Compréhension de programmes: Analyse de l'architecture GIMP

Groupe 1

Vendredi 7 Décembre 2018

Table des matières

1 Introduction

Pour réaliser notre projet GIMP nous avons formé un groupe de 5 membres qui sont :
- PANCHALINGAMOORTHY Gajenthran
- LOKO Loïc
- MOTAMED SALEHI Amin
- BOUCHIHA Anas
- GHOUIBI Ghassen.

Ce devoir sur GIMP était réparti en 4 étapes : explications les mécanismes utilisés pour la création et l'utilisation du menu, explications des différentes étapes permettant la modification du menu, puis les explications des différentes étapes permettant l'ajout d'un élément au menu et enfin ressortir les fichiers concernés par la gestion du menu.
Nous tenons à préciser avant de passer aux sections suivantes que les commandes seront en gras (`commande`) et que le nom des fichiers et des dossiers sera en italique (*fichier*).

2 Avant l'analyse des mécanismes...

Nous nous baserons sur les éléments "Help" ou encore "Tip of the Day" ("View" également) du menu afin de mieux comprendre le fonctionnement et l'organisation du menu. Pourquoi prendre deux éléments du menu au lieu d'un ? Tout simplement, pour pouvoir traiter le cas qui va nous ouvrir une boîte de dialogue ("Tip of the Day") ou sinon le cas où l'élément effectue directement une action comme un zoom, une redirection vers un lien... ("Help").
Pour cela, la commande `grep` nous a été d'une grande utilité car elle nous a permis de détecter les différents fichiers qui contenaient le mot "Help" et "Tip of the Day" avec la commande `grep -R "help" .`. Evidemment, la commande nous retrouve un très grand nombre de résultat car le mot "Help" est un mot courant (utilisé dans la doc, dans le nom de fonctions. . .), du coup il est plus judicieux d'utiliser le groupe de mot "Tip of the Day" ou plutôt une partie du groupe. Cela nous ressort plusieurs dossiers mais beaucoup moins que celui de "Help" : *./po*, *./NEWS*, *./libgimpwidgets*, *./data* et surtout *./app*. Vu que notre ancien devoir consistait à décrire les dossiers, nous avons pas eu énormément de mal à repérer que le dossier *./app* et (plus précisément *./app/actions/help-actions.c* et *./app/dialogs/tips-dialog.c* pour nos exemples) était celui que nous devions traiter en premier car elle concerne

l'application des différentes fonctions déclarées pour la gestion du menu (pas seulement, mais ici c'est ce qu'il nous intéresse).

3 Organisation du code pour la gestion du menu

Une fois les fichiers trouvés, il suffit d'examiner les différentes inclusions (dans les deux sens) mettant en relation les fichiers *app/actions/help-actions.c* et *app/dialogs/tips-dialog.c* afin de mieux comprendre l'organisation du menu.

```
#include "config.h"

#include <gtk/gtk.h>

#include "libgimpwidgets/gimpwidgets.h"

#include "actions-types.h"

#include "widgets/gimpactiongroup.h"
#include "widgets/gimphelp-ids.h"

#include "help-actions.h"
#include "help-commands.h"

#include "gimp-intl.h"
```

FIGURE 1 – Inclusions du fichier app/actions/help-actions.c

```
#include "config.h"

#include <gtk/gtk.h>

#include "libgimpbase/gimpbase.h"
#include "libgimpwidgets/gimpwidgets.h"

#include "dialogs-types.h"

#include "config/gimpguiconfig.h"

#include "core/gimp.h"

#include "widgets/gimphelp-ids.h"

#include "tips-dialog.h"
#include "tips-parser.h"

#include "gimp-intl.h"
```

FIGURE 2 – Inclusions du fichier app/dialogs/tips-dialog.c

On comprend ainsi que les fichiers commençant par *[a-zA-Z]-actions.c* et *[a-zA-Z]-commands.c* font partis des éléments du menu (pas tous mais la majorité) dans le dossier *app/actions*. Puis nous avons dans le dossier *app/dialogs*, les fichiers *[a-zA-Z]-dialog.c* traitant les boîtes de dialogue. Nous retrouverons le contenu des fichiers *[a-zA-Z]-dialog.c* dans le fichier *dialog.c*

qui va s'occuper de la gestion de la boîte de dialogue (initialisation, chargement, libération...) notamment en incluant le fichier *dialogs-constructor.c* (qui, comme son nom l'indique, s'occupe de la construction de la boîte de dialogue).
Ainsi nous avons donc d'un côté les éléments du menu générant une boîte de dialogue qui vont être utilisés dans certains fichiers et qui vont également inclure certains fichiers spécifiques (comme *dialogs-types.h*) et d'un autre côté les fichiers n'ayant pas de boîte de dialogue. Les deux cas seront traités dans le fichier *actions.c*.
Désormais, essayons de voir les différents fichiers qui sont appelées par les éléments du menu à savoir les fichiers *[a-zA-Z]-actions* et *[a-zA-Z]-commands* ou encore *[a-zA-Z]-dialog*.

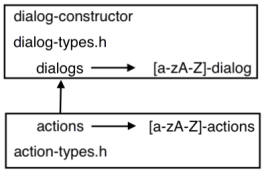

FIGURE 3 – Mécanisme des inclusions dans le dossier app concernant le menu

3.1 librairies

La librairie est un élément indispensable pour le logiciel Gimp, elle permet d'offrir les éléments « de base » pour les différentes fonctionnalités du logiciel. Il est donc évident qu'il soit utilisé dans les éléments du menu.
La plupart des éléments du menu utilisent la librairie qui permet de créer et utiliser des widgets et la majorité inclut le fichier *libgimpwidgets/gimpwidgets.h* car elle regroupe l'ensemble des fonctionnalités de base de la librairie widgets.
Un widget est une vignette interactive qui peut être considéré comme un mini-logiciel (source : Wikipedia). Les widgets vont nous permettre de construire des boîtes de dialogues. Il existe également des fichiers qui incluront d'autres librairies comme *libgimpmath/gimpmath.h* (des expressions mathématiques) ou encore *libgimpbase/gimpbase.h* (différentes fonctions essentiels concernant la gestion de la mémoire, des fichiers, des signaux, des vérifications...).

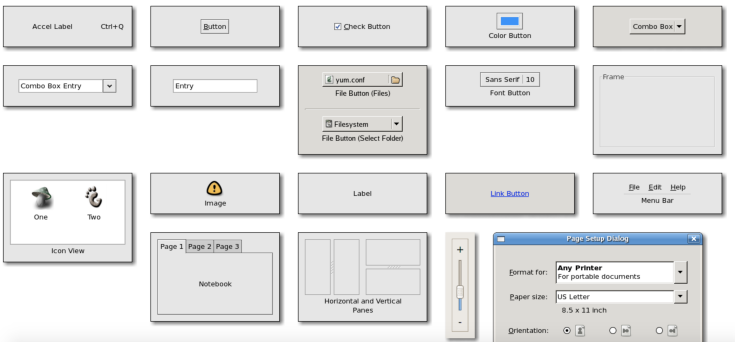

FIGURE 4 – Widgets de GIMP

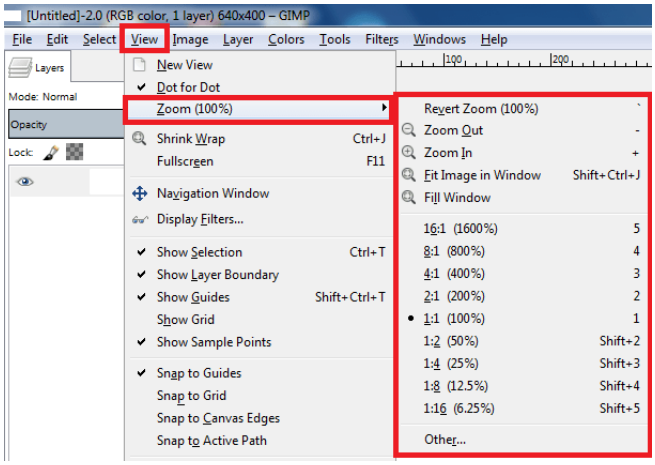

FIGURE 5 – Widgets de GIMP (2)

3.2 widgets

Plusieurs fichiers vont nous permettre l'implémentation de widgets avec chacun des fonctionnalités plus ou moins différentes. Le fichier *gimpactiongroup.h* (régulièrement appelé dans les fichiers que nous avions à traiter) génère des widgets qui vont nous permettre de créer des entrées de groupes d'actions (`GimpActionEntry`) qui correspondront aux différents éléments du menu. Il y plusieurs nuances d'entrées de groupes d'actions comme les `GimpRadioActionEntry` où l'on peut choisir seulement une des actions citées dans la structure. Par exemple, pour l'action "Zoom", nous pouvons seulement choisir qu'une seule valeur (nous ne pouvons pas avoir deux valeurs à la fois).
Puis ces entrées vont être regroupées dans des `GimpActionGroup`. Il y a également le fichier *app/widgets/gimphelp-ids.h* qui est souvent utilisé, tout simplement car il s'agit de l'identification de chaque élément du menu (donc

chaque élément du menu est unique sauf dans certains cas comme le « Zoom » que nous avons vu précédemment. Les différents éléments du "Zoom" partagent le même identifiant vu qu'il s'agit d'un radio (choix d'une seule valeur). Nous avons également d'autres fichiers utilisées dans le dossier *app/widgets* qui sont utilisées comme *app/widgets/gimprender.h* (l'affichage des éléments) mais que nous allons pas détailler dans le projet vu qu'ils nous paraissent moins importants.

3.3 fichiers récurrents

La grande majorité (si ce n'est l'intégralité) des fichiers traitant à propos du menu contiennent des fichiers de configurations (*app/actions/config.h* ou *app/config/gimpuiconfig.h*) indispensable pour le bon fonctionnement du logiciel mais également *gtk/gtk.h* qui veut dire "Gimp Toolkit" est également un point important de Gimp car il s'agit d'ensemble de fonctions qui permettent de réaliser des interfaces graphiques, *app/core/gimp.h* qui correspond au noyau du logiciel Gimp (en introduisant l'objet Gimp) ou encore le fichier *app/actions/gimp-intl.h*.

3.4 po

Les dossiers *po* regroupent l'ensemble des fichiers .po qui correspondent à la traduction des différents titres/éléments du menu (mais pas seulement). Chaque fichier est présenté de la forme suivante : la ligne du fichier sur lequel on effectue la traduction, le texte original et la traduction du texte (on peut se retrouver également avec le chemin sur lequel s'applique la modification).

3.5 app/menus et menus

Les fichiers *image-menu.xml* et *image-menu.xml.in* dans le dossier *menus* correspondant aux templates qui génèrent les éléments présents dans la barre de menus. La balise `<menu>` qui prend l'élément principale du menu et la/les balises `<menuitem>` de la balise `<menu>` regroupe(nt) l'ensemble des sous-éléments de la balise menu. Ainsi, cela permet de connaître l'organisation des différents éléments dans le menu. Par exemple, la balise `<separator>` permet de marquer une séparation à travers un trait horizontal dans la liste des sous-éléments d'un des éléments du menu. Ces fichiers seront traités dans le fichier *app/menus/menus.c*.

```
<menu action="help-menu" name="Help">
  <menuitem action="help-help" />
  <menuitem action="help-context-help" />
  <menuitem action="help-twitter"/>
  <menuitem action="dialogs-twitter"/>
  <menuitem action="dialogs-tips" />
  <menuitem action="dialogs-about" />
  <separator />
  <placeholder name="Programming" />
  <separator />
</menu>
```

FIGURE 6 – Fichier image-menu.xml

```
#: ../app/actions/actions.c:136
msgid "Dialogs"
msgstr "Boîtes de dialogue"
```

FIGURE 7 – Contenu d'un des fichiers du dossier po

3.6 autres fichiers

D'autres fichiers sont également utilisés par les éléments du menu :
- pour la gestion d'affichage de notre plan de travail (zoom), les sous éléments de "View" requièrent des expressions mathématiques (*libgimpmath/libgimpmath.h*) mais également des foncions concernant l'affichage (*app/display/gimpdisplay.h*, *app/display/gimpimagewindow.h* ...)
- pour "Tip of the Day", la boîte de dialogue concernée récupérera les données textuelles (les aides) dans le fichier *gimp-tips.xml* situé dans le répertoire *data*. Le fichier *gimp-tips.xml* possède plusieurs astuces dans différentes langues (`<thetip xml:lang="fr">`) à différents niveaux (`beginner`, `intermediaire`, `advanced`) dans le but d'aider le plus grand nombre de personnes (puis ce fichier sera parsé (découpé pour récupérer les informations importantes afin de pouvoir les exploiter) dans *tips-parsers.c* situé dans le même répertoire que *tips-dialog.c*)
- le fichier *about-dialog.c* a besoin d'informations sur le logiciel Gimp (version, auteurs, licence...), il va donc les récupérer dans d'autres fichiers tels que *git-version.h* (la version de git la plus récente), *about.h* (la structure des différents informations données, le chargement des logos) ou *authors.h* (les auteurs/contributeurs)

3.7 conclusion

Comme nous l'avons dit précédemment, de nombreux éléments du menu incluront souvent les mêmes fichiers comme les fichiers du dossier *widgets*,

po, *libgimpwidgets*, *gtk*, ou encore *config.h* et *gimp-intl.h* (voir section : les fichiers récurrents). On arrive donc à dessiner un chemin assez cohérent de l'organisation des fichiers/dossiers mettant en relation l'utilisation du menu. Mais les éléments se différencient par leur(s) fonctionnalité(s) ce qui engendrera l'inclusion d'autres fichiers.

4 Etapes permettant la modification du menu

Il existe plusieurs modifications possibles pour le menu : la modification du titre, la modification de l'emplacement ou encore la modification de la fonctionnalité du menu. Nous n'allons pas forcément modifier tous les fichiers/dossiers cités dans la section précédente qui constituent l'organisation du code pour la gestion du menu. Certains vont seulement être utilisés (comme les librairies) et d'autres vont être modifiés (comme *app/actions*).
Prenons et modifions les éléments "Help" et "Tip of the Day afin de mieux comprendre le mécanisme utilisé pour gérer le menu.
Comme nous l'avons vu précédemment, pour pouvoir modifier de façon "concrète" les éléments du menu, il faut se diriger vers le dossier *app/actions* et traiter les fichiers *[a-zA-Z]-actions.c* et *[a-zA-Z]-commands.c* OU les fichiers *dialogs-actions.c* et *dialogs.c*. Puis dans un second, répercuter les modifications dans les fichiers liées à ces derniers afin d'assurer le bon fonctionnement du logiciel et d'avoir un ensemble cohérent.

4.1 Structures des éléments du menu

Les éléments du menu correspondent à un ensemble de `GimpActionEntry` (ou de ses variantes : `GimpRadioActionEntry`, `GimpStringActionEntry`...) dont on avait vu dans le dossier *widgets*.

4.1.1 pour les boîtes de dialogue

Par exemple, pour "Tip of the Day", nous avons à utiliser un tableau de structure `GimpStringActionEntry` (légèrement différentes de `GimpActionEntry` dans le sens où elle ne possède pas de callback dans son champs) :
- le nom de l'élément (en rouge sur la figure suivante) qui commence par `dialogs-[nom]` où nom correspond au nom de l'élément
- l'id du stock (en vert) (l'ensemble des éléments qui forment un groupe traitant de la même idée : `GIMP_STOCK_INFO` qui gère les informations du logiciel)

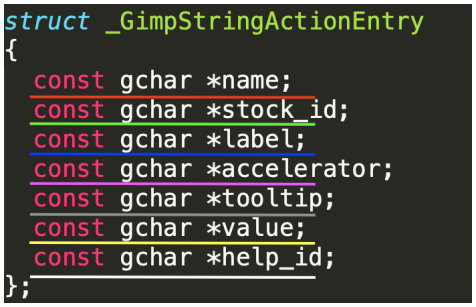

FIGURE 8 – Structure GimpStringActionEntry (élément d'un menu)

- le label de l'élément (en bleu) qui correspond au nom de l'élément affiché dans le menu du logiciel
- la description de l'élément (en gris) qui sera affichée lorsque notre curseur se trouve sur l'élément en question
- le raccourci (en violet) permettant d'accéder plus rapidement à l'élément. Si le champs vaut `NULL`, cela revient à dire qu'il n'y a pas de raccourci pour cet élément.
- valeur de l'élément du menu (en jaune) qui va ensuite être utilisé dans le fichier *dialogs.c* et qui va correspondre au fonctionnement de l'élément. En effet dans le fichier *dialogs.c*, chaque valeur de l'élément correspond est associé à une catégorie (`DOCKABLE`, `TOPLEVEL`) et à une fonction. - l'identifiant unique (en blanc), qui se trouve dans le fichier `gimphelp_ids.h`.

Puis dans le fichier *dialogs.c*, pour chaque valeur des éléments du menu, nous allons appeler une fonction afin de traiter les fonctionnalités des éléments.

4.1.2 pour les autres

Pour les autres éléments comme "Help", pas une grande différence comparé à la section précédente si ce n'est la structure qui est légèrement différente (`GimpActionEntry`) et l'ajout d'un champs `GCallback` :
- `GCallback` (en orange) qui va correspondre au fonctionnement de l'élément (fonction du menu appelé en boucle)

4.2 Modification du nom

Une fois la structure des éléments décortiquée, c'est beaucoup plus simple pour nous de réaliser les modifications. Pour modifier le nom, il suffit de modifier le champ `label` de la structure vu précédemment (dans les deux cas).

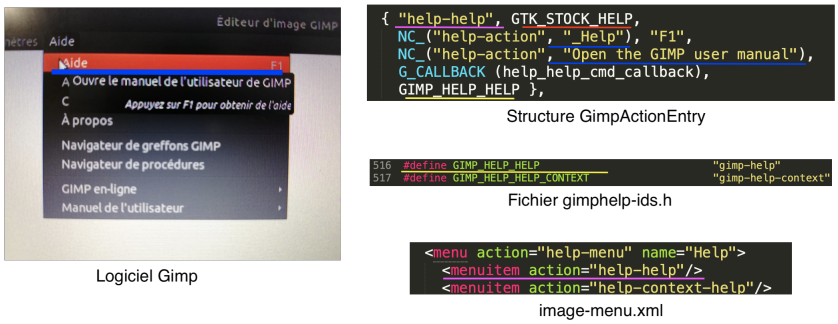

FIGURE 9 – GimpStringActionEntry et ses relations avec les autres fichiers

Nous ne traiterons pas dans ce devoir la traduction des noms du menu (dossier *po*), nous ajouterons probablement cela au rapport du devoir suivant.

4.3 Modification de l'emplacement

Pour modifier l'emplacement d'un élément, il faut modifier cette fois-ci le dossier *menus* et plus précisément les fichiers *image-menu.xml* et *image-menu.xml.in* que nous avons vu précédemment. Ces fichiers comportent le champ `name` de la structure de l'élément du menu.
Cependant, nous ne pouvons pas modifier n'importe comment, il faut que le changement d'emplacement soit cohérent. Nous ne pouvons pas nous permettre de déplacer une fonctionnalité traitant le zoom vers des éléments qui concernent la gestion des fichiers. C'est pour cela que le champ `stock_id` est nécessaire, il permet au développeur de mieux connaître les différents éléments qui gèrent un même ensemble.

4.4 Modification du comportement

C'est ici qu'on remarque la différence entre la structure des éléments utilisant des boîtes de dialogues et n'utilisant pas de boîtes de dialogues. En effet, comme nous l'avons évoqué dans les dernières sections, la structure `GimpStringActionEntry` et `GimpActionEntry` se différencent par leur champ `GCallback` qui est un champs appelant une fonction de manière répétitive (pour `GimpActionEntry`) et leur champ `value` qui renvoie une valeur (puis cette valeur sera utilisé dans le fichier *dialog.c* pour appeler une fonction).
Donc la modification du comportement de notre élément commencera dans le fichier *dialog.c* si il s'agit d'utiliser une boîte de dialogue et sinon il com-

mencera dans le fichier *[a-zA-Z]-actions.c* faisant référence à l'élément.

5 Etapes permettant l'ajout d'un élément au menu

Comme nous l'avons remarqué précédemment, pour pouvoir manipuler des éléments du menu, nous avons besoin de la structure `GimpActionEntry` ou de ses variantes. Donc pour créer un élément, il faut d'abord passer par la création d'une structure `GimpActionEntry`.

Certains champs de la structure devront respecter des conditions (citées dans les sections précédentes). Détaillons les champs de la structure `GimpActionEntry` :
- le champ `name` va nous permettre de savoir le nom de l'élément. Ce qui nous importe beaucoup, car il sera lié aux fichiers *image-menu.xml* et *image-menu.xml.in* du répertoire *menus*. Le contenu du champs est donc primordial car il nous donnera des indications sur la nature de l'élément (boîte de dialogue ou non) et sur le chemin de l'élément (si il ne s'agit pas de boîte de dialogue).
- le champ `stock_id` est également important car cela va nous permettre de classifier notre élément créé et de mieux se situer lorsqu'on souhaite le modifier. Nous pouvons également créer un nouveau `stock_id`. Pour cela, il faut ajouter un nouveau `stock_id` dans le fichier *libgimpwidgets/gimpstock.h*
- le champ `label` concerne l'affichage donc pas forcément primordial du point de vue du développeur car cela ne posera pas de soucis au niveau du code. Cependant, du point de vue de l'utilisateur, il est préférable de choisir un titre qui soit parlant, qui se rapprocherait du champ `name`.
- le champ `accelerator`, souvent facultatif, il n'est pas d'une grande importance pour le code, il peut même être ignoré. Mais si on souhaite accéder à une fonctionnalité plus rapidement sans passer par la sélection de l'élément, alors il peut s'avérer utile.
- le champ `tooltip`, concerne également l'affichage comme `label`. Il peut s'avérer nécessaire si on veut ajouter des précisions concernant notre élément.
- le champ gérant les fonctionnalités de notre élément qui se différencie selon la structure (voir section précédente). Si on souhaite ajouter un élément comportant une boite de dialogue, cela aura des impacts sur le fichier *dialogs.c* et le fichier en question (donc [a-zA-Z-dialog.c]) sinon seulement sur le fichier en question (donc [a-zA-Z]-actions.c).

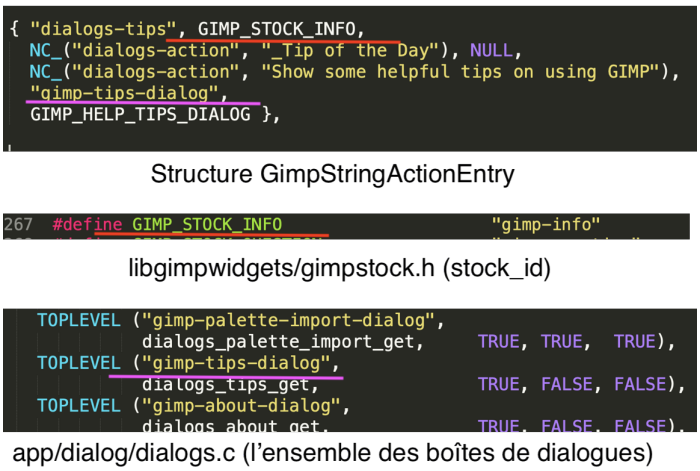

FIGURE 10 – GimpStringActionEntry et ses relations avec les autres fichiers

On remarque ainsi que seulement `name`, `stock_id` et le champ gérant la fonctionnalité vont avoir des répercussions sur les autres fichiers/dossiers. Nous apporterons plus de précisions lors du prochain rapport et des éventuels ajouts.

6 Arborescence : fichiers concernés par la gestion du menu

6.1 Arborescence à une grande échelle

6.2 Arborescence sur le dossier app