Université Pierre et Marie Curie

Modelisation Objet

TMEs 8, 9 & 10 – Visualisateur de Netlist

Auteur:
William Fabre

Professeur:
Madame Marie-Minerve Louerat

Année 2018-2019

Contents

TME 8

1.1 CellViewer

"Fenêtre principale, avec la version dégénérée (fournie) du CellWidget."

Il faut faire attention a ne pas oublier "public slot" pour decrire les fonctions qui seront par la fenetre QT. Apres adaptations des classes au moedele avec des dossier. On rajoute la fonction OpenCell qui a le meme format que saveCell recherche de la cell et on load si elle n'est pas existante. On rajoute cette nouvelle fonction en connectant la fonction avec "connect" et on rajoute une action dans le menu deroulant.

1.2 SaveCellDialog

"Dialogue de sauvegarde d'une Cell."

Rien a faire on prend par contre ceci comme exemple pour OpenCell. Cette classe decrit "la fenetre attachee (invisible) a la creation de l'objet". La methode run lance la fonction exec() qui effectue 3 operations:

1. montrer la fenetre
2. montrer le resultat (Widget qui tourne)
3. lorsque le resultat est produit, masque la fenetre et renvoyer le resultat dialogResult.

Tant qu'une option comme OK/CANCEL n'est pas choisie l'application graphique est bloquee.

1.3 OpenCellDialog

"Dialogue de chargement d'une Cell."

On va changer un petit peu par rapport a SaveCell car une remarque a ete demandee, transformer la methode run en static. On ne va donc pas ajouter un element au constructeur du CellViewer. On doit aussi mettre la fenetre parent dans le run car il n'a pas ete initialise du coup. L'element est cree a la volee et detruit a la volee, cela diminue son emprunte memoire.

TME 9

Dans les deux cas il faudra instancerr la classe qui gere le widget averc Qwidget(parent, Qt::Window) pour qu'il ne cree pas la fenetre a l'interieur de la fenetre parent et qu'il cree bien une nouvelle fenetre. La structure est la meme pour les deux types de fenetres, fenetre sur instances ou fenetre sur cells. Dans les fonctions de set on ne doit pas oublier de disconnect lorsqu'on setCellviewer ou setInstanceviewer avec la fonction disconnect(this, 0, 0 , 0) avant de reasigner un cellviewer. Pour fermer les fenetres sans les detruires il faut showMinimized() et ignorer l'evenement de fermeture avec event ignore()

2.1 InstancesWidget, InstancesModel

"Fenêtre affichant la liste des instances et leurs modèles."

Dans cette classe qui nous est deja donnee il n'y a rien a changer a part ce que j'ai precise en preembule pour avoir le comportement requis.

2.2 CellsLib, CellsModel

"Fenêtre affichant la liste de toutes les cells chargées en mémoire."

Ici on ne fait que copier la classe pour l'instance et l'adapter a la cell, on ne va donc pas get sur l'instance ou parcourir l'instance mais le faire sur les cells. L'affichage est legerement different.

TME10

3.1 CellWidget

"Widget affichant le dessin complet d'une Cell. On implantera progessivement l'affichage des différents éléments:" Tout se passe dans quelques fonctions, particulierement paintEvent et Query. Dans paint event on prepare des requetes de dessins et je n'ai pas du tout factorise le code, ce qui serait une grande amelioration. Le code se constitue donc comme une serie de query entre lesquels je change la couleur pour afficher les differents elements qui constituent la cell. Chaque query a un numero et va donc afficher une seule chose a chaque fois. Pour la resumer, la fonction query est un enorme switch case.

1. Dessin du symbole des instances : On prepare les differentes shapes et en fonction de ce qu'on nous demande nous allons creer des points avec les fonction pointToScreenPoint ou boxToScreen-Box. Par exemple pour une ligne on creera deux points, pour une ellipse, un rectangle, pour un arc j'ai precise dans le code qu'il faut une multiplication par 16 car nous avons 1/16 d'angle (void doc QTellipse), pour finir nous avons lels box ou il ne faut juste dessiner des Rect. On dessine le nom tout a la fin car si on translate la position avant, tout risque d'etre translate. Il serait bien de prendre une variable tampon pour ne pas directement translater la position de l'instance pour afficher le nom.
2. Dessin des connecteurs des instances : Pour les terms on va chercher avec getTerms de la cell et on va afficher les point avec un simple DrawPoint
3. Dessin des connecteurs du modèle : Pour lles connecteurs il faut gerer plusieurs cas pour avoir des petites formes de maisons pour les connecteurs exterieurs de la cell, des points pour les connecteurs internes et des rond pour les connections sur fils, les nodePoint. Je me suis permis de laisser des connecteurs en forme de maison pour une celle contenant des cells mais il faudrait peut etre songer a changer ca. Pour dessiner une petite maison on dessine le connecteurs en gros point et on dessine un triangle a cote de lui dans le bon sens en fonction de si c'est un in ou un out.
4. Dessin des fils : On va parcourrir les lines du net et on les affiche de la source vers la target avec des lignes.
5. Fonctions de déplacement (haut, bas, gauche droite). : en bi coordonnes cela donne : x est la variation gauche vers droite respectivement negatif vers positif et y est la variation haut vers bas respectivement negatif vers positif