Qt est un logiciel libre qui permet de créer des interfaces graphiques d’utilisateur (GUI, l’abréviation de Graphical User Interface). Un programme avec GUI représente beaucoup plus d’avantages par rapport à un programme sans GUI, telles que la lisibilité et la simplicité d’utilisation etc., c’est pour cela qu’on a opté la programmation en Qt. Ceci est faisable grâce à la documentation dédiée sur le site Openclassrooms.

À la création de notre programme, on a choisi un projet vide de type d’application Qt avec widgets. Car dans la suite grâce à des classes QObject, Qwidget etc. qui sont propres à Qt, on a pu créer des classes héritière de Qwidget pour faciliter le travail.

Au lieu d’utiliser les userformes (.ui), j’ai choisi de coder directement dans des classes C++ pour une raison simple: faciliter le déboggage. Par exemple, une fois qu’on commence à crée des boutons en mode design, il devient ultra compliqué si on veux switcher en mode éditeur. Prenons exemple de MainWindow, après avoir crée deux boutons en mode design, j’ai voulu ajouté une image png et à défaut d’un idée de le faire sans code, j’ai ensuite passé en mode éditeur et ai utilisé SetPixmap et QGridLayout, mais Qt n’a pas voulu en disant qu’il existe déjà un Layout dans la fenêtre. Finalement après avoir passé une demie journée en déboggage j’ai laissé tombé la fenêtre MainWindow et j’ai recréé une autre fenêtre et ai défini une classe proprement.

La plus grande difficulté réside dans l’allocation dynamique des mémoires. Pour que les variables survivent à la fonction dans laquelle elles ont été créée et qu’elles soient utilisables partout dans la classe, on a besoin de créer des variables globales (déclarées dans la classe) et d’utiliser des pointeurs avec allocation dynamique. On se rappelle que ces variables sont instanciées par new et doivent être détruit à la fin d’utilisation, c’est à dire, à la sortie de la classe. Ce qui nous arrive des fois c’est un message de warning à la fermeture des fenêtres, qui dit que le programme s’est terminé subitement. Il s’agissait bien un problème de destructeur. Il est peut-être bête de répéter la différence entre variable globale et variable locale, mais il est très important d’y penser, sinon on peut facilement se planter et est obligé de débogguer pendant des heures.

Une particularité des programme Qt Un objet type est défini par les attributs et les méthodes et en revanche, un objet définie en Qt est constitué des attributs, des méthodes, des signaux et des slots, ces deux derniers permettent de gérer des événements. La fonction Connect permet de lier un signal à un slot, une fois la signal émise, le slot va être déclenché de suite. On peut également lier un signal à un autre signal. Un exemple type dans le code est que quand on click sur un bouton il se passe quelques choses: la fermeture de la fenêtre, une fonction spécifique (open\_result(), disable() etc.)

les types de signal pourrait être clicked(), pessed(), released(), toggled() etc, et les slots peuvent être quit(), QMessageBox() ou des fonctions personnalisées.

Il faut faire attention que dans la définition de la classe, il est indispensable d’ajouter « Q\_OBJECT » et déclarer les slots sous « public slots: ».

La construction des fenêtres

La fenêtre d’accueil est nommé GuideWindow, qui comprend deux boutons et deux labels, l’une affichant un image et l’autre une phrase. Et les deux boutons se dirigent vers deux fenêtres filles, MyWindow et SimWindow respectivement.

Elles ont les mêmes interfaces et se différencient par la sortie, MyWindow donne des conseils d’ordre à placer sur le marché tandis que SimWindow simule la perte potentielle.

Enfin, la fenêtre ResultWindow affichant le résultat ou la simulation prend en argument de deux types de classe, on peut les définir séparément dans le constructeur en spécifiant les arguments.

Attention…

Pour que l’image se charge correctement, il faut que l’image soit stocké dans le fichier de l’exécutable mais pas celui des ressources.