

Mise en œuvre de *timer* – classe QTimer

« un petit programme vaut mieux qu'un long discours... », vous allez pouvoir utiliser l'exemple donné à http://grimaldi.univ-tln.fr/category/files/exemple_de-timer.zip.



Travail demandé :

En utilisant ce que vous avez appris au cours des programmes précédents, écrire une application graphique qui affiche un compte à rebours :

- au début, l'interface contient un LCDNumber qui affiche la valeur 10 et un bouton « lancer »
- quand on clique sur le bouton, il disparaît et la valeur décroît toutes les secondes (10, 9, 8, 7, 6, ..., 0)
- lorsque la valeur arrive à zéro, le bouton réapparaît et la valeur 10 revient dans le LCDNumber.
- un bloc de trois RadioButton devra permettre de choisir si la valeur est affichée en décimal, en binaire ou en hexadécimal.



Dessin direct sur l'interface et gestion de la souris

Dessin sur l'interface - méthode QPaintEvent

Principe :

Pour dessiner directement sur la fenêtre de l'application (sur l'interface), il faut tout d'abord redéfinir une méthode de type SLOT dans la classe représentant l'interface (widget.h ou mainwindow.h en général).

La déclaration du prototype de cette méthode doit être :

```
private slots:  
void paintEvent(QPaintEvent *event);
```

N'oubliez pas d'inclure le header (.h) correspondant : `#include <QPaintEvent>`

Dans le fichier d'implémentation (widget.cpp) vous pouvez maintenant écrire la méthode :

```
void Widget::paintEvent(QPaintEvent *event)  
{  
...  
}
```

Cette méthode sera appelée **automatiquement** par votre application, à chaque fois que la fenêtre doit être **dessinée** (« **paint** ») :

- au lancement de l'application, la première fois
- à chaque fois que la fenêtre est couverte/découverte par une autre fenêtre
- à chaque fois qu'on la réduit/agrandit
- .etc.

Vous avez également la possibilité **de forcer** ce dessin vous même par programme en appelant la méthode **repaint()**.

Pour dessiner (« peindre ») à proprement dit, il faut avoir recours aux services d'un *peintre* (« painter »). C'est lui, le painter, qui dessinera pour nous. « il est fort ce painter ! »

Comme toujours, « un petit programme vaut mieux qu'un long discours... ». Vous pouvez vous inspirer du projet à:

<http://grimaldi.univ-tln.fr/category/files/comment-dessiner-sur-l-interface.zip>.



Travail demandé :

- Dessiner les diagonales de la fenêtre en bleu puis un cercle rouge de rayon 50 au centre ; "en dur", c'est juste pour essayer !
- Créer une **Cercle2D** représentant un cercle du plan (vous pouvez réutiliser la classe **Position2D** du TP1 pour représenter son centre), avec une méthode **draw** qui dessine le cercle sur l'interface (avec l'aide d'un *peintre*, **QPainter**, qui sait dessiner une ellipse !); prototype : **void Cercle2D::draw(QPainter *p)**
- Utiliser cette méthode pour tracer le cercle dans la fenêtre (dans son **paintEvent**). Comme toujours, vérifier votre programme à chaque phase d'avancement !
- Ajouter une méthode **deplace** qui déplace le cercle avec rebonds sur les bords de la fenêtre. prototype **void Cercle2D::deplace(int width, int height)**; les arguments représentent la dimension de la fenêtre. Ajoutez des attributs s'il le faut.
- Au moyen d'un **timer**, écrire un programme qui fait déplacer automatiquement le cercle dans la fenêtre avec rebonds sur les bords.
- Si vous avez bien fait les choses, il est maintenant très simple de faire déplacer N cercles en utilisant un tableau de **Cercle2D**.
- Si vous avez fini et que vous ne savez pas quoi faire, faites déplacer une image que vous choisirez sur le Web (le *peintre* sait aussi dessiner une image)