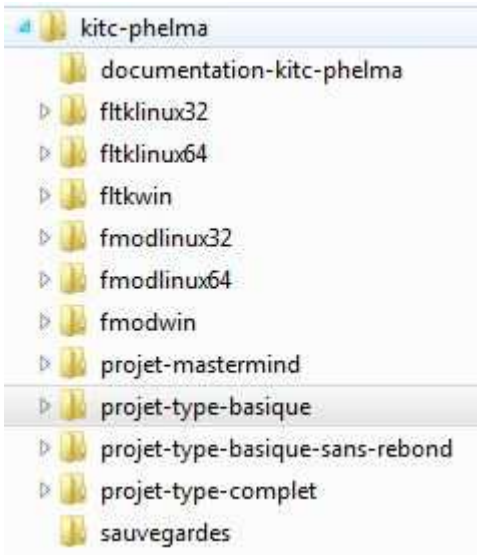
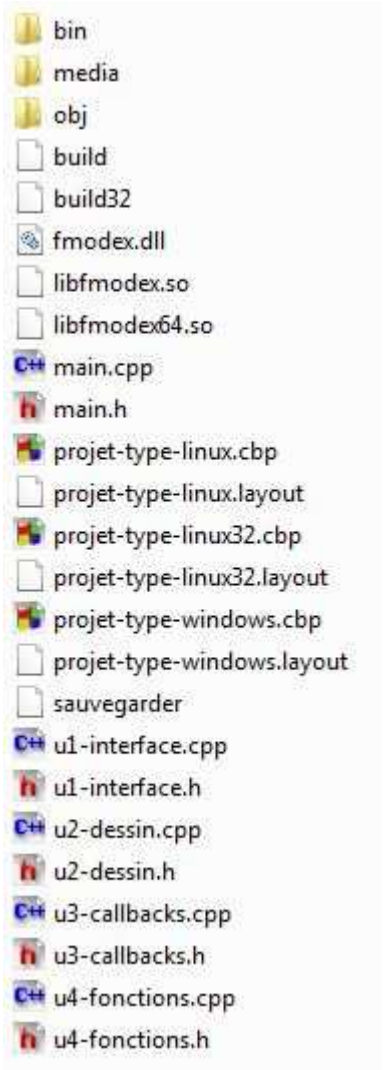
Kitc-Phelma

Objectifs :

- Comprendre la structure du kitc-phelma
- Créer un nouveau projet à l'intérieur du kit
- Utiliser le kit et les projets sous linux ou windows, sur un PC phelma ou un PC personnel



Projet-type-basique



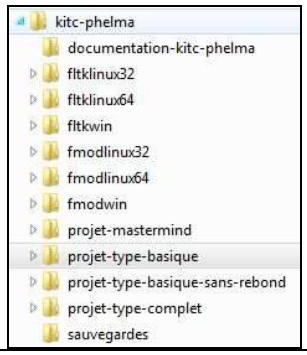
Icônes utilisées dans le document :

 concerne l'environnement système linux

 concerne l'environnement système windows

 concerne l'environnement de développement codeblocks

Ce que contient le kitc-phelma :



Répertoire : documentation-kitc-phelma

Ce répertoire contient les documentations suivantes :
- Aperçu rapide kitc-phelma et projet-type
- Documentation kitc-phelma (ce document)
- Documentation projet-type
- Documentation fltk (librairie graphique)

Répertoires : fltklinux64, fltklinux32 et fltkwin

Ces répertoires contiennent les versions linux 64 bits et 32 bits (fltklinux64, fltklinux32) et windows (fltkwin) de la librairie graphique fltk, version 1.1.10
A l'intérieur de ces répertoires, il y a 2 sous répertoires et 2 types de fichiers importants pour les projets :
- Sous répertoire ***FL*** : fichiers entête ***.h*** de la librairie
- Sous répertoire ***lib*** : fichiers objets ***.a*** de la librairie. Ces fichiers objets sont fournis dans le kit en version linux et windows. 6 librairies compilées et générées sont fournies : libfltk.a, libfltk_forms.a, libfltk_gl.a, libfltk_images.a, libfltk_jpeg.a, libfltk_png.a

Nota bene : sans ces répertoires, les projets ne pourront pas être générés car ils ont besoin de cette librairie fltk

Répertoires : fmodlinux64, fmodlinux32 et fmodwin

Ces répertoires contiennent les versions linux 64 bits et 32 bits (fmodlinux64, fmodlinux32) et windows (fmodwin) de la librairie son fmod version 4.2.
On y trouve les fichiers entête ***.h*** de la librairie, les fichiers objets **.a** et **.lib** et les fichiers dynamiques **.so** et **.dll** de la librairie.
Nota bene : sans ces répertoires, les projets utilisant le son ne pourront pas être générés.

Répertoires projets

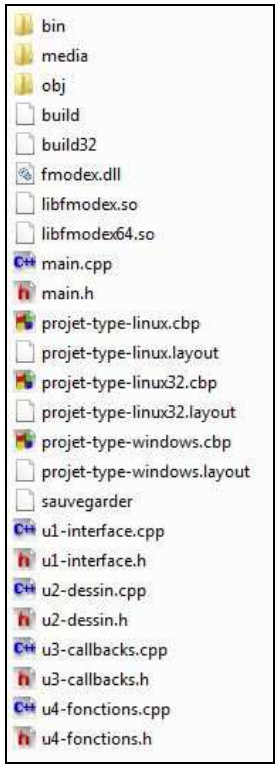
Qu'est ce qu'on appelle un projet ? C'est un répertoire qui contient les fichiers du programme réalisé (.cpp, .h) ainsi que les fichiers liés à l'environnement de développement (.cbp si codeblocks est utilisé), à la compilation et au link (.o) et à la génération finale de l'exécutable (.exe sous windows).
Plusieurs projets de démonstration sont fournis dans le kit. Parmi ces projets :
- ***projet-type-complet*** : démonstration d'un projet type (boule qui parcourt une zone de dessin) avec une interface complète présentant tous les objets disponibles. Cela servira de modèle pour créer des objets similaires dans vos propres projets
- ***projet-type-basique*** : la même démonstration d'un projet type avec une interface minimale. Il est conseillé d'utiliser ce projet comme base de départ pour vos propres projets
- projet-type-basique-sans-rebond : projet utilisable pour faire l'exercice d'ajout de la fonction « rebond » (case à cocher + callback et fonction associées)
- projet-mastermind : démonstration du projet appliqué au jeu mastermind. Un exemple concret et complet, intéressant à étudier.

Répertoire : sauvegardes

Sous linux, chaque projet est doté d'une fonction « sauvegarder » qui crée un fichier compressé du projet courant (exemple : projet-mastermind.03-04-2009-09-50.tar.bz). Ce fichier compressé de sauvegarde est stocké dans ce répertoire « sauvegardes ».

Ce que contient un répertoire projet

Tous les projets doivent avoir la même structure que le projet-type qui est fourni (basique ou complet)



Répertoire : bin

Ce répertoire contient les fichiers exécutables générés par la fonction « build » de l'environnement de développement codeblocks.
2 sous répertoires peuvent être présents et contenir les exécutables : debug et release.

Répertoire : media

Ce répertoire est destiné à stocker les fichiers utilisés par vos programmes : fichiers texte .txt de configuration, images .jpg , .bmp, .gif, sons .mp3, etc....

Répertoire : obj

Ce répertoire contient les fichiers binaires compilés .o depuis codeblocks. Il y a un fichier .o pour chaque fichier programme .cpp. La fonction « build » de l'environnement de développement codeblocks assemble ensuite ces fichiers objets pour produire l'exécutable final.

Commande : build, build32 

Si on préfère ne pas utiliser codeblocks, cette commande est utilisable sous linux pour compiler/linker votre projet. Sous un terminal, à l'intérieur du répertoire projet, taper : ./build
Cette commande est un utilitaire minimal permettant de générer votre exécutable. Il est bien sur possible d'aller plus loin et écrire un Makefile si on veut paramétrer plus finement la méthode de génération.

Nota bene : la commande "build" est à utiliser en environnement linux 64 bits, c'est le cas sur les PC de Phelma et la plupart des PC récents équipés de plus de 3 Go de mémoire. En environnement linux 32 bits, il faut utiliser la commande "build32"

Contenu détaillé de la commande build (64 bits) :
g++ -I../fmodlinux64 -I../fltklinux64 -I../fltklinux64/jpeg -o projet-type main.cpp u1-interface.cpp u2-dessin.cpp u3-callbacks.cpp u4-fonctions.cpp ../fltklinux64/lib/libfltk.a ../fltklinux64/lib/libfltk_forms.a ../fltklinux64/lib/libfltk_gl.a ../fltklinux64/lib/libfltk_images.a ../fltklinux64/lib/libfltk_jpeg.a ../fltklinux64/lib/libfltk_png.a ./libfmodex64.so -lX11 -lXext -ldl -lm

Fichiers : fmodex.dll, libfmodex64.so, libfmodex.so 

Librairie dynamique fmod version 4.2 pour le son. Elle est nécessaire en plus de l'exécutable généré si on veut utiliser le son :
- fmodex.dll est la version windows
- libfmodex64.so est la version linux 64 bits (à utiliser sur les PC Phelma)
- libfmodex.so est la version linux 32 bits (à utiliser sur les PC Phelma)

Nota Bene : Si on souhaite diffuser son projet, il faut placer cette librairie à côté de l'exécutable.

Fichiers .cbp : projet-type-linux.cbp, projet-type-linux32.cbp et projet-type-windows.cbp 

Il s'agit des fichiers de paramétrage codeblocks du projet courant. Ce paramétrage est très important :
- Ces fichiers contiennent aussi la liste des fichiers .cpp et .h qui constituent le projet lui même.
- Ces fichiers contiennent aussi les chemins pour trouver les fichiers entêtes (.h) et compilés (.a) des librairies graphique (fltk) et son (fmod).

Les fichiers de paramétrage fournis sont adaptés au kitc-phelma et à la localisation des librairies fltk et fmod selon chaque environnement système.
- **Sous linux 64 bits (c'est le cas des PC Phelma), il faut utiliser projet-type-linux.cbp**
- Sous linux 32 bits, il faut utiliser projet-type-linux32.cbp
- **Sous windows, il faut utiliser projet-type-windows.cbp**

Pour en savoir plus :

Détails de projet-type-linux.cbp (chemins utilisés pour la compilation et le link sous linux 64 bits) :

```
<Compiler>
        <Add option="`fltk-config --cxxflags`" />
        <Add directory="../fltklinux64" />
        <Add directory="../fmodlinux64" />
</Compiler>
<Linker>
        <Add option="-lX11 -lXext -ldl -lm" />
        <Add option="`fltk-config --ldstaticflags`" />
        <Add library="../fltklinux64/lib/libfltk.a" />
        <Add library="../fltklinux64/lib/libfltk_forms.a" />
        <Add library="../fltklinux64/lib/libfltk_gl.a" />
        <Add library="../fltklinux64/lib/libfltk_images.a" />
        <Add library="../fltklinux64/lib/libfltk_jpeg.a" />
        <Add library="../fltklinux64/lib/libfltk_png.a" />
        <Add library="./libfmodex64.so" />
</Linker>
```

Détails de projet-type-windows.cbp (chemins utilisés pour la compilation et le link) :

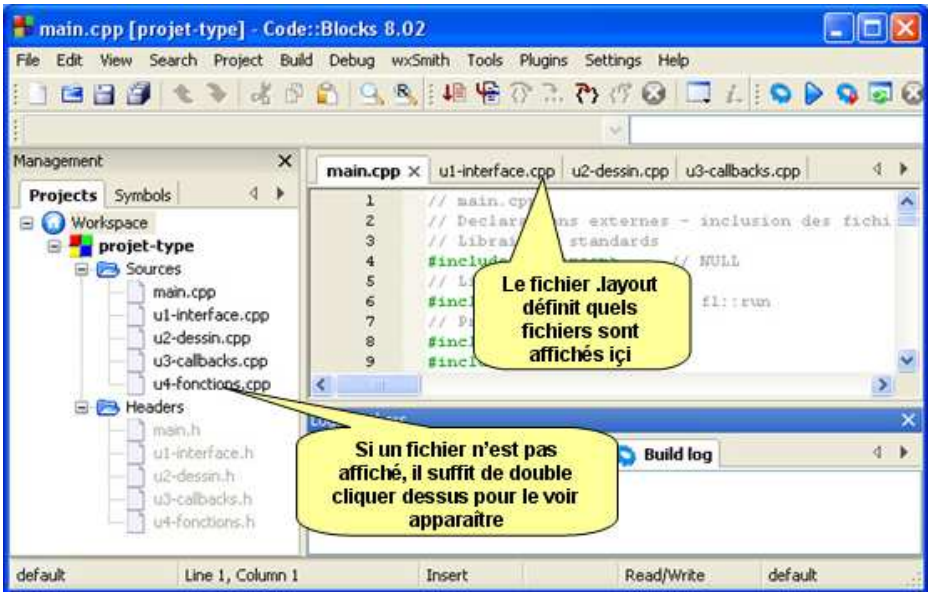
```
<Compiler>
        <Add option="`fltk-config --cxxflags`" />
        <Add directory="..\fltkwin" />
        <Add directory="..\fmodwin" />
</Compiler>
<Linker>
        <Add option="-lfltk -lmsvcrt -lgdi32 -lwsock32 -lole32 -luuid -lcomctl32 -lm -luser32 -lkernel32" />
        <Add option="`fltk-config --ldstaticflags`" />
        <Add library="..\fltkwin\lib\libfltk.a" />
        <Add library="..\fltkwin\lib\libfltkforms.a" />
        <Add library="..\fltkwin\lib\libfltkgl.a" />
        <Add library="..\fltkwin\lib\libfltkimages.a" />
        <Add library="..\fltkwin\lib\libfltkjpeg.a" />
        <Add library="..\fltkwin\lib\libfltkpng.a" />
        <Add library="..\fmodwin\api\lib\libfmod.a" />
        <Add directory="..\fltkwin\lib" />
        <Add directory="..\fmodwin\api\lib" />
</Linker>
```

Détails des projet-type-xxx.cbp (fichiers constituant le projet lui-même, c'est-à-dire vos programmes)

```
<Unit filename="main.cpp" />
<Unit filename="main.h" />
<Unit filename="u1-interface.cpp" />
<Unit filename="u1-interface.h" />
<Unit filename="u2-dessin.cpp" />
<Unit filename="u2-dessin.h" />
<Unit filename="u3-callbacks.cpp" />
<Unit filename="u3-callbacks.h" />
<Unit filename="u4-fonctions.cpp" />
<Unit filename="u4-fonctions.h" />
```

Fichiers .layout : projet-type-linux.layout et projet-type-windows.layout

Ces fichiers sont aussi liés au paramétrage du projet dans codeblocks. Ils définissent quels sont les fichiers du projet affichés et visibles par défaut quand on lance le projet sous codeblocks.



Comment utiliser le kitc-phelma sous linux ou windows ?

Les pré requis sont les suivants :

- Un environnement de programmation c/c++
- Sous linux : soit l'environnement gcc de base soit l'environnement de développement intégré (IDE) Codeblocks qui viendra en surcouche de gcc
- Sous windows : l'environnement de développement intégré (IDE) Codeblocks accompagné de mingw qui fournit le compilateur gcc
- Utilitaires de compression utiles : tar et gzip sous linux, **7zip sous windows**

Pour compiler/linker un projet :

- Sous linux solution 1 : en mode terminal, utiliser la commande **./build** (./build32 pour les linux 32 bits) à l'intérieur du projet concerné
- Sous linux solution 2 : lancer codeblocks, ouvrir le fichier **projet-type-linux.cbp** (projet-type-linux32.cbp pour les linux 32 bits) du projet concerné et utiliser la commande build&run
- Sous windows : lancer codeblocks, ouvrir le fichier projet-type-windows.cbp du projet concerné et utiliser la commande build&run

Pour créer votre programme :

- Editer les fichiers .cpp et .h du projet concerné : u1… u2… u3… u4…
- Sous linux : utiliser un éditeur de type gedit/nedit ou l'éditeur intégré de codeblocks
- Sous windows : utiliser l'éditeur intégré de codeblocks

Quel environnement système pour utiliser le kitc-phelma ?

Le kitc-phelma est optimisé pour fonctionner sous linux ou windows.
Quand vous êtes sur un PC de Phelma, il faut utiliser l'environnement linux où gcc et codeblocks sont installés et disponibles tous les 2.
Quand vous êtes sur votre PC personnel, vous pouvez utiliser linux ou windows (après avoir installé codeblocks-mingw).

Nota bene :

- Linux est « case sensitive » c'est-à-dire qu'il fait la différence entre minuscules et majuscules. Ce n'est pas le cas de windows.
- Sous linux par exemple, écrire #include <FL/Fl_BMP_Image.H> conduira à une erreur de compilation si le nom du fichier réel est fl_bmp_Image.h. Sous windows, cela ne posera aucun problème
- Recommandation 1 : même sous windows, écrivez les noms des fichiers librairies .h et .a tels qu'ils sont écrits dans l'environnement linux en respectant bien minuscules et majuscules
- Recommandation 2 : **pour les fichiers que vous créez vous-mêmes, prenez par exemple la convention simple de mettre les noms de fichiers toujours tout en minuscules**.

J'ai un ordinateur sous mac OS, comment je fais ?

Les librairies fltk et fmod ainsi que gcc et codeblocks sont disponibles sous mac OS. Le projet-type peut être porté sous mac OS. Mais nous n'avons pas encore intégré dans le kitc-phelma les librairies précompilées fltk, fmod pour mac OS.

Solution de secours : créer une machine virtuelle et installer windows ou linux dans cette machine. Installer ensuite codeblocks et le kitc-phelma.

Comment créer mon propre projet ?

Recommandation forte : à l'intérieur du répertoire kitc-phelma, il faut dupliquer projet-type-basique et renommer le duplicata en projet-xxx.

- **Il est extrêmement important que projet-xxx reste dans le répertoire kitc-phelma** pour pouvoir utiliser les librairies fltk, fmod fournies lors de la génération de votre programme.
- A l'intérieur du projet-xxx, il suffit alors d'éditer les fichiers .cpp et .h pour écrire votre propre programme en respectant l'architecture du projet-type fourni.
- Pour ajouter de nouveaux objets, vous pouvez vous prendre modèle sur les exemples supplémentaires du projet-type-complet. Mais il est recommandé de ne pas partir d'un duplicata du projet-type-complet pour créer votre nouveau projet (trop d'objets, variables et fonctions inutiles seraient alors à supprimer).

Comment sauvegarder mon projet ?

Recommandation forte :

- **Sous linux à Phelma, utiliser à chaque séance de travail la commande ./sauvegarder** dans le répertoire de votre projet. Cela génère un fichier sauvegarde compressé et daté dans le répertoire kitc-phelma/sauvegardes.
- **Sous Windows : créer une sauvegarde compressée (zip, tar.gz) de votre répertoire de projet à chaque séance de travail**.

Comment installer codeblocks sur mon PC personnel ?

Sous Windows :

- Télécharger la dernière version de codeblocks sur le site http://www.codeblocks.org. Attention de bien prendre la version incluant le compilateur C mingw.
- Exemple au 01/03/2012 : télécharger codeblocks-10.05mingw-setup.exe
- L'installation est automatique.

Sous linux :

- Prérequis : libX11-dev, libXext-dev, gtk2-devel et wxWidget 2.8.x
- Installer libX11-dev et libXext-dev par package :
  - Sous centos : yum install libX11-dev libXext-dev
  - Sous ubuntu : apt-get install libX11-dev libXext-dev
- Installer Gtk2 par package :
  - Sous centos : yum install gtk2-devel
  - Sous ubuntu : apt-get install gtk2-devel
- Installer wxWidgets-2.8.12 (version stable) en partant des sources
  - Télécharger wxWidgets-2.8.12.tar.gz (http://sourceforge.net/projects/wxwindows/files/2.8.12/wxWidgets-2.8.12.tar.gz/download, site officiel : http://www.wxwidgets.org/downloads/)
  - tar xvf wxWidgets-2.8.12.tar.gz
  - cd <répertoire wxWidgets...>
  - ./configure
  - make
  - make install
- Installer codeblocks en partant des sources
  - Pour prendre en compte wx_widget au make install :
    - export LD_LIBRARY_PATH=/usr/local/lib:${LD_LIBRARY_PATH}
    - export LD_LIBRARY_PATH=/usr/lib64:${LD_LIBRARY_PATH}
  - Télécharger codeblocks-10.05-src.tar.bz2 (ou les sources de la dernière version stable) (http://prdownload.berlios.de/codeblocks/codeblocks-10.05-src.tar.bz2 , site officiel : http://www.codeblocks.org/downloads)
  - tar xvf codeblocks-10.05-src.tar.bz2
  - cd <répertoire codeblocks...>
  - ./configure
  - make
  - make install

Pour aller plus loin sous linux :

Les librairies fltk et fmod sont fournies dans le kitc-phelma. Néanmoins, vous pouvez aussi en faire une installation complète et "propre" sous votre linux.

- **Fltk** : partir des sources de la **version 1.1.10** (c'est celle utilisée dans les programmes du kitc-phelma) sur le site http://www.fltk.org/software.php et installer via les commandes ./configure, make, make install.
- **Fmod** : partir des sources de la **version 4.32.08** (Nota Bene : ne pas utiliser les versions supérieures avec lesquelles des problèmes ont été constatés) sur le site http://www.fmod.org/index.php/download/find et installer via les commandes ./configure, make, make install.

Comment transférer mon projet de mon volume personnel Phelma à mon PC personnel et vice versa ?

Il faut avoir le kitc-phelma installé sur son PC personnel Linux ou Windows (ou sur une machine virtuelle sous mac OS).

Solution 1 : on transfère le projet dans sa totalité

- De Phelma à son PC personnnel :
  - Utiliser la commande ./sauvegarder du répertoire projet et transférer sur votre PC via clé usb ou sftp le fichier ainsi sauvegardé dans kitc-phelma/sauvegardes.
  - Décompresser le fichier sauvegarde à l'intérieur du kitc-phelma de votre PC
- De son PC personnel aux PC Phelma :
  - Créer un fichier compressé du répertoire du projet (commande ./sauvegarder si cous êtes sous linux ou via 7zip si vous êtes sous windows)
  - Transférer le fichier sauvegarde compressé sur votre volume personnel Phelma via clé usb ou sftp.
  - Décompresser le fichier sauvegarde à l'intérieur du kitc-phelma de votre volume personnel Phelma

Solution 2 : on ne transfère que les fichiers programmes du projet (les autres fichiers de paramétrage linux/windows/codeblocks sont toujours les mêmes quel que soit le projet)

- Liste des fichiers programmes à transférer :
  - main.cpp
  - main.h
  - u1-interface.cpp
  - u1-interface.h
  - u2-dessin.cpp
  - u2-dessin.h
  - u3-callbacks.cpp
  - u3-callbacks.h
  - u4-fonctions.cpp
  - u4-fonctions.h
  - répertoire media si utilisé
- On recopie ces fichiers programmes dans un duplicata de projet-type-basique que ce soit sur le PC personnel ou le volume personnel Phelma

Les addons du kitc-phelma (kitc-phelma-addons.zip)

Les addons contiennent :

- Les sources de fltk pour éventuellement générer des fichiers de librairies supplémentaires (par import de projet vc2005 sous windows ou par make sous linux)
- Les sources de fmod (linux et windows)
- Les sources des documentations fournies aux formats doc et odt, ppt et odp
- Le projet : projet-mastermind-correction qui n'est pas fourni dans le kitc-phelma-initial
- 7zipxxx.exe : installeur 7zip windows pour dézipper les fichiers .tar.gz issus de l'environnement linux

FAQ : les problèmes connus et leurs solutions

1) Recommandation installation codeblocks :
Sous windows, choisir la version codeblocks intégrant mingw (codeblocks-10.04mingw-setup.exe). Choisir l'option « Installation complète ».

2) Problème : La compilation sous linux génère plein d'erreurs (ex : fonctions fltk non trouvées) alors qu'elle fonctionne bien sous windows
Solution : Bien vérifier les minuscules et majuscules dans les #include des noms des fichiers .h d'entêtes des librairies (Rappel : linux est « case sensitive », windows non).

3) Problème : Sous codeblocks windows (certaines versions), build&run fait correctement le "build" mais "plante" lors du "run"
Solution : Renommer c:\program files\codeblocks\cb_console_runner.exe en cb_console_runner-ori.exe

4) Problème : sous linux, le message suivant apparaît : "error while loading shared libraries: ./libfmodex.so: cannot restore segment prot after reloc: Permission denied"
Solution : en mode terminal, tapez la commande suivante : /usr/sbin/setenforce 0

5) Problème : je lance ./build ou build&run mais il y a plein de messages d'erreur lors du link liés à l'absence des librairies fltk/fmodxxx
Solutions :

- a) Vérifier que le projet est bien à l'intérieur du kitc-phelma
- b) Vérifier que le bon fichier .cbp est utilisé si vous êtes sous codeblocks (projet-type-windows.cbp ou projet-type-linux.cbp ou projet-type-linux32.cbp selon l'environnement système)
- c) Recopier les fichiers suivants dans votre répertoire de projet à partir du projet-type-basique initial :
    - projet-type-linux.cbp
    - projet-type-linux.layout
    - projet-type-linux32.cbp
    - projet-type-linux32.layout
    - projet-type-windows.cbp
    - projet-type-windows.layout
    - build
    - build32
    - sauvegarder

6) Problème : sous linux, les commandes build, build32 et sauvegarder ne fonctionnent pas (pas de droits d'exécution)
Solution : en mode terminal, tapez la commande : chmod u+x build sauvegarder

Liens

Codeblocks : http://www.codeblocks.org/
Librairie fltk : http://www.fltk.org/
Librairie fmod : http://www.fmod.org/
Mingw (windows) : http://www.mingw.org/
Machine virtuelle : http://www.virtualbox.org/