dans les classes processus concrètes. L'effort que nous avons fait pour factoriser le code dans AniMathsAbstractProcess nous permet de développer beaucoup plus rapidement et avec plus de facilité de nouveaux processus d'interaction. C'est gage de maintenabilité et d'évolutivité !

E:évènement / F:Fonction	AniMathsCoreInteraction	AniMathAbstractProcess	ConcreteProcess
- E - Grab	si l'évènement a lieu sur un objet préalablement sélectionne : on génère un évènement <i>GrabSelected</i>		
- E - GrabSelected		on stocke l'objet sélectionné, puis on appel la fonction <i>isProcessInvolved</i>	
- F - isProcessInvolved		si la réponse est oui : on génère l'évènement <i>ProcessInterested</i>	est-ce que le processus peut proposer une interaction avec l'objet sélectionné ?
- E - ProcessInterested	Préparer le déplacement, écouter les évènements <i>onDrag</i>		
- E - Drag	Bubbling : on génère des évènements <i>DragSelected</i> à partir de l'intérieur (objet survolé) vers l'extérieur (équation)		
- E - DragSelected		on récupère la zone et l'élément survolé (<i>where</i>)	

		et on appel à chaque fois la fonction <i>getTagOfProcess</i>	
- F - getTagOfProcess		si le tag est supérieur à celui enregistré, on sauvegarde cette interaction (<i>MathObject choosenWhere</i>) et on génère un évènement <i>TagDeclaration</i>	est-ce que le processus peut proposer une interaction sur cette zone de cette objet ? renvoie le tag.
- E - TagDeclaration	si le tag est supérieur à celui enregistré, on gere la signalétique correspondante, on stocke le process qui s'est annoncé, et on lui ajoute un écouteur de l'évènement <i>ProcessLaunch</i>		
- E - Drop	générer un évènement <i>DropSelected</i> et <i>ProcessLaunch</i>		
- E - DropSelected		réinitialiser les écouteurs	
- E - ProcessLaunch		replacer <i>choosenWhere</i> dans <i>where</i> et <i>choosenZone</i> dans <i>zone</i> , appeler la fonction <i>questionAsk</i>	
- F - questionAsk		Afficher une boite de dialogue de chargement et appeler la fonction <i>onQuestionAsk</i>	
- F - onQuestionAsk			Poser la question, Si l'utilisateur répond correctement, ça appelle la fonction <i>ExecuteProcess(>o)</i> , sinon, <i>ExecuteProcess(<o)</i>
- F - executeProcess(i)		Afficher un retour informatif en fonction du paramètre, les infos sont récupérées à l'aide de la fonction <i>getMessage(i)</i> . Si $i > 0$, appeler la fonction <i>onExecuteProcess</i> , puis générer l'évènement <i>ProcessDone</i> et enfin mettre à jour le <i>MODynamicPresenter</i>	
- F - getMessage(i)			Quel est ton message ?
- F - onExecuteProcess			feu vert pour modifier l'équation !
- E - ProcessDone	Générer l'évènement <i>NewLine</i> (puis gérer le scroll de la zone d'interaction)		

Stockage des données sur le serveur

Comme nous l'avons décrit lors de la présentation de la phase 1, nous souhaitons que les exercices soient stockés sur le serveur. Cette idée a plusieurs intérêts :

1. permettre l'accès à notre projet à davantage de personnes, quel que soit leur localisation et leurs contraintes horaires ;
2. permettre aux enseignants de partager leurs ressources.

Considérer cette approche dès le début nous permet de poser des jalons pour les développements futurs. Le second point, par exemple, laisse la possibilité d'une évolution profitant des techniques de réseaux sociaux. On pourrait en effet envisager d'implémenter un wiki, un système de votes pour les exercices type digg.com ; cela permettrait ainsi de faire émerger les exercices les plus pertinents.

Pour le moment, nous utilisons cette approche pour récupérer les énoncés et les équations des exercices, les fiches du tutoriel ainsi que la liste de ces éléments.

Communication client-serveur (RPC)⁵

La plupart des navigateurs n'accordant qu'un seul Thread pour exécuter le JavaScript les échanges entre le client et le serveur doivent être faits de façon asynchrone. Dans le cas contraire notre application sera bloquée le temps que la réponse à une requête arrive.

GWT possède une architecture propre pour faire des appels RPC (*Remote Call Procedure*) : il faut écrire le service côté serveur et la vue qu'en a le client. Chacun de ces services doit être placé respectivement dans le package *fr.upmf.animaths.server* et *fr.upmf.animaths.client*. L'interface client est quand à elle doublée par une interface asynchrone.

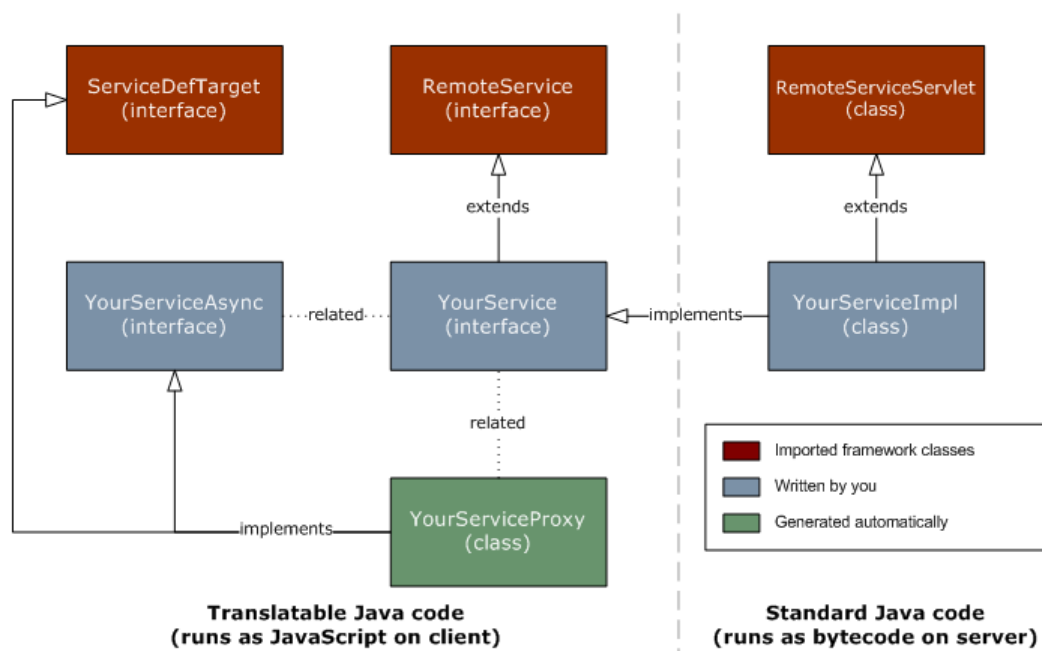


Figure 30 : Schéma d'architecture RPC (Plumbing Diagram)GWT⁶.

⁵ Communicate with a Server @ Google: <http://code.google.com/webtoolkit/doc/latest/DevGuideServerCommunication.html#DevGuideHttpRequests>

Codage

La création d'un *service* GWT distant se crée en trois étapes :

1. Définition des interfaces (une synchrone pour la servlet, une asynchrone pour le client AJAX) ;
2. Implémentation du service ;
3. Écriture de l'appel au service (décrit partiellement dans les sections concernant les exercices et le tutoriel).

Définition de l'interface client du service⁷

Le code suivant correspond au code qui serait exécuté côté client pour effectuer des appels RPC du service en JavaScript.

```
package fr.upmf.animaths.rpc.client;
import com.google.gwt.user.client.rpc.RemoteService;

public interface LoadEquationService extends RemoteService {
    /* liste des méthodes */
    public String sayHello (String name);
}
```

Implémentation coté serveur du service

```
package fr.upmf.animaths.rpc.server;
import com.google.gwt.user.server.rpc.RemoteServiceServlet;
import fr.upmf.animaths.rpc.client.RemoteExerciceManager;

public class LoadEquationServiceImpl extends RemoteServiceServlet
implements RemoteExerciceManager {
    @Override
    public String sayHello(String name) {
        // corps de fonction
    }
}
```

Définition de l'interface asynchrone du service, elle sera appelé du coté client

Cette interface est calquée sur l'interface du service à la différence que tous les méthodes retournent *void* et qu'elle prennent un argument supplémentaire de type *AsyncCallback*.

```
package fr.upmf.animaths.rpc.client;
import com.google.gwt.user.client.rpc.AsyncCallback;
```

⁶ RPC Plumbing Diagram :

<http://code.google.com/webtoolkit/doc/latest/DevGuideServerCommunication.html#DevGuidePlumbingDiagram>

⁷ Interface concept : <http://java.sun.com/docs/books/tutorial/java/concepts/interface.html>

```

interface LoadEquationServiceAsync {
    /* liste des méthodes. */
    void sayHello (String name, AsyncCallback<String> callback);
}

```

La classe *AsyncCallBack* contient deux méthodes qui permettent de gérer le résultat de l'appel : *onSuccess(Object result)* qui sera exécuté si la requête aboutit et retourne l'objet *result*, *onFailure(Throwable ex)* en cas d'échec, qui jettera une exception.

Ils faut par la suite associer une URL au service pour que celui soit accessible, ceci correspond aux lignes suivantes dans le web.XML :

```

<servlet>
  <servlet-name>LoadEquationServiceImpl</servlet-name>
  <servlet-class>fr.upmf.animaths.server.LoadEquationServiceImpl</servlet-
class>
</servlet>
<servlet-mapping>
  <servlet-name>LoadEquationServiceImpl</servlet-name>
  <url-pattern>/animaths/loadEquationService</url-pattern>
</servlet-mapping>

```

Ainsi que la ligne suivante dans *AniMaths.gwt.XML* :

```

<inherits name='com.google.gwt.user.User' />

```

Nous n'irons pas plus loin dans les détails du fonctionnement des appels RPC. pour plus d'informations, le lecteur se référera à la documentation officielle⁸ ou à la table de références. Nous allons maintenant nous attacher à décrire l'utilisation que nous avons fait de ces services, pour la gestion des exercices et du tutoriel.

Exercices & Tutoriaux

L'objectif premier de la communication avec le serveur était de pouvoir récupérer les énoncés et les équations correspondants à un exercices. Une autre fonctionnalité envisagé, mais non implémenté, était de pouvoir sauver un exercice et les manipulations sur le serveur.

Structure des fichiers

Les exercices sont placés dans le dossier *war/exercises/* et sont des fichiers aux format XML. Nous avons décidé d'utiliser le format XML plutôt que JSON. En effet, même si il génère plus de trafic, il permet une gestion et une manipulation aisée à la fois pour la machine et l'individu.

De plus, ce format nous permet de déclarer une DTD ou un schéma XSD permettant de contrôler la validité du document. Tandis que JSON aurai rendu le validation complexe et lourde à gérer.

⁸ Making Remote Procedure Calls @ Google : <http://code.google.com/webtoolkit/doc/latest/tutorial/RPC.html>

Au commencement de la communication client-serveur, nous avons utilisé les fichiers XML uniquement pour décrire l'équation d'un exercice. Par la suite il nous à paru logique d'ajouter l'énoncé à ces fichiers. Le fichier est structuré comme suit :

- `<exercice>`, l'élément racine. Il peut contenir l'attribut `level`
 - un élément `<wording>`, qui contient l'énoncé de l'exercice. Il peut contenir du HTML, des graphique et bien sûr du MathML ;
 - un élément `<moe>` qui correspond à la description d'une équation.

```
<?XML version="1.0" encoding="UTF-8"?>
<!-- <!DOCTYPE agency SYSTEM "AniMaths.dtd" -->
<exercice level="5">
  <wording>
    <!-- Isoler --><moi>x</moi> <!-- dans l'équation ci-dessous. -->
  </wording>
  <moe>
  <moac>
    <mose isMinus='true'>
    <moi>x</moi>
  ...
  </momc>
</moe>
</exercice>
```

Analyse

Le service implémenté coté serveur prend en paramètres un nom de fichier est retourne sont contenu sous forme de chaîne de caractère. Le contenu retourner étant une chaîne de caractère représentant un fichier XML, il nous faut la *parser*. On utilise pour ça la classe XMLParser.

Après avoir parser le contenu du premier élément (`wording`) on attache le résultat à l'emplacement adapté. La méthode d'analyse de la chaîne XML utilise une approche récursive.

1. Pour chaque nœud rencontré,
2. On détermine si c'est un nœud de type élément, si c'est le cas alors
 1. On l'analyse avec une méthode générique // car on ne connaît pas encore sa nature.
 2. Puis on détecte la nature de la balise
 3. Pour ensuite appliquer une analyse sur les enfants,
 1. Chaque élément retourne une chaîne de type MathML

Dans notre interface nous avons souhaité que l'utilisateur puisse naviguer d'un exercice à l'autre. Pour se faire un utilise une méthode différente, appelé `loadPathNames(String path)` qui récupère la liste des fichiers XML situés dans le dossier `path`. C'est à partir que cette liste que seront construit les boutons de navigation, et ses élément servirons de paramètres à la méthode de chargement de problème appelé `loadProblem(String path)`.

Tutoriel

Étant donné que nous travaillons sur une application pédagogique, il nous paraissait très important d'intégrer un mécanisme d'auto-apprentissage : un tutoriel. Le fonctionnement du tutoriel est similaire à celui utilisé pour les exercices : on récupère des fichiers XML contenant énoncé et équation.

La différence se trouve d'abord dans les énoncés, qui expliquent les mécanismes de l'application, et surtout dans le contrôle qui est fait sur les possibilités de l'utilisateur (cf #4.o). En effet, selon le niveau du tutoriel certaines fonctionnalités de manipulation ne seront pas activées. Ceci est effectué par une approche en palier et une imbrication de branchement conditionnel. Le système sera certainement à revoir si l'on veut pouvoir contrôler de façon plus fine les manipulations disponibles.

BILANS

Retour sur les objectifs

Nous pouvons affirmer que nous avons rempli bon nombre des objectifs que nous nous étions fixés. Nous sommes fiers de proposer une application AniMaths fonctionnelle, fortement maintenable et aux possibilités d'évolution certaines. Sur les besoins exprimés initialement, il reste :

- A implémenter plus de possibilités d'interaction, en particulier le développement et la factorisation, après quoi on pourra vraisemblablement résoudre n'importe quel exercice qui peut être réduit à la forme $ax+b=0$...
- A afficher les `MathObjectProcessWidget` pour garder une trace plus forte des manipulations effectuées
- Toute la partie animations, que nous avons effleuré par des tests, mais que nous n'avons pas eu le temps de vraiment commencer
- A proposer un environnement d'édition d'objets `MathObjects` mathématiques pour stocker d'autres exercices sur le serveur
- A proposer une gestion d'une communauté d'étudiants et de professeurs.

Bien que notre projet soit terminé, nous constatons que l'application AniMaths est loin de l'être ! cependant, elle permet déjà de bien voir les possibilités et les attraits de ce genre d'approche. L'architecture que nous avons adoptée permettra une extension des possibilités de façon simple. Certains bouts de code ne sont pas optimaux et gagneraient à être retravaillés, nous savons exactement ce que nous changerions si nous avions plus de temps pour cela, mais nous pensons néanmoins avoir développé une base de départ solide pour des développements futurs.

D'un autre côté, nous avons rempli des objectifs que nous ne nous étions pas fixés initialement, mais qui -nous l'espérons- donneront envie à tous : Aux utilisateurs de l'utiliser, et aux développeurs passionnés comme nous, de continuer à le développer. x des avancées de notre application dans le laps de temps relativement court qu'était la 2ème phase, et nous pouvons dès à présent affirmer que nous serions très enthousiastes à l'idée qu'il soit poursuivi. Nous pensons également qu'une semaine supplémentaire ou deux suffiraient pour aboutir à une version utilisable en situation réelle

MathML, Firefox, GWT et le HTML

Nous ne réexposerons pas ici ce que nous avons décrit dans le paragraphe #5.4, mais pour résumer, les technologies que nous avons choisi d'utiliser - celles que nous avons eu besoin d'utiliser - sont très peu nombreuses, très récentes et pas très stables. Nous avons joué avec un cocktail explosif de nouvelles technologies, c'était risqué, mais nous pouvons affirmer aujourd'hui que le pari était tenable !

GWT

GWT est une technologie à la fois puissante et complexe. Elle s'appuie de façon poussée sur les concepts de la programmation orienté objet. Ce qui peut rendre l'utilisation de la technologie GWT complexe est qu'elle émule un sous-ensemble de bibliothèques Java, ce qui peut aboutir à des situations de blocage, car les ressources que l'on pensait avoir ne sont pas disponibles.

Un autre facteur de difficulté est la connexion et l'amalgame que GWT fait entre Java, Javascript, la programmation côté serveur et la programmation côté client. Malgré la complexité d'un concept, s'il existe des ressources pertinentes et une communauté pour introduire les débutants, le problème ne dure pas. Or la documentation GWT fournie par Google est souvent orientée vers des développeurs chevronnés. Quand à la communauté, elle est très limitée (20-30 personnes sur le salon #gwt) et très localisée en Amérique du Nord. Le décalage horaire rendant les contacts laborieux et éparpillés.

Eclipse

Le seul plugin officiel GWT est celui pour l'IDE Eclipse. Nous avons lors de la phase 1 testé l'IDE Netbeans, mais le plugin GWT4NB n'était pas suffisamment intégré pour un développement efficace. Nous avons également décidé d'utiliser Subclipse pour l'aspect collaboratif du projet, qui n'existe pas sous Netbeans.

Eclipse étant connu et reconnu, nous pensions être sûr d'un environnement stable et simple à utiliser. Ce projet nous a montré le contraire à maintes reprises. Tout d'abord Eclipse est une IDE professionnelle qui veut faire beaucoup de choses et sûrement trop, rendant l'application extrêmement complexe à maîtriser et à comprendre. Certes, elle apporte un nombre important d'outils d'aide au développement, mais là encore on se retrouve noyé sous les possibilités, les menus, les options. Tout ceci rajouté au fait que nous n'étions pas de grands experts de Java a plus apporté de confusion que d'aide.

Le projet avançant nous avons commencé à prendre nos marques, mais là encore des lacunes persistent. Et chaque jour apporte son lot de surprises : environnement qui refuse de se synchroniser, gestion étrange de certaines actions, etc.

Subclipse, le plugin SVN que nous avons utilisé à lui aussi était source de perplexité. Il est en effet possible d'écraser, effacer, ou corrompre des fichiers sans qu'aucun message ne prévienne l'utilisateur. Le SVN étant à la base assez fondé sur un paradigme orienté vers le serveur ou dépôt et non pas vers l'utilisateur, on se retrouve alors parfois à faire l'inverse de ce que l'on veut. On aurait pu croire qu'une IDE nous aurait facilité sur ce domaine, mais là encore les écueils sont nombreux, les ambiguïtés trop nombreuses et les problèmes trop réguliers.

Il faut tout de même reconnaître à Eclipse que c'est un outil puissant, mais pour lequel il est nécessaire d'avoir un apprentissage long, qui peut s'avérer pénible. De même Subclipse provoque quelques

angoisse, quand plus personnes n'a de version fonctionnelle, mais apporte une aide indispensable quand on travail en équipe.

UiBinder⁹

UiBinder est un framework de création d'interface utilisant une approche déclarative. En effet, la syntaxe utilisée par UiBinder est de type XML, ce qui permet à des gens sans connaissance de Java de créer les interfaces graphique. Nous avons essayé d'utiliser ce framework qui permet à la fois de gérer :

- le style de l'application ;
- l'internationalisation ;
- et pousser à l'extrême la séparation Vue-Presenter du modèle MVP

UiBinder ayant été intégré tout récemment dans GWT (présent depuis la version 2.0, actuellement en version 2.0.1), la documentation et les tutoriaux le concernant ne sont pas très nombreux. Les rares présent disponible et décrivant l'implémentation d'un tel framework sont peu explicites et difficiles à comprendre. De plus, il semble qu'il ne soit pas évident de l'utiliser avec l'injecteur de dépendance (GIN), que nous utilisons.

⁹ UiBinder devGuide : <http://code.google.com/webtoolkit/doc/latest/DevGuideUiBinder.html>

Maxime

D'un point de vue de la programmation, je suis heureux que nous ayons ainsi abouti à un programme fonctionnel, imposant avec ses 250 fichiers et 6500 lignes, Il y reste tant de bonnes choses à faire !

Peut-être vais-je me répéter, mais j'ai été enchanté de travailler sur ce sujet qui traite de l'enseignement, en particulier celui des matières scientifiques. Je pense qu'il me sera bientôt possible d'offrir cet outil à Cyril, élève à qui j'ai donné des cours personnels au premier semestre et que cette application aiderait probablement énormément ! Pour Cyril comme pour beaucoup d'autre, j'aimerais que "rigueur des mathématiques" rime avec casse-tête ; puzzle ; jeu plutôt qu'avec prise de tête, ou ennui !

Mon binôme est quelqu'un de très motivé, il a toujours de bonnes idées, pose les bonnes questions, et a une culture générale impressionnante quand aux technologies du web. Il m'a été d'une grande aide et il a su se faire violence pour se familiariser avec Java... je lui tire mon chapeau pour l'avoir aussi bien fait sur ce gros projet ! C'était vraiment un plaisir de travailler sur ce projet avec Edouard.

Édouard

Découvrant Java cet année, le projet AniMaths était mon premier projet d'envergure. Je ne maîtrisais pas très bien Java. La complexité de GWT, celle d'Eclipse ainsi que le manque de ressources disponibles sur le web ont un peu pénalisé ma participation. Ce n'est que dans les dernières semaines que j'ai commencé à me sentir à l'aise avec Java et Eclipse. Il me reste donc beaucoup à apprendre pour maîtriser Java, mais j'ai vraiment l'impression d'avoir beaucoup appris grâce à ce projet.

Sur le plan technique, le travail de collaboration intensif que nous avons eu m'a beaucoup apporté car j'ai découvert GWT et toutes les dépendances que cela implique (Java, RPC). Je suis un peu déçu de ne pas avoir réussi à implémenter UiBinder, cela m'aurait permis de travailler de façon poussée sur l'expérience utilisateur et l'ergonomie.

J'ai cependant approfondi ma connaissance de SVN, des systèmes de contrôle de version et des problèmes liés au travail collaboratif (surtout technique).

Travailler avec Maxime est toujours intéressant et enrichissant, il faut cependant s'accrocher car il a une grande capacité de travail et il est facile de ce laissé distancer. Il est à mon avis un excellent collaborateur, car motivé, ouvert, patient et appliqué.

Références

1. Communicate with a Server:
<http://code.google.com/webtoolkit/doc/latest/DevGuideServerCommunication.html#DevGuideHttpRequests>
2. GWT Tutorial – Building a GWT RPC Service:
<http://developerlife.com/tutorials/?p=125>
3. Making Remote Procedure Calls:
<http://code.google.com/webtoolkit/doc/latest/tutorial/RPC.html>
4. Tutoriel GWT :
Création d'un service distant: <http://haveacafe.wordpress.com/2008/10/11/tutoriel-gwt-creation-dun-service-distant/>
5. GWT : Créer un service RPC:
<http://www.mkhelif.fr/2008/07/07/gwt-crer-un-service-rpc.html>
6. Salon IRC:
 - o #java
 - o #gwt
7. YouTube - Graphviz dot http://www.youtube.com/watch?v=ZmxEt_OgVpg&feature=related
8. Trivial GWT Example <http://roberthanson.blogspot.com/2006/06/trivial-gwt-example.html>
9. Declarative Layout with UiBinder - Google Web Toolkit - Google Code
http://code.google.com/webtoolkit/doc/latest/DevGuideUiBinder.html#Hello_Widget_World
10. Developer's Guide - CSS Style - Google Web Toolkit - Google Code
<http://code.google.com/webtoolkit/doc/latest/DevGuideUiCss.html#cssfiles>
11. GWT 2.0 Declarative Interfaces | You don't know it all yet (Neither do I)
<http://eggyslife.co.uk/2009/11/01/gwt-2-0-declarative-interfaces/>