*C*entre *I*nformatique pour les *L*ettres
et les *S*ciences *H*umaines

Apprendre C++ avec Qt : Utiliser les `QTimer`

La classe `QTimer` permet de générer automatiquement des signaux ayant une périodicité contrôlable. En connectant ces signaux à des slots, il est donc possible d'obtenir le déclenchement programmé de divers traitements.

1 - Mise en place

La mise en place d'un timer implique trois opérations :
- création d'une instance de la classe `QTimer` ;
- connexion du signal `timeout()` de ce `QTimer` à la fonction dont l'appel doit être déclenché périodiquement;
- mise en route du timer.

Les instructions effectuant ces trois opérations peuvent être placées dans le constructeur du dialogue de base (si le timer doit être mis en route dès le lancement du programme) ou dans une autre fonction membre de la classe de dialogue (si le timer ne doit être mis en route que lorsque cette fonction est exécutée).

```
1  QTimer * timer = new QTimer(this);
2  connect( timer, SIGNAL(timeout()), this, SLOT(fonctionAAppeler()) );
3  timer->start(500, FALSE ); //500 ms = deux fois par secondes
```

L'instanciation de la classe `QTimer` *(1)* est réalisée par *allocation dynamique*, ce qui permet de créer un objet qui n'est pas local à la fonction (seul le pointeur `timer` est local, l'objet dont il contient l'adresse continuera à exister jusqu'à ce le dialogue soit refermé).

La connexion *(2)* du signal "c'est l'heure" (`timeout`) à une fonction reprend très précisément la logique de la connexion d'un signal à un slot à l'aide de Qt Designer :

Les quatre paramètres de la fonction `connect()` correspondent exactement au quatre colonnes du dialogue d'édition des connections : sender, signal, receiver et slot.

Le receiver est le dialogue dont la `fonctionAAppeler()` est membre. Dans la mesure où le code établissant la connexion est lui-même placé dans une fonction membre de la classe dialogue, celui-ci peut être désigné à l'aide du pointeur `this` (cf. Leçon 9).

Une fois la connexion établie, la mise en route du timer passe simplement par l'appel de la fonction `start()`, dont les deux arguments indiquent respectivement la périodicité de l'appel de la `fonctionAAppeler()` et si l'appel doit être unique (en d'autres termes, si le second argument vaut `TRUE`, la fonction ne sera appelée qu'une seule fois, au bout d'un délai spécifié par le premier argument). Bien entendu,

Le fonctionnement de ce dispositif exige que la `fonctionAAppeler()` soit un slot.

Même si la connexion signal/slot n'est pas effectuée à l'aide de Qt Designer, rien ne s'oppose à ce que le slot soit créé avec cet outil.

2 - Contrôler le fonctionnement d'un timer

Il arrive évidemment que le timer ne soit pas simplement mis en route au début du programme en vue de le laisser fonctionner jusqu'à la fin de l'exécution de celui-ci. Lorsque des arrêts ou des changements de fréquence doivent avoir lieu pendant l'exécution du programme, il est nécessaire que le timer soit rendu accessible aux fonctions responsables de ces modifications. Le "pointeur sur `QTimer`" utilisé pour stocker l'adresse du `QTimer` créé par `new` ne doit donc pas être local à la fonction qui met le dispositif en place, mais membre de la classe de dialogue. Les autres fonctions membre pourront alors, par exemple, arrêter le timer et le remettre en route avec une fréquence différente :

```
timer.stop();
timer.start(333, FALSE); //333 ms = trois fois par secondes
```