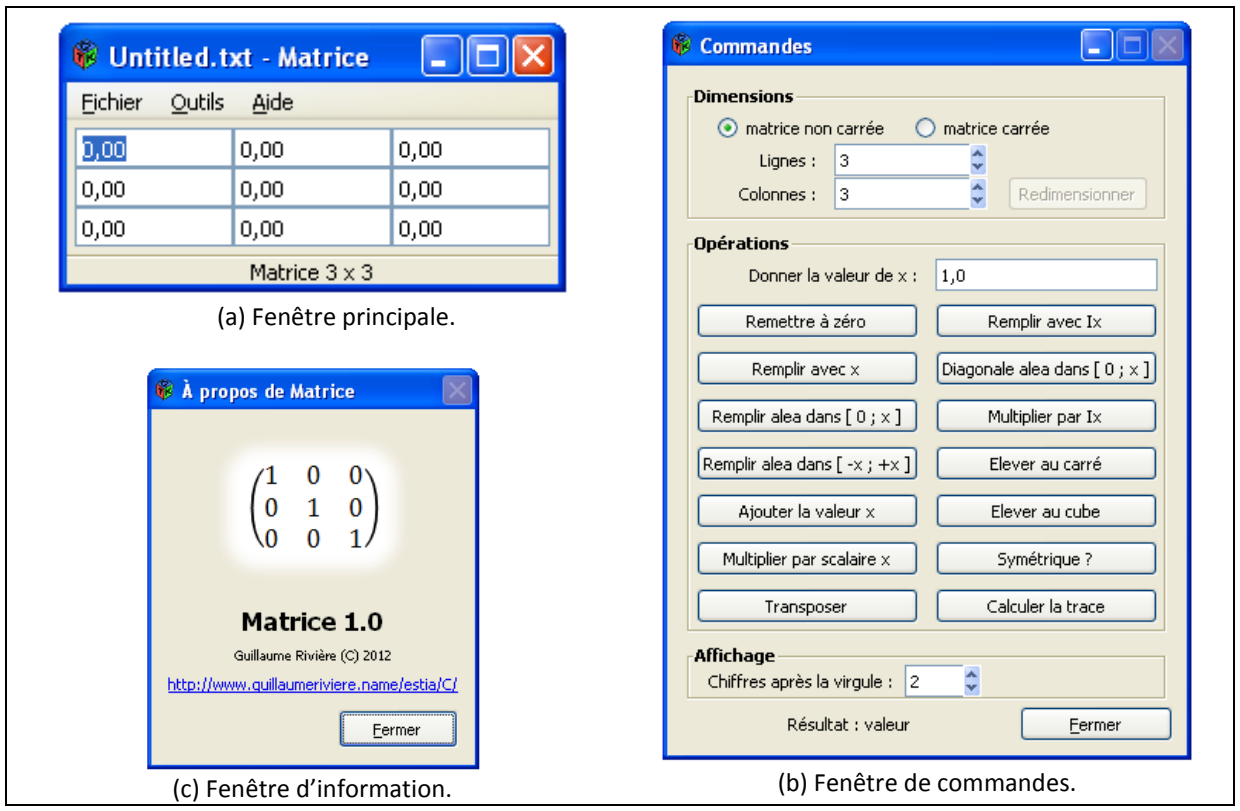
# L'APPLICATION UTILITAIRE « MATRICE »

Documentation programmeur de la Version 1.0

Guillaume Rivière - Mars 2012

## PRESENTATION GENERALE

Cette application utilitaire de manipulation d'une matrice permet d'afficher et modifier graphiquement les valeurs de la matrice. Les dimensions de la matrice (lignes et colonnes) peuvent être modifiées. Une douzaine d'opérations sont proposées via les boutons de l'interface. Certaines sont réservées aux matrices carrées. De manière non exhaustive, il est ainsi possible d'initialiser automatiquement la matrice avec des valeurs définies ou tirées au hasard, d'additionner ou multiplier les cases de la matrice, de transposer la matrice ou encore, pour les matrices carrées, d'élever la matrice au carré au cube, de calculer la trace et de vérifier si elle est symétrique. De plus, la matrice peut être enregistrée dans un fichier texte, pour être rechargée plus tard dans l'application, ou encore exportée au format CSV pour être importée dans un tableur.




(a) Fenêtre principale.

(c) Fenêtre d'information.

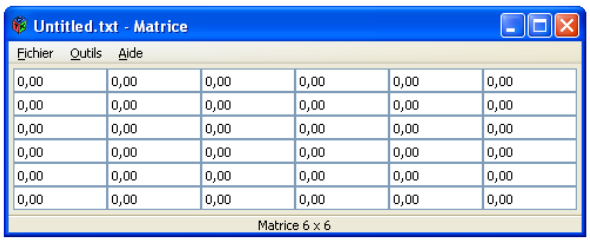
(b) Fenêtre de commandes.

**Figure 1 : Les trois fenêtres de l'application.**

Au lancement de l'application, la fenêtre principale de l'application est ouverte (voir figure 1-a) avec la matrice construite par défaut qui est une matrice 3x3 nulle (remplie de zéros). Depuis le menu « Outils > Commandes » l'utilisateur peut ensuite demander l'affichage de la fenêtre de commandes (voir figure 1-b). Cette fenêtre de commande est constituée de trois zones :

1. Une zone permettant de spécifier les nouvelles dimensions voulues pour la matrice.
2. Une zone permettant de réaliser des opérations sur la matrice.
3. Une zone permettant de spécifier le nombre de chiffres à afficher après la virgule (jusqu'à 9 chiffres).

Aussi, via le menu « Aide > À propos », une fenêtre d'information sur l'application peut être affichée. Parmi les fonctionnalités proposées par la fenêtre de commandes, se trouve le redimensionnement de la matrice. La figure 2-a présente la fenêtre principale après un redimensionnement de la matrice initiale en 6x6. Les dimensions de matrice possibles vont de 1x1 à 10x10 et ce pour des matrices carrées ou non. Parmi les opérations, également proposées dans la fenêtre de commandes, le remplissage aléatoire permet de donner des valeurs aux cases de la matrice dans l'intervalle indiqué par l'utilisateur (voir figure 2-b).




(a) Après redimensionnement de la matrice de départ en Matrice 6x6.


(b) Après une initialisation aléatoire de la matrice avec des valeurs entières comprises dans [-20 ; +20]

**Figure 2 : Redimensionnement et initialisation.**

## COMPATIBILITE ET DEPENDANCES

Cette application a été écrite en langage C sous Windows XP et 7. La version 2.24 de GTK+ a été utilisée pour l'interface graphique. Le générateur d'interface Glade 3.8 a servi à produire le fichier XML `matrice.glade` utilisé pour construire les éléments des fenêtres (avec la fonction `gtk_builder_add_from_file()`).

Le code source a été compilé avec le compilateur GCC v4.6.1 (provenant de MinGW 7.2) depuis l'interpréteur de commandes (la console) à la fois sous Windows 7 et XP (un fichier `Makefile` est disponible). Il a aussi été compilé et testé sous Visual Studio 2010 (Windows 7 et XP) et Eclipse Galileo (Windows 7). De plus, ce code a été compilé et testé sous GNU/Linux (Debian 6 « Squeeze »).

## DESCRIPTION DU CODE SOURCE

L'architecture de l'application est inspirée du modèle de Seeheim[1]. Ainsi, le code source est organisé selon trois modules principaux :

1. Présentation : tous les éléments concernant l'affichage et les saisie (en contact avec l'utilisateur).
2. Modèle : tous les éléments concernant le calcul et les structures de données (noyau applicatif).
3. Contrôle : Organise le dialogue entre les modules Présentation et Modèle (contrôleur de dialogue).

La bibliothèque graphique GTK+ utilisant un principe de fonctions « callbacks » pour signaler un évènement provenant d'une fenêtre ou d'un de ses widgets[2], un fichier entier (`callbacks.c`) est consacré à ces fonctions. Le fonctionnement des appels de fonction entre les modules est schématisé figure 3.

[1] Pfaff, G. E., editor (1985). User Interface Management Systems: Proceedings of the Workshop, Berlin. Springer-Verlag. proceedings of the Workshop on User Interface Management Systems, held in Seeheim, FRG, November 1-3, 1983.

[2] Widget : contraction de "Window Gadget", désigne un composant d'une interface graphique (bouton, zone de saisie de texte, fenêtre, …)

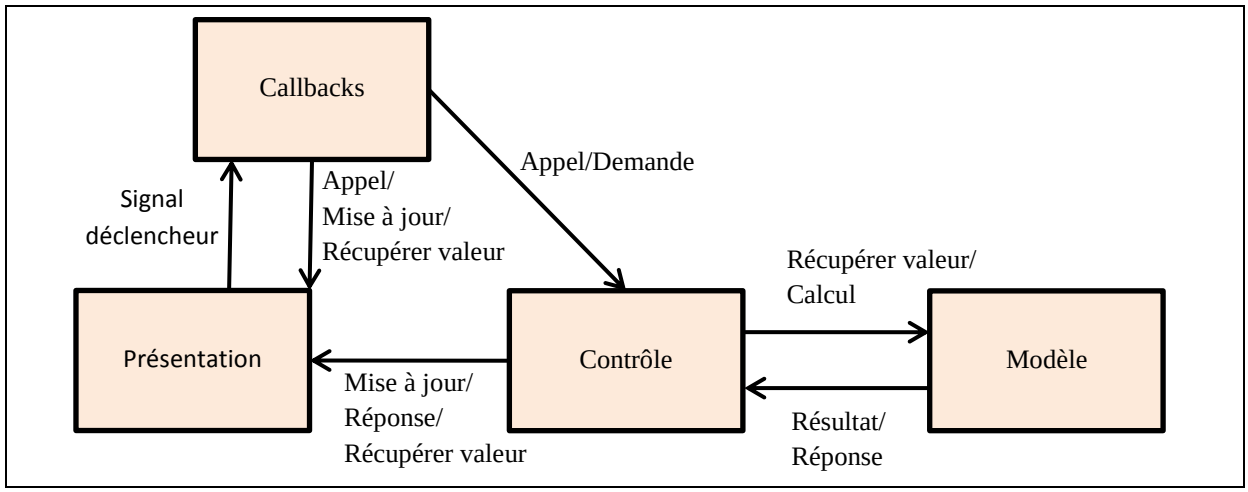
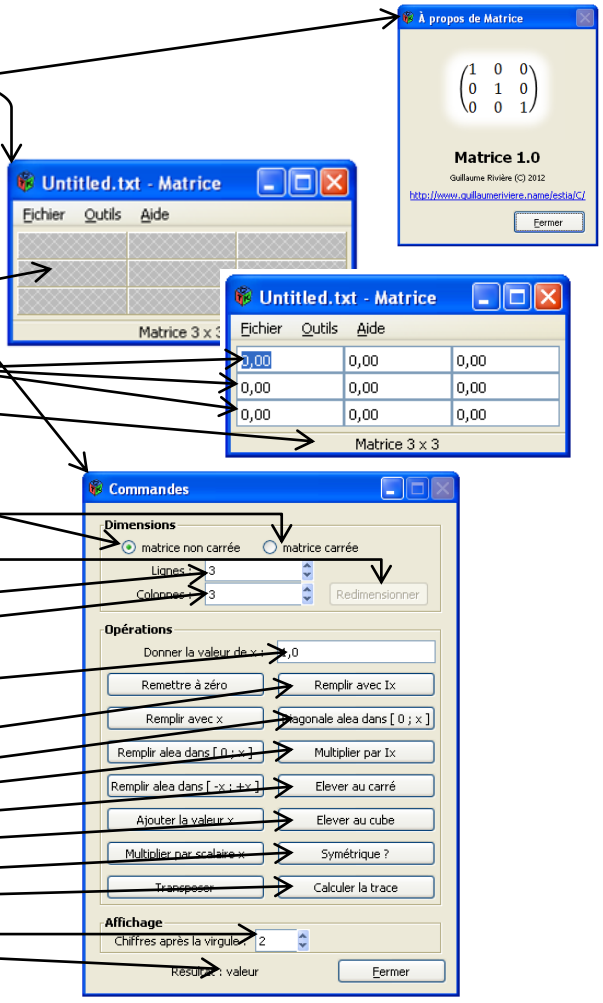



**Figure 3 : Architecture générale en modules de l'application matrice.**

## LE MODULE PRESENTATION

La fonction `presentation_create()` crée, initialise et retourne une structure `Presentation` qui est reliée aux widgets de l'interface graphique comme suit dans le script 1 :

```
typedef struct presentation {

  GtkWindow *mainwindow ;
  GtkWindow *commandwindow ;
  GtkAboutDialog *aboutdialog ;

  int N ; /* Nombre de lignes */
  int M ; /* Nombre de colonnes */

  char formatage[20] ;
  char application_titre[100] ;

  /*-- MAIN WINDOW widgets --*/

  GtkTable *table ;
  GtkEntry *entries[DIM_MAXI][DIM_MAXI] ;
  GtkLabel *label_message ;

  /*-- COMMAND WINDOW widgets --*/

  GtkRadioButton *radio_non_carree ;
  GtkRadioButton *radio_carree ;

  GtkButton *button_redimensionner ;

  GtkSpinButton *spin_lignes ;
  GtkSpinButton *spin_colonnes ;
  GtkSpinButton *spin_chiffres ;

  GtkEntry *entry_x ;

  GtkButton *button_fill_idx ;
  GtkButton *button_diag_alea_x ;
  GtkButton *button_mult_idx ;
  GtkButton *button_carre ;
  GtkButton *button_cube ;
  GtkButton *button_symetrique ;
  GtkButton *button_trace ;

  GtkLabel  *label_resultat ;

} *Presentation ;
```




**Script 1 : Widgets pointés par la structure Presentation après initialisation avec presentation_create().**

Dans l'interface GTK dessinée avec Glade dans `matrice.glade`, seule une grille d'agencement GtkTable a été insérée. C'est lors de la création de la structure `Presentation`, avec la fonction `presentation_create()`, que tous les GtkEntry nécessaires sont construits (en prévision de la dimension maximale 10x10) et attachés à la grille d'agencement. Ensuite, seulement ceux utilisés pour l'affichage de la matrice en cours sont visibles (les autres sont rendus invisibles) et la grille d'agencement est redimensionnée à la taille de la matrice.

Il est nécessaire de référencer, depuis la structure `Presentation`, les widgets `radio_non_carree`, `radio_carree`, `spin_lignes`, `spin_colonnes`, `entry_x` et `spin_colonnes` afin de pouvoir consulter leur état ou leur contenu. En ce qui concerne le bouton `button_redimenssionner` il doit pouvoir être grisé lorsque la matrice ne peut pas être redimensionnée (quand `spin_lignes` et `spin_colonnes` indiquent la même dimension que la matrice en cours), tout comme les boutons `button_fill_idx`, `button_diag_alea_x`, `button_mult_idx`, `button_carre`, `button_cube`, `button_symetrique` et `button_trace` qui sont réservés aux matrices carrées et doivent pouvoir être désactivés lorsque la matrice en cours est non carrée. Enfin, concernant l'étiquette `label_resultat`, il doit être possible d'y accéder pour modifier son contenu.

## LE MODULE MODELE

Le modèle permet de faire des calculs sur des matrices de différentes dimensions. Ce noyau applicatif supporte les opérations suivantes :

- Création d'une matrice de calcul ;
- Remplissage de la matrice de calcul (avec des zéros, avec une valeur définie ou une valeur aléatoire, ou encore avec la matrice identité) ;
- Redimensionnement de la matrice de calcul (sans perte de valeurs si possible) ;
- Élévation de la matrice au carré et au cube ;
- Addition d'une valeur, multiplication par un scalaire ou par la matrice identité ;
- Calcul de la transposée, de la trace et de la symétrie ;
- Enregistrement de la matrice de calcul dans un fichier, chargement depuis un fichier et exportation dans un fichier au format CSV.

La structure de données utilisée pour coder les opérations matricielles de base est implantée dans le module `matrice.h` et `matrice.c` *(rappel : c'était l'objet de l'exercice 4 du TD4)*.

## LE MODULE CONTROLE

Le module contrôle fait le lien entre la partie présentation et la partie modèle. C'est le contrôleur de dialogue. La seule fonction `controle_calculer()` du contrôleur de dialogue permet d'appeler les fonctions du modèle et de la présentation pour la réalisation des opérations suivantes :

1. OP_RESET . . . . . : réinitialiser les valeurs de la matrice,
2. OP_FILL_X . . . . . : remplir la matrice avec la valeur du champ x,
3. OP_FILL_ALEA_X . . : remplir la matrice avec des valeurs aléatoires dans [ 0 ; x ],
4. OP_FILL_ALEA_XX : remplir la matrice avec des valeurs aléatoires dans [ -x ; +x ],
5. OP_ADD_X . . . . . : ajouter la valeur x à la matrice,
6. OP_MULT_X . . . . . : multiplier la matrice par la valeur x,
7. OP_TRANS . . . . . : transposer la matrice,
8. OP_FILL_IDX . . . . . : affecter la matrice IDx,
9. OP_DIAG_ALEA_X : remplir la diagonale de la matrice avec des valeurs aléatoires dans [ 0 ; x ],

10. OP_MULT_IDX . . . . : multiplier la matrice par IDx,
11. OP_CARRE . . . . . : élever la matrice au carré,
12. OP_CUBE . . . . . : élever la matrice au cube,
13. OP_SYM . . . . . : vérifier si la matrice est symétrique,
14. OP_TRACE . . . . . : calculer la trace de la matrice.

Remarques :

- Les opérations 8 à 14 sont réservées aux matrices carrées.
- Les opérations 2, 3, 4, 5, 6, 8, 9 et 10 utilisent la valeur du champ x spécifiée par l'utilisateur.
- Les opérations 1, 2, 3, 4 et 8 n'utilisent pas en entrée les valeurs de la matrice en cours.
- Les opérations 5, 6, 7, 9, 10, 11, 12, 13 et 14 utilisent en entrée les valeurs de la matrice en cours.

## EXEMPLES DE TRANSACTIONS

Afin d'illustrer la répartition du travail entre les modules, deux diagrammes synthétisant les appels de fonctions pour deux opérations :

- Remplissage de la matrice avec la valeur x, dans le diagramme 1 (opération n'utilisant pas en entrée les valeurs de la matrice en cours).
- Élévation de la matrice au cube, dans le diagramme 2 (opération utilisant les valeurs en cours).
- Calcul de la trace (opération qui ne nécessite pas de mise à jour de la matrice après le calcul).

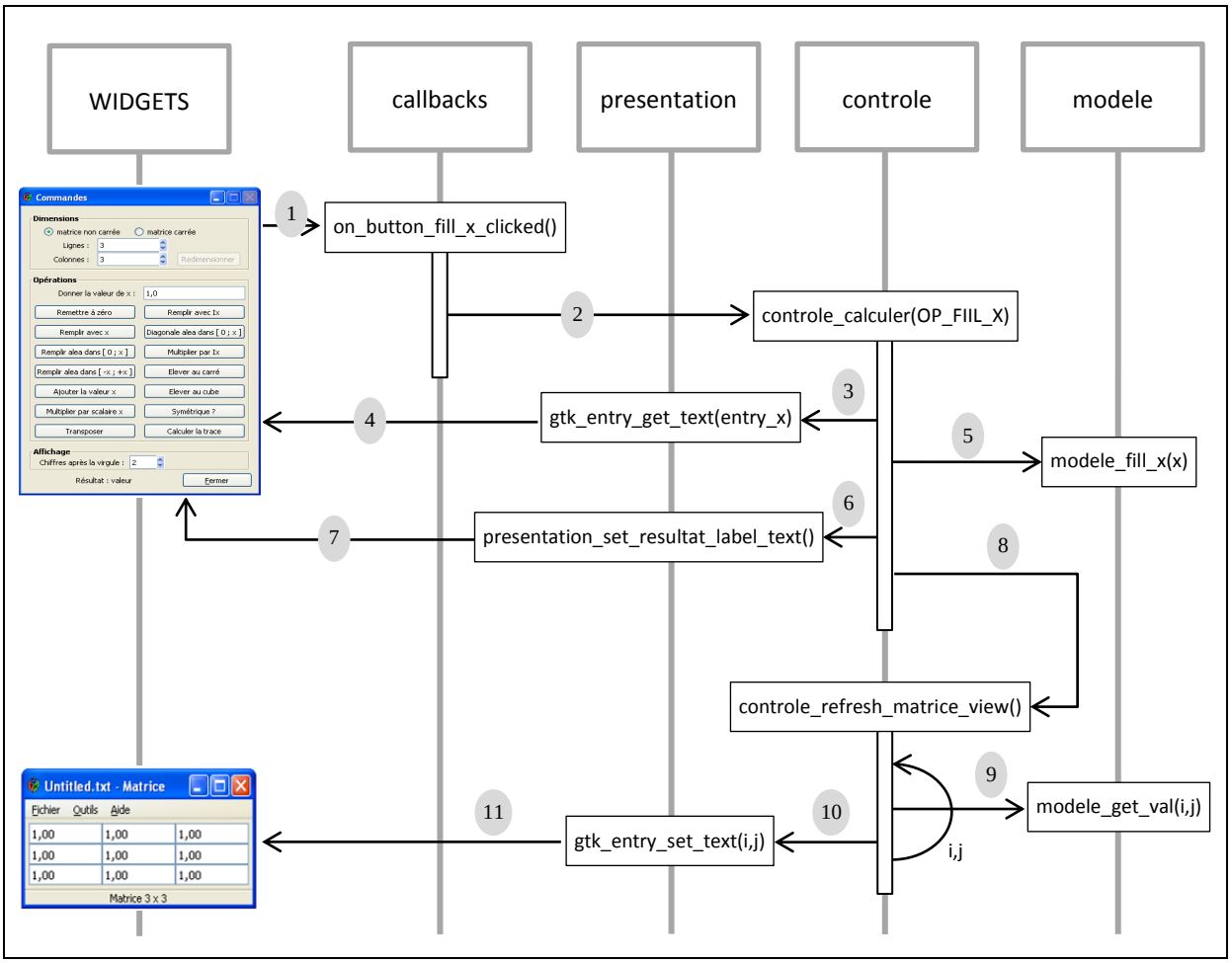



**Diagramme 1 : Séquencement des appels de fonctions pour l'opération de remplissage de la matrice avec la valeur x.**

Les 5 grandes phases décrites sur le diagramme 1 pour l'opération de remplissage sont :

- Déclenchement de l'action : **1** **2**
- Récupération de la valeur x : **3** **4**
- Modification sur le modèle : **5**
- Affichage du message de réussite : **6** **7**
- Mise à jour de la présentation à partir du modèle : **8** **9** **10** **11**

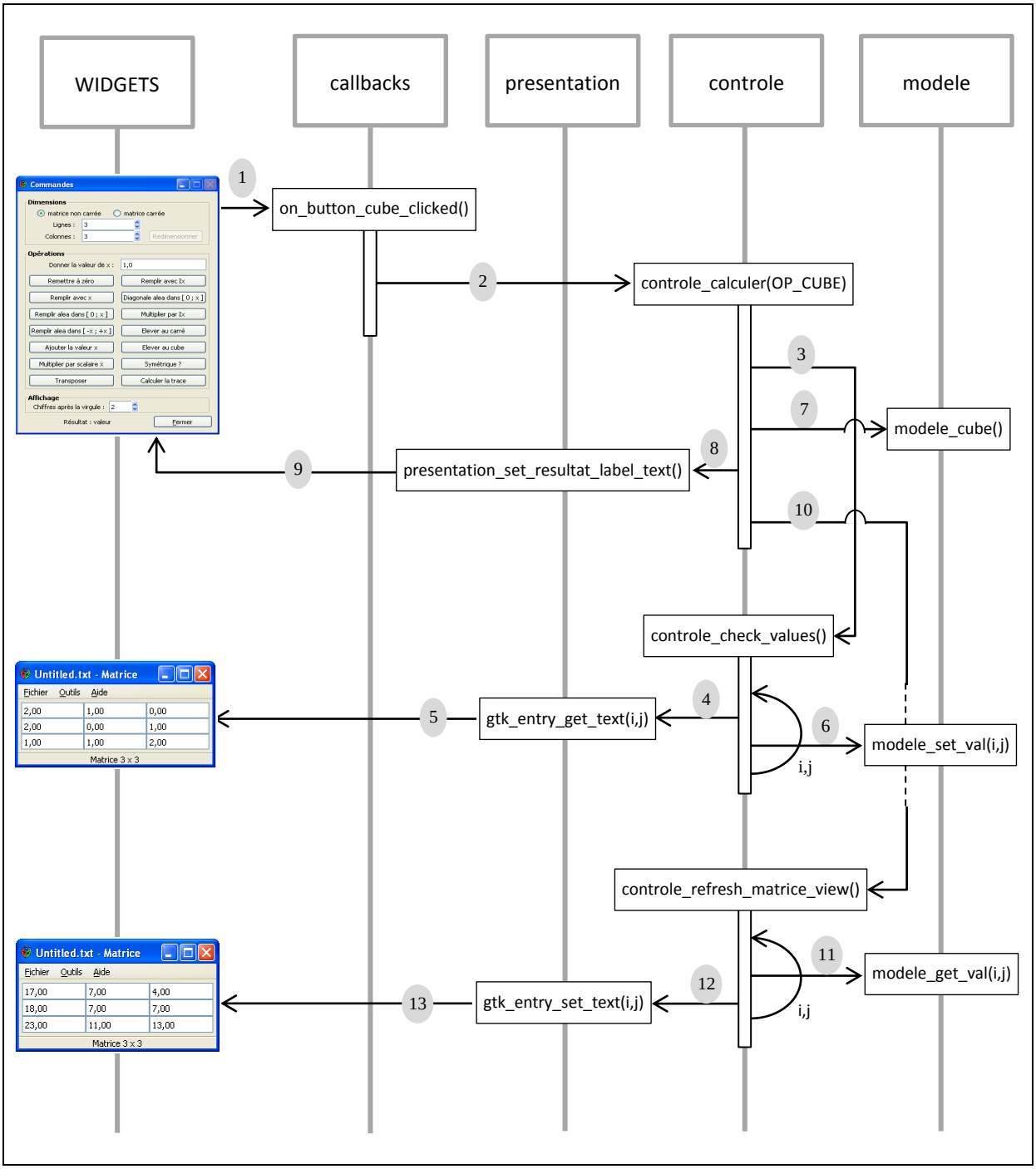



**Diagramme 2 : Séquencement des appels de fonctions pour l'opération d'élévation au cube de la matrice.**

Les 5 grandes phases décrites sur le diagramme 2 pour l'opération d'élévation au cube sont :

- Déclenchement de l'action : 1 2
- Mise à jour du modèle avec les valeurs de la présentation : 3 4 5 6
- Calcul et modification sur le modèle : 7
- Affichage du message de réussite : 8 9
- Mise à jour de la présentation à partir du modèle : 10 11 12 13

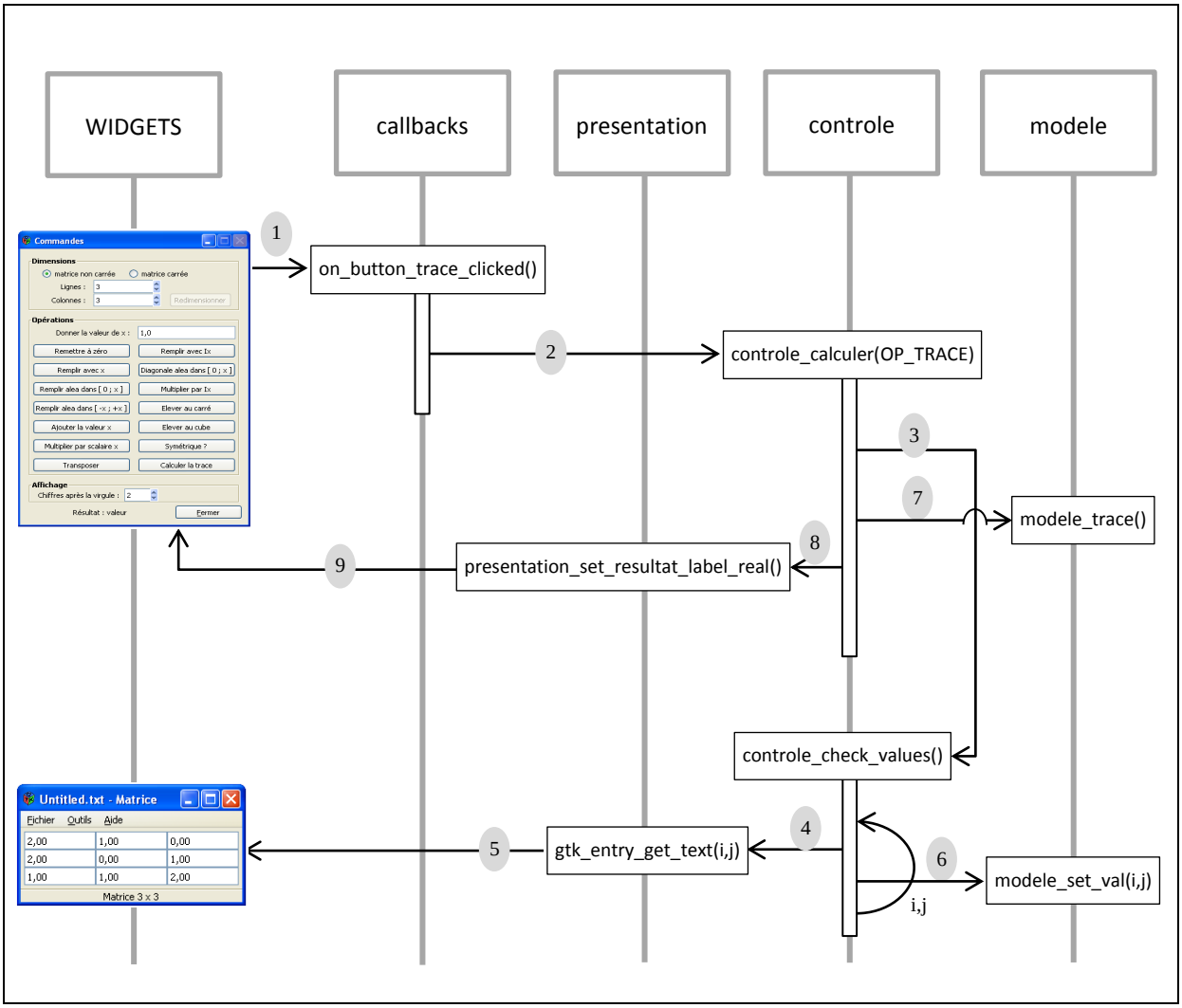



**Diagramme 3 : Séquencement des appels de fonctions pour l'opération de calcul de la trace.**

Les 4 grandes phases décrites sur le diagramme 3 pour l'opération de calcul de la trace sont :

- Déclenchement de l'action : 1 2
- Mise à jour du modèle avec les valeurs de la présentation : 3 4 5 6
- Calcul sur le modèle : 7
- Affichage du résultat : 8 9

## METRIQUES

Le nombre de lignes de code a été compté avec l'utilitaire cncc[3] (version 1.3.1). Cet analyseur de code permet notamment de calculer les métriques suivantes :

- SLOC (Source Lines Of Code) . . . . . : Lignes ne contenant que du code
- C&SLOC (Code and Comments Lines Of Code) : Lignes avec à la fois du code et des commentaires
- CLOC (Comment Lines Of Code) . . . . . : Lignes de commentaires seulement (sans code)
- BLOC (Blank Lines Of Code) . . . . . : Lignes vides
- LOC (Lines Of Code) . . . . . : Nombre total de lignes de code

Les mesures effectuées sur l'ensemble des fichiers du projet sont présentées dans le tableau 1.

| | SLOC | C&SLOC | CLOC | BLOC | LOC | SLOC | C&SLOC | CLOC | BLOC |
|---|---|---|---|---|---|---|---|---|---|
| Fichiers .c + .h | 1135 | 53 | 377 | 404 | **1969** | 58% | 3% | 19% | 21% |
| Fichiers .h | 184 | 14 | 95 | 88 | **381** | 48% | 4% | 25% | 23% |
| Fichiers .c | 951 | 39 | 282 | 316 | **1588** | 60% | 3% | 18% | 20% |
| | | | | | | | | | |
| presentation.h | 42 | 3 | 19 | 27 | **91** | 50% | 3% | 19% | 28% |
| callbacks.h | 30 | 1 | 6 | 14 | **51** | 59% | 2% | 12% | 28% |
| controle.h | 39 | 1 | 15 | 11 | **66** | 59% | 2% | 23% | 17% |
| modele.h | 30 | 2 | 15 | 10 | **57** | 53% | 4% | 26% | 18% |
| matrice.h | 22 | 4 | 7 | 5 | **38** | 58% | 11% | 18% | 13% |
| hasard.h | 4 | 1 | 8 | 4 | **17** | 24% | 6% | 47% | 24% |
| parametres.h | 13 | 1 | 15 | 7 | **36** | 36% | 3% | 42% | 19% |
| vs10settings.h | 4 | 1 | 10 | 10 | **25** | 16% | 4% | 40% | 40% |
| | | | | | | | | | |
| presentation.c | 145 | 3 | 57 | 57 | **262** | 55% | 1% | 22% | 22% |
| callbacks.c | 96 | 2 | 20 | 35 | **153** | 63% | 1% | 13% | 23% |
| controle.c | 304 | 28 | 99 | 89 | **520** | 59% | 5% | 19% | 17% |
| modele.c | 209 | 1 | 59 | 68 | **337** | 62% | 0% | 18% | 20% |
| matrice.c | 151 | 3 | 9 | 41 | **204** | 74% | 2% | 4% | 20% |
| hasard.c | 18 | 0 | 11 | 7 | **36** | 50% | 0% | 31% | 19% |
| main.c | 28 | 2 | 27 | 19 | **76** | 37% | 3% | 36% | 25% |

**Tableau 1 : Nombre de lignes de code du projet.**

## LIMITATIONS ET SUGGESTIONS

Sur cet exemple de code, l'encapsulation des données n'est pas complète. Un module ne devrait pas accéder à la structure de donnée d'un autre module, mais utiliser une fonction de ce module prévue à cet effet.

Suggestions de fonctionnalités supplémentaires pour parfaire cet utilitaire de matrice :

- Un bouton annuler pour rétablir les valeurs précédentes après une opération (voire gérer un historique, permettant d'annuler et de répéter) ;
- Initialiser une matrice symétrique de manière aléatoire ;
- Initialiser une matrice avec des valeurs réelles de manière aléatoire (par exemple, pour avoir une matrice avec des réels de 3 décimales compris entre 0 et 1 : la remplir avec des valeurs aléatoires entre 0 et 1000, puis diviser par 1000) ;
- Importer une matrice au format CSV (i.e. les dimensions ne sont pas connues avant la lecture) ;
- Calculer la matrice inverse, trianguläriser la matrice, etc.

[3] "Counter of C/C++ source lines and bytes" http://sourceforge.net/projects/cncc/