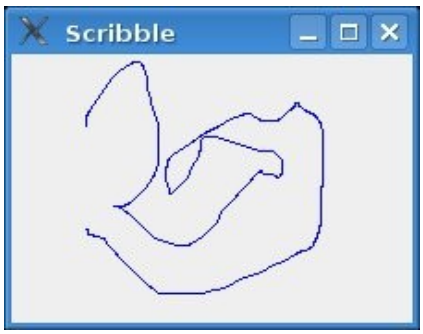
TP3 — *Logiciel de dessin : création de widget, QMainWindow et QSplitter*

Pour ce TP, il est demandé de réaliser une petite application de dessin permettant de dessiner à la souris des formes à main levée.

Exercice 1 : Implémentation des fonctionnalités de base

Dans un premier temps, il faut créer l'interface présentée ci-dessous :



Il faut implémenter la gestion du dessin sur le composant de dessin. Pour dessiner, il faut que le bouton de la souris soit enfoncé. Dès lors, lorsque la souris bouge un trait bleu apparaît reliant tous les points visités par la souris. Le dessin s'arrête dès que le bouton de la souris est relevé.

Pour ce faire, il faut redéfinir dans notre composant de dessin les méthodes de gestion des évènements liés à la souris :

```
void mousePressEvent(QMouseEvent *event);
void mouseMoveEvent(QMouseEvent *event);
void mouseReleaseEvent(QMouseEvent *event);
```

Ainsi notre composant de dessin aura des réponses spécifiques lors du passage de la souris sur sa zone, ou encore lorsqu'un des boutons sera enfoncé et relevé.

Indications :

- Il est conseillé de réaliser deux classes spécifiques. Une classe `DrawWidget` héritant de `QWidget` et qui contiendra les composants de l'application et notamment notre composant de dessin. Une classe `ZoneDessin` qui implémentera le composant de dessin.
- Pour pouvoir utiliser les fonctionnalités de dessin en dehors de la méthode `paintEvent`, il est nécessaire d'ajouter « `setAttribute(Qt::WA_PaintOutsidePaintEvent);` » dans le constructeur de notre zone de dessin;
- On remarquera que dès que la souris sort de la zone de l'application, l'intégralité du dessin disparaît. Ceci est normal pour l'instant et sera amélioré dans les phases suivantes.

Exercice 2 : Gestion du dessin en « double buffering »

Maintenant, il est demandé de modifier le programme précédent pour corriger le fait que le dessin ne se conserve pas dès que la souris sort de la zone de dessin ou que l'application est iconifiée ou redimensionnée. Une solution pour corriger ce défaut consiste à gérer un buffer pour le dessin et de rafraîchir la zone à dessiner en fonction de ce buffer.

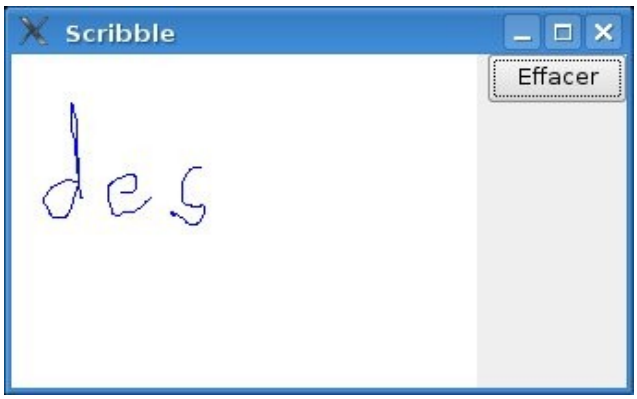
Plus précisément, il est nécessaire de gérer une image (un `QImage` par exemple) dans laquelle seront effectués tous les dessins. Puis, lorsque que l'application aura besoin de se redessiner (lors d'un redimensionnement par exemple), il faudra copier le contenu de l'image sur la zone de dessin. À chaque fois qu'un composant à besoin de se redessiner, sa méthode `paintEvent` est appelée. De plus, il est possible de forcer le composant à se rafraîchir (se redessiner) en appelant sa méthode `update`.

Il est nécessaire que la taille de l'image buffer soit au moins de la taille de la zone de dessin quelque soit la taille de la zone dessin. On en profitera pour mettre un fond blanc par défaut à la zone de dessin (i.e. l'image buffer a aussi un fond blanc) mais on doit pouvoir choisir la couleur de fond lors

de l'instanciation d'un objet DrawWidget. On doit donc obtenir une application du type :

Exercice 3 : Gestion de l'effacement

Pour pouvoir recommencer notre dessin, nous rajoutons un bouton d'effacement à notre application. Le bouton « Effacer » efface la zone de dessin. Attention, lorsqu'il n'y a rien à dessiner (c-à-d au lancement de l'application ou lorsque l'on vient juste d'appuyer sur le bouton « Effacer ») le bouton est inactif :



La gestion de l'effacement peut être gérée grâce à l'implémentation de « slots » et « signaux » sur notre composant de dessin. Il faut donc rajouter le bouton à notre application et créer les slots et signaux nécessaires à son fonctionnement.

Faites attention à la politique de dimensionnement de vos objets ZoneDessin(i.e. : le bouton effacer doit toujours être à sa taille minimum).

Exercice 4 : La classe QMainWindow et le widget QSplitter

Nous allons maintenant créer une classe MainWindow qui héritera de QMainWindow. Cette classe doit nous permettre de réaliser les affichages ci dessous. Les étirements des widgets DrawWidget seront possibles grâce au widget QSplitter. Vous devrez attentivement lire la documentation de QtAssistant pour réaliser cet exercice.