(1)
Installer GL4D

Tout est là https://gl4d.api8.fr/FR/
(On se donne une heure ???)

 —

Extrait du manuel d'installation disponible en ligne

Université Paris 8 - Vincennes à Saint-Denis

GL4D

Annexe : Installation

Farès Belhadj

Date de MAJ : 6 octobre 2021

email : mailto:amsi@up8.edu
github : https://github.com/noalien/GL4Dummies
web [GL420] : https://gl4d.api8.fr

Extrait du manuel d'installation disponible en ligne

Annexe A

Installation de la Bibliothèque *GL4Dummies*

GL4Dummies étant une bibliothèque multiplateformes et *Open Source*, son code est disponible en téléchargement [1] et il est possible de la recompiler spécifiquement pour votre système d'exploitation.

La première section A.1 traite de l'installation multi-OS de *GL4Dummies* à partir de son code source.

La seconde section A.2, spécifique aux utilisateurs de *Windows 10*, détaille l'utilisation de *GL4Dummies*, avec l'IDE *Visual Studio Community 2019*, sans recours à la recompilation de la bibliothèque.

A.1 À partir du code source

Cette section décrit l'installation de la bibliothèque *GL4Dummies* à partir de son code source. Dans chaque sous-partie vous trouverez les indications à suivre en fonction de votre système d'exploitation : Linux / Unix-like / Mac OS X et Windows.

A.1.1 Prérequis : la bibliothèque *SDL2*

SDL2 (`http://www.libsdl.org/`) est une bibliothèque de développement multi-plateformes permettant un accès *hardware* aux ressources graphiques présentes sur le système. Elle permet aussi l'accès bas niveau aux ressources

1. Vous pouvez télécharger *GL4Dummies* à partir de l'adresse : `https://github.com/noalien/GL4Dummies`

Extrait du manuel d'installation disponible en ligne

d'interface (clavier, souris et joystick). Aussi, des extensions de la bibliothèque donnent un accès simple aux ressources audio, à la gestion des *pthread*, à l'utilisation de *fonts Truetype*, à l'ouverture et l'enregistrement de multiples formats d'images, etc.

GL4Dummies dépend de la bibliothèque *SDL2* principalement pour ce qui concerne la gestion de fenêtres et du contexte OpenGL® (pour faire court : zone de l'écran dans laquelle OpenGL® effectue ses rendus). Il est nécessaire d'avoir la bibliothèque de développement (soit les paquets contenant les *headers* et les bibliothèques statiques) de *SDL2* afin de pouvoir recompiler *GL4Dummies*. Aussi, pour quelques exemples proposés avec la bibliothèque, il sera nécessaire d'avoir certaines extensions de la bibliothèque, telles que : *SDL_images, SDL_mixer* et *SDL_ttf*. Faire attention à récupérer les versions pour *SDL2* et non *SDL1.2* (l'ancienne version de la bibliothèque ne permet pas de gérer un contexte OpenGL® *3 et plus*).

Afin de vérifier que *SDL2* est bien installée, et pour faire simple, nous pouvons considérer que la bibliothèque de développement est présente sur le système si la commande `sdl2-config` est présente et répond en donnant les bons chemins vers les différents fichiers (voir `sdl2-config --help`). Sinon, en fonction de l'OS (système d'exploitation) faire :

Pour les utilisateurs Linux, utiliser le gestionnaire de paquets afin d'installer la bibliothèque de développement (lib dev) de *SDL2*. Par exemple un utilisateur Ubuntu fera :
```
sudo apt install libsdl2-dev
```

Pour les utilisateurs Mac OS X, commencer par installer les *MacPorts*[2] (`http://www.macports.org`). Une fois les *MacPorts* pleinement fonctionnels, installer *SDL2* en tapant (dans un terminal) :
```
sudo port install libsdl2
```

Pour les utilisateurs Windows, pas besoin d'installer *SDL2* tout est dans l'archive *GL4Dummies* sauf si vous souhaitez utiliser une version plus récente de *SDL2*.

Enfin, il est aussi possible d'installer *SDL2* à partir de son source, néanmoins, cette procédure n'est pas recommandée car certaines dépendances liées aux exemples ne seront pas automatiquement incluses dans la bibliothèque ; par exemple : le support de différents formats d'images ou de fichiers audio.

2. Concernant les débutants, par expérience nous ne recommandons pas l'usage du gestionnaire de paquets *Homebrew*. Ce gestionnaire est flexible et offre un très bon niveau de contrôle sur ce qui est installé et comment les paquets sont installés. Le revers de la médaille est qu'un débutant en fera souvent un mauvais usage l'emmenant assez rapidement à des installations bancales.

Extrait du manuel d'installation disponible en ligne

A.1.2 Compilation et installation de *GL4Dummies*

(1) Récupérer le code source de la bibliothèque :

— **Soit** en récupérant, dans le sous-dossier releases, une release *GL4Dummies* et en la décompressant (par exemple : tar zxvf gl4dummies-X.Y.Z.tar.gz où X.Y.Z est la version de la release) afin d'obtenir le dossier souces de *GL4Dummies* ;

— **Soit** en récupérant directement le GIT (installer `git`, `pkg-config` et les outils autotools `automake, autoconf` et `libtool`) :

```
git clone https://github.com/noalien/GL4Dummies.git GL4Dummies
cd GL4Dummies
make -f Makefile.autotools
```

(2) Attention cette étape est découpée en trois parties, chacune correspond à une situation particulière de l'utilisateur (*Posix* avec droits administrateur, *Posix* sans droits administrateur et enfin *Windows*). Ainsi, depuis le dossier contenant le source de *GL4Dummies*, faire selon la situation :

Sous Posix (Linux, OSX, ...) avec droits administrateur, faire :

```
./configure
make
sudo make install
```

Puis, sur certains Linux, il sera nécessaire de spécifier que la variables d'environnements `$PATH` et `$LD_LIBRARY_PATH` pointent respectivement sur `/usr/local/bin` et `/usr/local/lib`. Ainsi, selon votre OS, à la racine de votre compte, vous devriez avoir (par ordre de préférence) un ou plusieurs des fichiers suivants : `.bashrc`, `.bash_profile`, `.profile` ou encore `.zprofile`. Editez l'un d'eux en ajoutant les lignes suivantes à la fin du fichier :

```
export PATH=$PATH:/usr/local/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
```

Ces fichiers permettent de paramétrer votre environnement utilisateur au moment du login (`.bash_profile` ou `.zprofile`) ou lors de l'ouverture d'un nouveau bash (`.bashrc`). Pour forcer la relecture d'un de ces fichiers de paramétrage, utilisez la commande `source`. Par exemple, si vous aviez ajouté les deux lignes au `.bashrc`, faire :

```
source ~/.bashrc
```

Extrait du manuel d'installation disponible en ligne

Sous Posix (Linux, OSX, ...) sans droits administrateur, il peut être nécessaire de créer un dossier local à la racine de votre compte et y installer la bibliothèque (bibliothèque statique et dynamique, *headers* et autres fichiers nécessaires). En premier lieu, vérifiez que votre variable d'environnement `$HOME` pointe bien à la racine de votre compte (`echo $HOME`) et tapez :

```
[ -d $HOME/local ] || mkdir $HOME/local
./configure --prefix=$HOME/local
make
make install
```

et là il vous reste à paramétrer vos chemins en fonction de `$HOME/local` où l'installation a été faite.
Pour que le système ait connaissance de ce chemin d'installation particulier (les dossiers `bin include lib` et `share` ont été ajoués dans `$HOME/local`) il faut renseigner ses variables d'environnement `$PATH` et `$LD_LIBRARY_PATH`. Selon votre OS, à la racine de votre compte, vous devriez avoir (par ordre de préférence) un ou deux des fichiers suivants `.bashrc`, `.bash_profile`, `.profile` ou encore `.zprofile`. Éditez l'un d'eux en ajoutant les deux lignes suivantes à la fin du fichier :

```
export PATH=$PATH:$HOME/local/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HOME/local/lib
```

Ces fichiers permettent de paramétrer votre environnement utilisateur au moment du login (`.bash_profile` ou `.zprofile`) ou lors de l'ouverture d'un nouveau bash (`.bashrc`). Pour forcer la relecture d'un de ces fichiers de paramétrage, utilisez la commande source. Par exemple, si vous aviez ajouté les deux lignes au `.bashrc`, faire :

```
source ~/.bashrc
```

Sous *Windows*, ouvrir le répertoire *Windows* à l'emplacement contenant le code source de *GL4Dummies*, et, selon votre IDE : *Code*::*blocks* ou *Visual Studio*, vous trouverez respectivement un projet `GL4Dummies.cbp` ou `GL4Dummies.sln`. Ouvrez celui qui correspond à votre IDE et compiler le projet. Dans le cas de *Visual Studio*, il compilera la bibliothèque puis les exemples utilisant la bibliothèque, il ne reste plus qu'à exécuter. Dans le cas de *Code*::*blocks* la bibliothèque est compilée et les binaires créés dans le dossier `bin/Win32` à partir de la racine du dossier contenant les sources. Les projets exemples (fichiers `.cbp`) se trouvant dans *Windows* peuvent maintenant être compilés et exécutés.

Extrait du manuel d'installation disponible en ligne

A.1.3 Paramétrage de vos futurs projets

En fonction de l'emplacement où est installée la *GL4Dummies*, il faudra paramétrer vos futurs projets (par exemple paramétrer le `Makefile`) pour qu'ils puissent compiler et *linker* avec *GL4Dummies*. Vous devez ainsi vérifier que la ligne de compilation contient bien le chemin vers le dossier `include` de *GL4Dummies* et que la ligne de *link* (`ld`) contient le chemin vers le dossier `lib` et demande à utiliser *GL4Dummies* (par exemple `-lGL4Dummies`).

Par exemple, quand la bibliothèque est installée dans un `$HOME/local`, en utilisant un `Makefile`, vérifier dans ce dernier que les variables `CPPFLAGS` et `LDFLAGS` contiennent au minimum :

```
CPPFLAGS = -I. -I$(HOME)/local/include $(shell sdl2-config --cflags)
LDFLAGS = -lm -L$(HOME)/local/lib -lGL4Dummies $(shell sdl2-config --libs)
```

Pour la situation spécifique d'utilisateurs travaillant sous *Visual Studio*, se référer à la sous-section A.2.1 qui détaille le paramétrage de projets dans ce cas.

A.2 À partir des binaires : Windows 10 – Visual Studio Community 2019

Pour cette partie, nous considérons que le lecteur utilise la version précompilée – pour Windows 10 et Visual Community 2019 – de *GL4Dummies* disponible à partir de l'adresse suivante :

`https://github.com/noalien/GL4Dummies/tree/master/builds/MSVC_19`

Une fois le dossier `MSVC_19` récupéré (cloner le *repository* ou en récupérer une archive), vous devez :

1. Copier les DLLs depuis le dossier `DLLs` de la manière suivante :
   (a) Copier toutes les DLLs depuis `DLLs/x86/` vers `"%systemroot%\SysWoW64"` (habituellement `"C:\Windows\SysWOW64\"`) ;
   (b) Copier toutes les DLLs depuis `DLLs/x64/` vers `"%systemroot%\System32"` (habituellement `"C:\Windows\System32\"`) ;
2. Copier le répertoire GL4D vers `"C:\"`, de manière à avoir `"C:\GL4D\"` sur votre système [3]

3. Vous pouvez aussi modifier les options de Visual Studio pour ajouter respectivement `"C:\GL4D\include;C:\GL4D\include\SDL2"` comme répertoires additionnels d'inclusion (*includes* ou *headers*), et `"C:\GL4D\lib\x86"` et/ou `"C:\GL4D\lib\x64"` comme répertoires additionnels respectivement pour des bibliothèques (les `.lib`) de plateformes `Win32` ou `x64`.

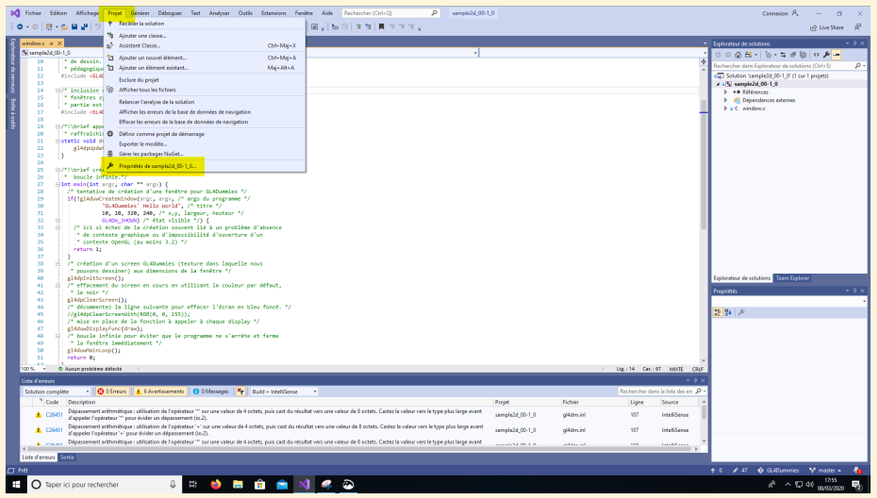
Extrait du manuel d'installation disponible en ligne

3. Retourner au *github* de *GL4Dummies*, téléchargez un *Sample Code* (dossier `samples`) vers votre répertoire de travail, utilisez le fichier `sln` – *the Visual Studio Solution file* (s'il y en a un, sinon créer le vôtre en mettant vos bons chemins vers les *includes* et les *libs*) – afin de compiler et exécuter l'exemple.

A.2.1 Paramétrer les projets pour utiliser *GL4Dummies*

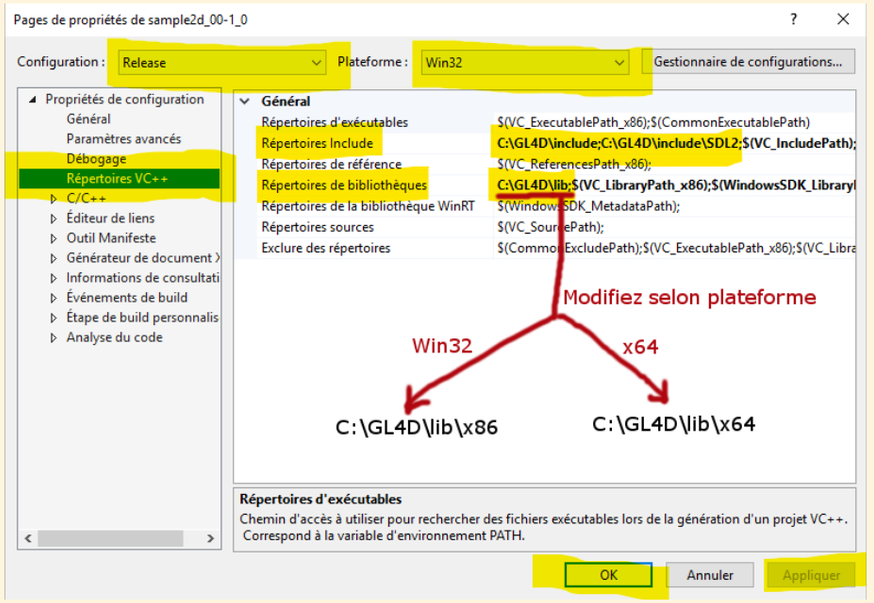
Nous reprenons, au travers des captures d'écran qui suivent, quelques éléments essentiels à la bonne configuration d'un projet Visual Studio afin que ce dernier puisse utiliser les bonnes bibliothèques permettant de compiler, lier et exécuter un programme graphique utilisant *GL4Dummies*, *SDL2* et OpenGL®. La typologie de projet généralement utilisée est "Application console" car elle permet de garder une sortie texte pour les messages de type informatif ou d'erreur mais elle n'est pas toujours nécessaire.

La capture d'écran ci-après montre le menu contextuel permettant d'accéder aux propriété du projet mis au premier plan (une solution pouvant être composée de plusieurs projets) :



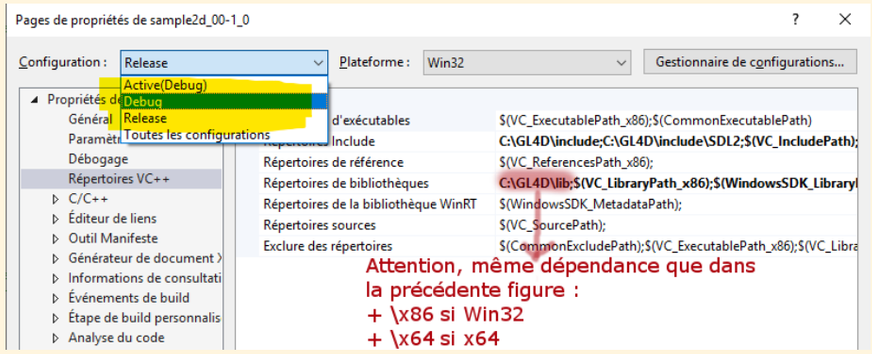
Extrait du manuel d'installation disponible en ligne

La fenêtre "page de propriétés" du projet permet d'accéder aux multiples options de "fabrication" du projet. Nous attirons votre attention sur le fait que ces options sont potentiellement liées à plusieurs configurations (ici *Debug* ou *Release*) et à plusieurs plateformes d'exécution (par exemple *Win32* ou *x64*). Après chaque changement dans une configuration ou au niveau de la plateforme, vous devez valider le changement via le bouton "Appliquer" avant de passer à une autre configuration ou plateforme. Ici nous mettons en valeur une catégorie d'options – *Répertoires VC++* – pour laquelle nous paramétrons les dossiers d'inclusion (où aller chercher les fichiers `"*.h"`) et les dossiers contenant les bibliothèques (où aller chercher les fichiers `"*.lib"`) en ajoutant des chemins liés à *GL4Dummies* :

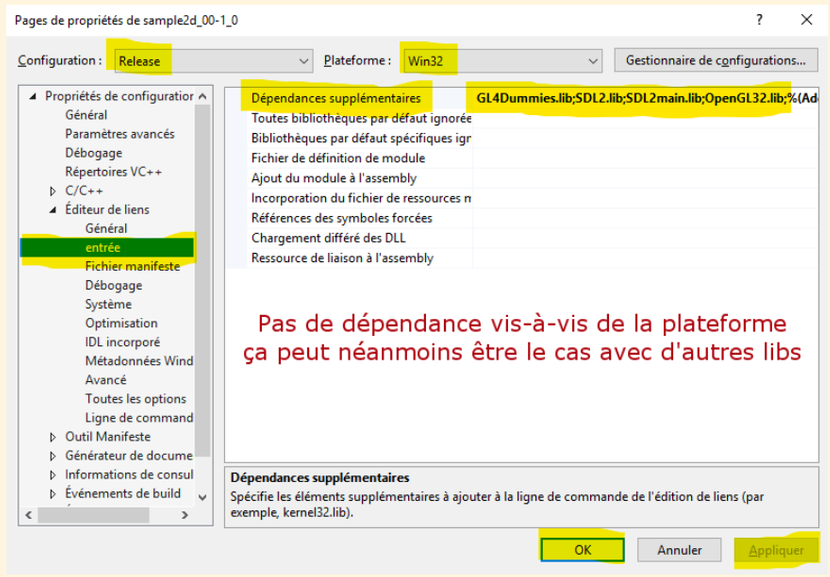


La capture d'écran ci-dessous montre justement un changement de configuration. Chacune a ses propres options, parfois certaines options sont communes et peuvent être modifiées en une fois via la sélection de la configuration "Toutes les configurations" (ceci est aussi valable pour les plateformes) :

Extrait du manuel d'installation disponible en ligne



Enfin, cette capture d'écran illustre les modification de la propriété "éditeur de liens → entrée". Le champs "Dépendances supplémentaires" indique nommément les bibliothèques utilisées lors de l'édition de liens. Dans notre cas, ceci rend le programme dépendant des DLLs *GL4Dummies*, *SDL2* et OpenGL® :

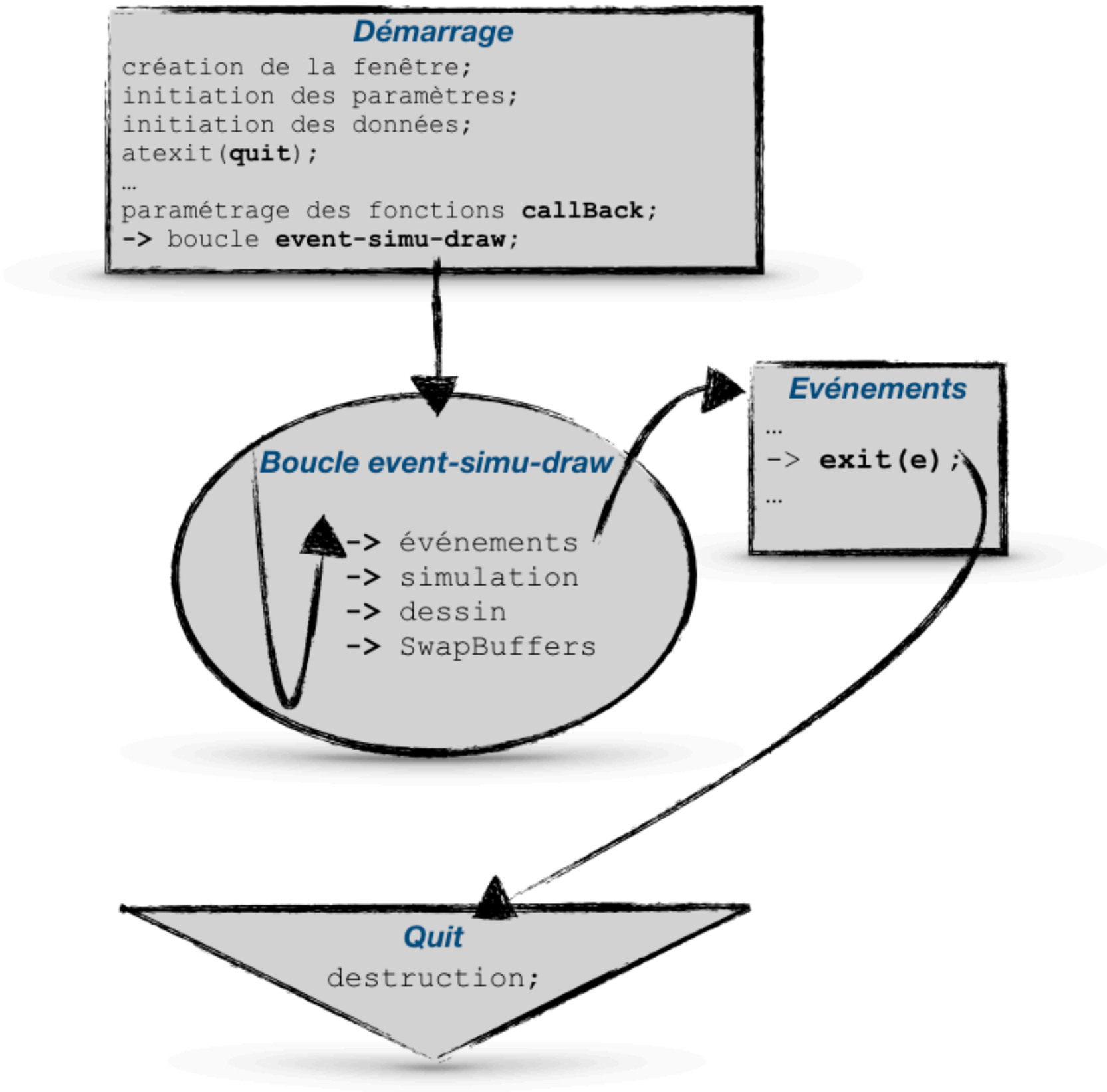


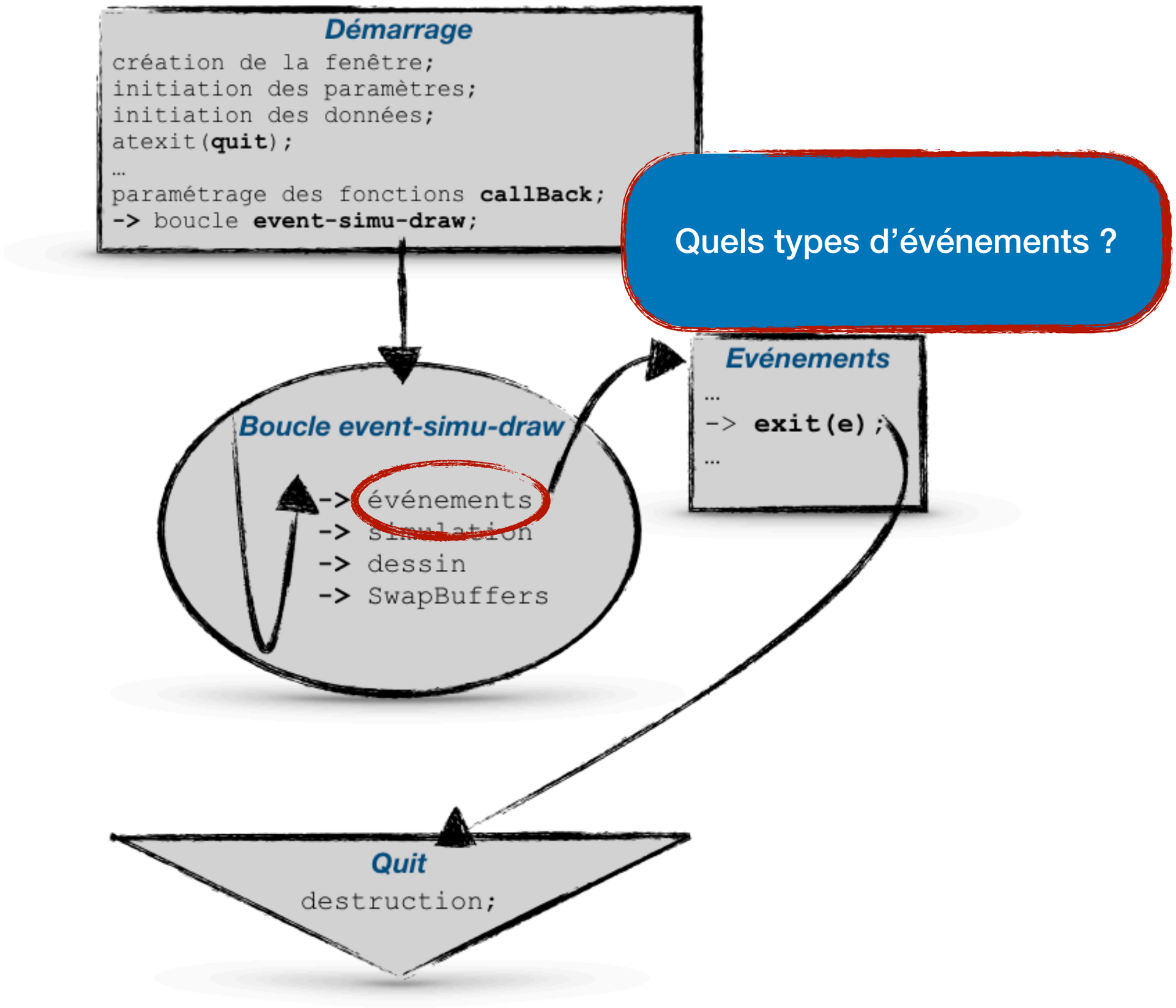
(2)
Introduction à l'interaction (slides)

Appliquée à GL4Dummies

Voir aussi **la documentation de référence** à l'adresse
https://gl4d.api8.fr/doxygen/html/index.html

Spécialement les fonctions là :
https://gl4d.api8.fr/doxygen/html/d1/d36/gl4duw__SDL2_8h.html
et celles là :
https://gl4d.api8.fr/doxygen/html/da/d0a/gl4dp_8h.html


Démarrage
création de la fenêtre;
initiation des paramètres;
initiation des données;
atexit(quit);
…
paramétrage des fonctions callBack;
-> boucle event-simu-draw;
Boucle event-simu-draw
-> événements
-> simulation
-> dessin
-> SwapBuffers
Evénements
…
-> exit(e);
…
Quit
destruction;


Démarrage
création de la fenêtre;
initiation des paramètres;
initiation des données;
atexit(quit);
…
paramétrage des fonctions callBack;
-> boucle event-simu-draw;
Quels types d'événements ?
Boucle event-simu-draw
-> événements
-> simulation
-> dessin
-> SwapBuffers
Evénements
…
-> exit(e);
…
Quit
destruction;

Quels types d’événements ?

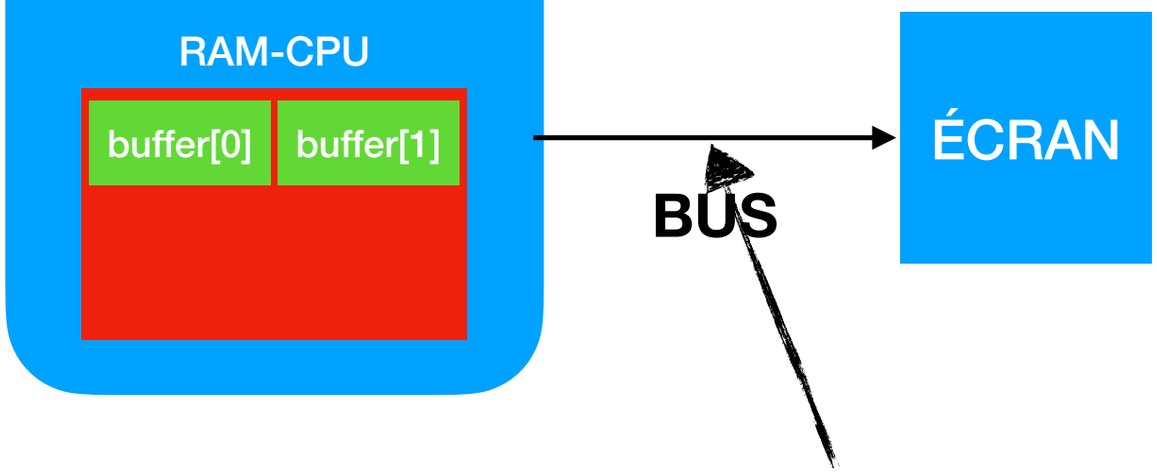
- Interactions clavier (plusieurs types)
- Interactions souris (plusieurs types)
- Actions sur la fenêtre :
    - retailler, perte de focus, fermeture, …
- Autres interactions possibles (extérieurs) :
    - Divers capteurs, caméra RGB, caméra RGBD, Motion Leap, …

Quels types d'événements ? (GL4Dummies)

- Interactions clavier (plusieurs types)
  - ➡ gl4duwKeyDownFunc(void(*)(int keycode) **mafonction**) => void **mafonction**(int code_de_latouche) { … }
  - ➡ gl4duwKeyUpFunc(void(*)(int keycode) **mafonction**) => void **mafonction**(int code_de_latouche) { … }
- Interactions souris (plusieurs types)
  - ➡ gl4duwMouseFunc(void(*)(int button, int state, int x, int y) **mafonction**) => void **mafonction**(int button, int state, int, x, int y) { … }
  - ➡ gl4duwMotionFunc(void(*)(int x, int y) **mafonction**) => void **mafonction**(int, x, int y) { … }
  - ➡ gl4duwPassiveMotionFunc(void(*)(int x, int y) **mafonction**) => void **mafonction**(int, x, int y) { … }
- Actions sur la fenêtre :
  - retailler, perte de focus, fermeture, …
    - ➡ gl4duw…
- Autres interactions possibles (extérieurs) :
  - Divers capteurs, caméra RGB, caméra RGBD, Motion Leap, …
    - ➡ **manuellement** dans gl4duwIdleFunc(void(*)(void) **mafonction**) => void **mafonction**(void) { … }

La parenthèse SwapBuffers (double buffering)
Améliorer la fluidité de l'application graphique

- **Deux buffers existent : buffer[0] et buffer[1]**
- **Mettre i, une variable d'indexation, à 0**
- **Début boucle**
- Calculer image et Écrire dans buffer[i]
- Quand terminé : demander le transfert depuis buffer[i] vers écran
  - Pendant ce temps le CPU est libre (non bloquant)
- Le CPU est donc libre de commencer à calculer l'image suivante
  - il modifie i en **i = (i + 1)%2** pour le faire dans l'autre buffer
    (si le transfert de ce dernier - démarré lors du précédent tour - est bien terminé, c'est généralement le cas)
- **Recommencer**



(3)
Suite dans le document d’introduction à la programmation 2D avec GL4D