Ouvrir la fenêtre de travail

Version 1.0

***

1. Configurer le projet

2. Code minimal et explication

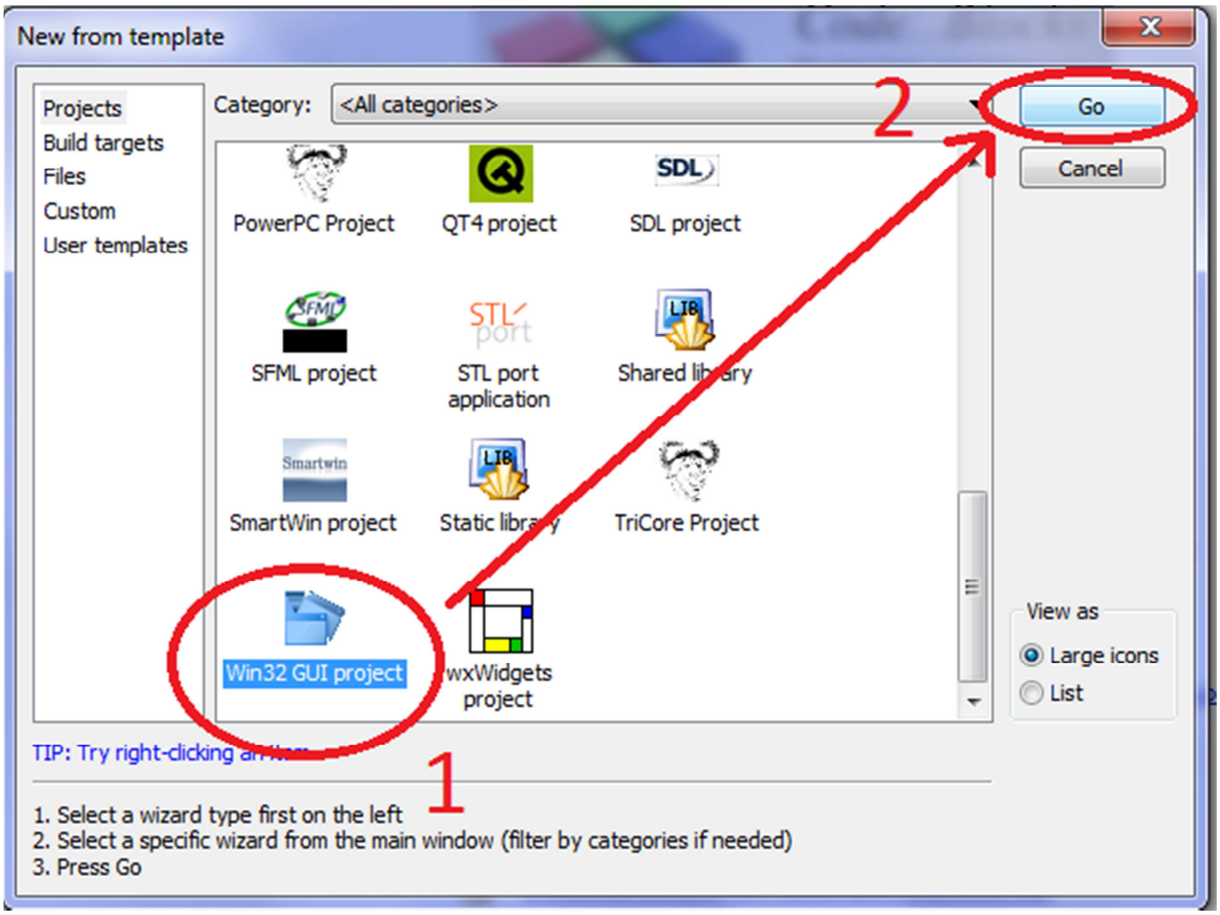
3. Descriptifs des structures et fonctions

***

Configurer le projet

L'une des premières difficultés que vous allez rencontrer est de configurer le projet. En effet, vous avez téléchargé des .h ou des .cpp sur le serveur, mais vous allez en faire quoi ?

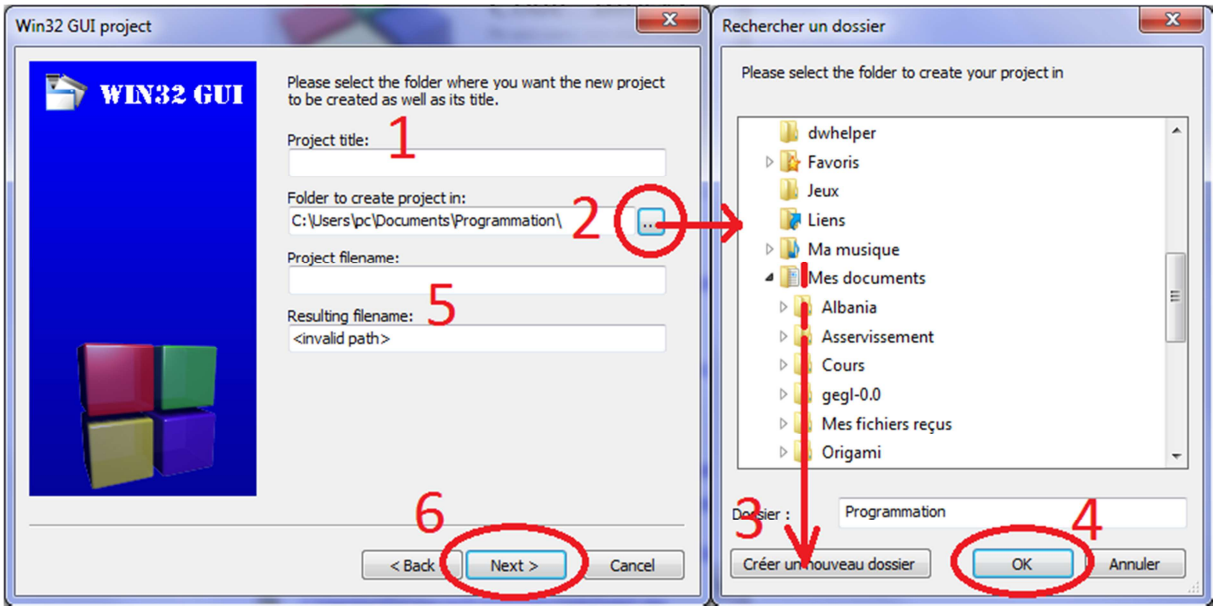
Tout d'abord, ouvrez Code::Blocks. Fermez toutes les fenêtres qu'il vous ouvre puis créez un nouveau projet (File > New > Project …). Une fenêtre va apparaître :



Choisissez Win32 GUI project (1) et faites Go (2).

Sélectionnez ensuite « Frame Based » puis faites Next.

Vous arrivez alors à la fenêtre suivante :



D'abord, choisissez un nom de projet (« projet_de_groupe » convient) (1). Ensuite, faites un choix de chemin pour le projet (là où seront entreposés les fichiers nécessaires à la création du programme). Pour cela, cliquez sur les « … » (2). Créez un nouveau dossier si vous le voulez (3) mais en principe, Code::Blocks va créer un dossier pour vous. Ne le mettez pas dans un endroit « exotique » (exemple de chemin exotique : C:\Program Files\...). Faites OK (4). En principe, les 2 derniers champs se remplissent automatiquement. Faites enfin Next (6).

Faites Finish à la dernière fenêtre.

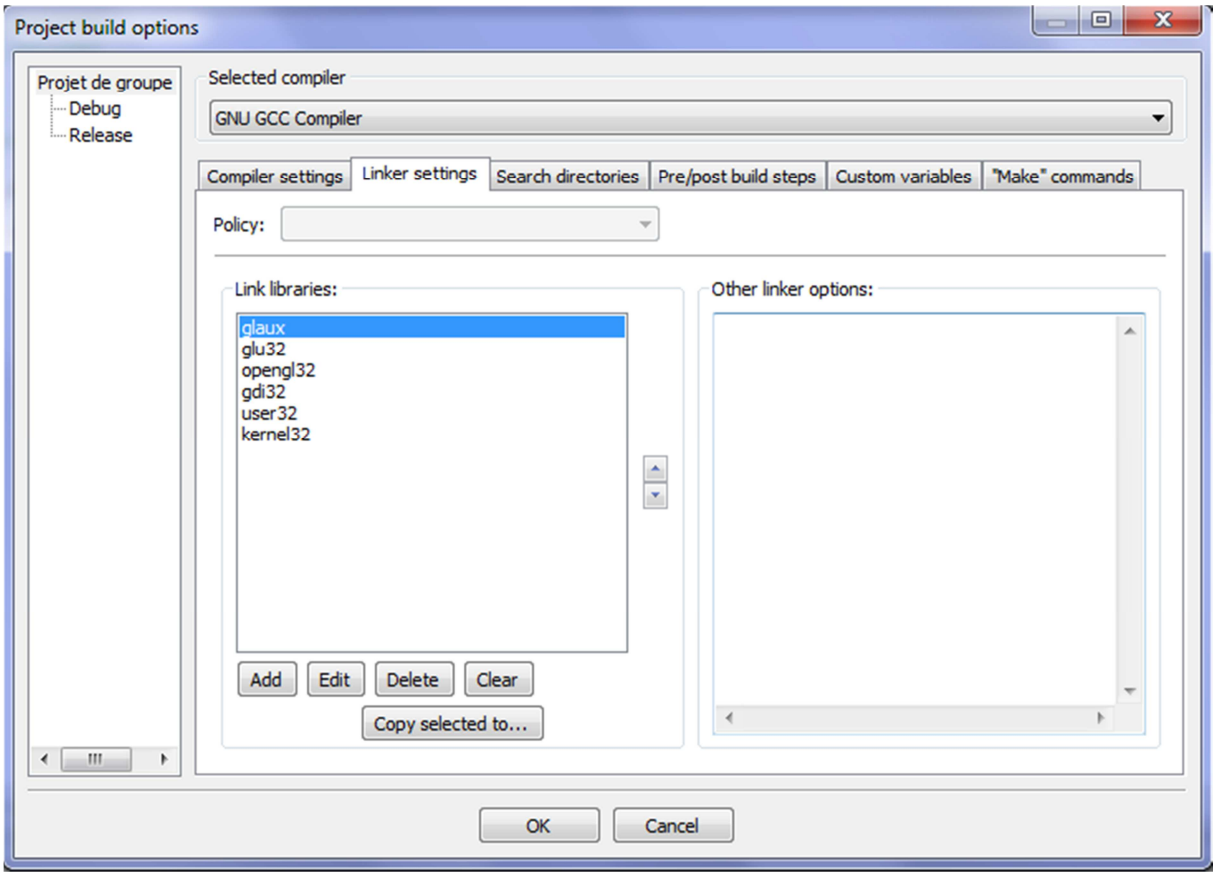
Tadaam ! Et là … On voit rien -__-' .

En fait, des fichiers sources ont été créés à votre insu. Sur la gauche, déroulez le [+] sous le mot « Sources ». Cliquez sur main.cpp et admirez l'horreur. Essayez de le compiler (F9 ou Build > Build and Run). Chouette ! Ça marche !

Bon bon bon, … c'est bien beau, mais on ne veut pas ça. Prenez les fichiers .h et .cpp du serveur (sauf ceux dans le dossier glaux) et collez les à l'emplacement que vous avez spécifié précédemment (en principe, vous remplacez main.cpp par la même occasion). Revenez dans Code::Blocks. En principe, il devrait râler un peu. Répondez Yes et normalement le main.cpp change en un truc beaucoup plus beau.

Il faut alors ajouter les autres fichiers à notre projet. Faites un clic droit sur le nom de votre projet (en gras dans le panneau à gauche). Sélectionnez « Add Files… ». Là, choisissez tous les fichiers que vous avez précédemment collés (Ctrl + Clic pour ajouter un fichier à la sélection). Faites Ok. Dans la fenêtre qui s'ouvre, cochez Debug et Release. Le panneau de gauche vient de se remplir.

Dernière étape de ce parcours du combattant, ajouter les bibliothèques. Allez dans Project > Build Option. Dans le panneau de gauche, prenez garde de bien prendre le nom du projet puis allez dans l'onglet « Linker Settings ». Faites Add puis écrivez : « opengl32 ». Refaites Add puis écrivez : « glu32 ». Enfin, Add puis « glaux ». Vous devriez alors avoir la vision suivante (l'ordre compte) :



Problème : en principe, vous n'avez pas glaux. (glu32 et opengl32 sont en principe par défaut déjà dans Code::Blocks). Il faut alors l'ajouter et là c'est encore la galère. Déjà, vous trouverez les fichiers nécessaires dans un dossier du serveur « glaux ». Placez le fichier libglaux.a dans « …\CodeBlocks\MinGW\lib » (cet emplacement se trouve là où vous avez installé Code::Blocks, « C:\Programm Files » le plus souvent). Placez le fichier glaux.h dans « …\CodeBlocks\MinGW\include\GL ».

Voilà ! C'est fini, normalement, le projet est configuré. Essayez de compiler et regardez si ça marche.

Code minimal et explications

A ce stade, vous avez normalement la fenêtre qui s'affiche. Une explication assez succincte du code minimal s'impose.

Pour ouvrir une fenêtre, il faut faire un include de « fenetre.h ». Dedans est contenu tout ce qu'il faut pour ouvrir la fenêtre.

Vous aurez ensuite remarqué que le main est assez bizarre, d'ailleurs, il s'appelle WinMain. Ça, c'est pour que l'application soit Windows. Les paramètres que prend le main, on s'en tape, si ce n'est qu'on en aura besoin pour ouvrir la fenêtre après.

Pour ouvrir la fenêtre, vous aurez besoin de 2 variables : Fenetre et Contexte_GL.

La fonction init_fenetre va alors mettre les paramètres et ouvrir la fenêtre.

La fonction afficher_fenetre va l'afficher et update_fenetre va la mettre à jour.

On entre alors dans la boucle principale. La fonction recuperer_evenement va récupérer les évènements. La boucle s'arrête quand recuperer_evenement renvoie le signal d'extinction de la fenêtre.

Dans la boucle, la fonction traiter_evenement va faire le lien entre les évènements et les dispatcher dans la zone qu'il faut. C'est la fonction CallBack qui est dans evenement.cpp qui contient ce que doit faire la fenêtre en réponse à ces évènements.

C'est à cet endroit que l'on dessine dans la fenêtre OpenGL.

Enfin, on met à jour la fenêtre OpenGL.

La fonction end_fenetre va libérer la mémoire allouée lors de init_fenetre.

Descriptif des structures et fonctions