# !/usr/bin/env python3  
# -\*- coding: utf-8 -\*-  
# Created by Axel Tremaudant on 10/06/2022  
  
*"""  
Fichier contenant toutes les valeurs utiles a l'application dans la classe InitData.  
Ces valeurs ne changent pas pendant que l'application tourne.  
"""*from PyQt5 import QtCore, QtWidgets, QtGui  
  
  
class Init:  
 *"""  
 Classe contenant toutes les valeurs utiles a CrubsRunner.  
 """* def \_\_init\_\_(self):  
 self.window = { # Donnees pour la fenetre principale  
 'app\_title': "CrubsRunner",  
 'app\_icon': "icon/icon\_app.png",  
 'organisation\_name': "Crubs",  
  
 'name\_tool\_bar': "Projet",  
 'tool\_bar\_movable': False,  
 'speed\_tip': "Vitesse de déplacement du robot",  
 'cursor\_while\_opening': QtCore.Qt.WaitCursor,  
 'normal\_cursor': QtCore.Qt.ArrowCursor,  
  
 'new\_project\_name': "Nouveau",  
 'new\_project\_status\_tip': "Créer un nouveau projet",  
 'new\_project\_shortcut': QtGui.QKeySequence.New, # Ctrl + N  
 'new\_project\_icon': "icon/icon\_new.png",  
  
 'open\_project\_name': "Ouvrir",  
 'open\_project\_shortcut': QtGui.QKeySequence.Open, # Ctrl + O  
 'open\_project\_status\_tip': "Ouvrir un projet",  
 'open\_project\_icon': "icon/icon\_open.png",  
 'open\_project\_dialog\_title': "Ouvrir un projet",  
  
 'save\_project\_name': "Enregistrer",  
 'save\_project\_shortcut': QtGui.QKeySequence.Save, # Ctrl + S  
 'save\_project\_status\_tip': "Enregistrer le projet",  
 'save\_project\_icon': "icon/icon\_save.png",  
  
 'save\_as\_project\_name': "Enregistrer sous",  
 'save\_as\_project\_shortcut': QtGui.QKeySequence.SaveAs, # Ctrl + Shift + S  
 'save\_as\_project\_status\_tip': "Enregistrer sous le projet",  
 'save\_as\_project\_icon': "icon/icon\_save\_as.png",  
 'save\_as\_project\_dialog\_title': "Enregistrer le projet",  
 'project\_extension': "CrubsRunner project (\*.crp)",  
 'project\_default\_name': "SansNom",  
  
 'saving\_file\_first\_line': "## Fichier de sauvegarde de projet CrubsRunner du {date}\n",  
 'window\_first\_line': "\n## Window\n",  
 'grid\_first\_line': "\n## Grid\n",  
 'board\_first\_line': "\n## Board\n",  
 'main\_robot\_first\_line': "\n## Main robot\n",  
 'second\_robot\_first\_line': "\n## Second robot\n",  
 'gcrubs\_first\_line': "\n## gcrubs\n",  
 'vinyl\_first\_line': "\n## vinyl\n",  
  
 'import\_name': "Importer",  
 'import\_shortcut': QtGui.QKeySequence.Italic, # Ctrl + I  
 'import\_status\_tip': "Importer un composant",  
 'import\_icon': "icon/icon\_import.png",  
 'import\_dialog\_title': "Choisir le composant à importer",  
 'import\_dialog\_modal': True,  
 'import\_radio\_board\_name': "Plateau",  
 'import\_radio\_vinyl\_name': "Tapis",  
 'import\_radio\_main\_robot\_name': "Robot principal",  
 'import\_radio\_second\_robot\_name': "Robot secondaire",  
 'import\_radio\_main\_robot\_checked': False,  
 'import\_radio\_board\_checked': True,  
 'import\_radio\_vinyl\_checked': False,  
 'import\_radio\_second\_robot\_checked': False,  
 'import\_cancel\_btn\_name': "Annuler",  
 'import\_ok\_btn\_name': "Valider",  
 'import\_cancel\_btn\_checked': True,  
 'import\_ok\_btn\_checked': False,  
 'import\_height': 150,  
 'import\_width': 280,  
  
 'export\_name': "Exporter",  
 'export\_shortcut': QtGui.QKeySequence(QtCore.Qt.CTRL | QtCore.Qt.Key\_E),  
 'export\_status\_tip': "Exporter un composant",  
 'export\_icon': "icon/icon\_export.png",  
 "export\_dialog\_title": "Exporter un composant", # Other parameters : same as import  
 'export\_component\_dialog\_min\_size': (280, 150),  
  
 'top\_view\_action\_name': "Vue de dessus",  
 'top\_view\_action\_shortcut': QtGui.QKeySequence.AddTab, # Ctrl + T  
 'top\_view\_action\_status\_tip': "Vue de dessus",  
 'top\_view\_action\_icon': "icon/icon\_top\_view.png",  
  
 'start\_view\_action\_name': "Vue de départ",  
 'start\_view\_action\_shortcut': QtGui.QKeySequence.Underline, # Ctrl + U  
 'start\_view\_action\_status\_tip': "Vue de départ",  
 'start\_view\_action\_icon': "icon/icon\_start\_view.png",  
  
 'bottom\_view\_action\_name': "Vue de dessous",  
 'bottom\_view\_action\_shortcut': QtGui.QKeySequence.Bold, # Ctrl + B  
 'bottom\_view\_action\_status\_tip': "Vue de dessous",  
 'bottom\_view\_action\_icon': "icon/icon\_bottom\_view.png",  
  
 'undo\_name': "Annuler le déplacement",  
 'undo\_shortcut': QtGui.QKeySequence.Undo, # Ctrl + Z  
 'undo\_status\_tip': "Annuler le déplacement",  
 'undo\_icon': "icon/icon\_undo.png",  
 'max\_len\_doing': 50,  
  
 'redo\_name': "Remettre le déplacement",  
 'redo\_shortcut': QtGui.QKeySequence.Redo, # Ctrl + Shift + Z ou Ctrl + Y  
 'redo\_status\_tip': "Remettre le déplacement",  
 'redo\_icon': "icon/icon\_redo.png",  
  
 'key\_action\_name': "Choisir les touches",  
 'key\_action\_status\_tip': "Choisir les touches",  
 'key\_action\_icon': "icon/icon\_key.png",  
  
 'edit\_action\_icon': "icon/icon\_edit\_gcrubs.png",  
 'del\_btn\_icon': "icon/icon\_del.png",  
 'key\_btn\_icon': 'icon/icon\_key.png',  
 'run\_action\_icon\_stopped': "icon/icon\_run\_stopped.png",  
 'run\_action\_icon\_running': "icon/icon\_run\_running.png",  
 'stop\_run\_action\_icon': "icon/icon\_stop\_run.png",  
  
 'speed\_simulation\_btn\_name': "x {multi}",  
 'speed\_simulation\_btn\_tip': "Vitesse de la simulation",   
 'speed\_simulation\_btn\_values': (0.25, 0.5, 1, 2, 4),  
  
 'window\_title': "CrubsRunner",  
 'accept\_drops': True,  
 'window\_start\_width': 1200,  
 'window\_start\_height': 800,  
  
 'add\_component\_dock\_area': QtCore.Qt.RightDockWidgetArea, # Place le dockWidget a droite  
 'name\_component\_dock': "Composants",  
 'component\_dock\_allowed\_areas': QtCore.Qt.RightDockWidgetArea | QtCore.Qt.LeftDockWidgetArea,  
 'component\_dock\_features': QtWidgets.QDockWidget.DockWidgetFeature.DockWidgetMovable, # Ne peut etre ferme  
 'properties\_dock\_allowed\_areas': QtCore.Qt.RightDockWidgetArea | QtCore.Qt.LeftDockWidgetArea,  
 'add\_properties\_dock\_area': QtCore.Qt.RightDockWidgetArea,  
 'properties\_dock\_features': QtWidgets.QDockWidget.DockWidgetFeature.DockWidgetMovable,  
 'sequence\_dock\_allowed\_areas': QtCore.Qt.RightDockWidgetArea | QtCore.Qt.LeftDockWidgetArea,  
 'add\_sequence\_dock\_area': QtCore.Qt.LeftDockWidgetArea,  
 'sequence\_dock\_features': QtWidgets.QDockWidget.DockWidgetFeature.DockWidgetMovable,  
  
 'import\_message\_box\_type': QtWidgets.QMessageBox.Warning,  
 'import\_message\_box\_title': "Erreur d'importation",  
 'import\_message\_box\_message': "Vous devez supprimer le composant avant d'en importer un autre.",  
 "drop\_message\_box\_message": "Les deux robots sont déjà attribués, veuillez en supprimer un avant "  
 "d'en ajouter un autre.",  
  
 'status\_bar\_message': "Position du {element} : ({x}, {y}) mm",  
 'color\_status\_message': "Couleur : r = {r}, v = {v}, b = {b} ",  
 'position\_status\_message': "Position : x = {x} mm, y = {y} mm, angle = {angle} degres ",  
  
 'menu\_bar\_menu1': "&Fichier",  
 'menu\_bar\_menu2': "&Édition",  
 'menu\_bar\_menu3': "&Run",  
  
 'error\_open\_file\_type': QtWidgets.QMessageBox.Critical,  
 'error\_open\_file\_title': "Erreur à l'ouverture ou à la lecture",  
 'error\_open\_file\_message': "Le fichier '{filename}' n'a pas été trouvé ou n'a pas pu être lu.'",  
  
 'error\_format\_file\_type': QtWidgets.QMessageBox.Critical,  
 'error\_format\_file\_title': "Format non supporté",  
 'error\_format\_file\_message': "Le format du fichier '{filename}' n'est pas supporté.'",  
  
 'keys\_modal': True,  
 'keys\_title': "Choix des touches",  
 'keys\_lbl\_0': "Aller a droite : ",  
 'keys\_lbl\_1': "Aller a gauche : ",  
 'keys\_lbl\_2': "Aller en haut : ",  
 'keys\_lbl\_3': "Aller en bas : ",  
 'keys\_lbl\_4': "Tourner a droite : ",  
 'keys\_lbl\_5': "Tourner a gauche : ",  
 'keys\_lbl\_key': "Touche : {key}",  
 'keys\_apply\_btn\_name': "Appliquer",  
 'keys\_apply\_default': True,  
 'keys\_apply\_cursor': QtCore.Qt.PointingHandCursor,  
 'keys\_close\_btn\_name': "Annuler",  
 'keys\_close\_default': False,  
 'keys\_close\_cursor': QtCore.Qt.PointingHandCursor,  
 } # end self.window  
  
 self.board = { # Contient toutes les donnees pour le plateau  
 'type': "board",  
 'name': "Plateau",  
  
 'new\_message\_box\_type': QtWidgets.QMessageBox.Information,  
 'new\_message\_box\_title': "Information",  
 'new\_message\_box\_message': "Choisir le plateau",  
 'file\_dialog\_open\_title': "Choisir le plateau",  
 'file\_dialog\_open\_extensions': "All files (\*.stl \*.obj \*.3mf \*.crb) ;; STL (\*.stl) ;; OBJ (\*.obj)"  
 ";; 3MF (\*.3mf) ;; CrubsRunner board (\*.crb)",  
  
 'save\_extension': "CrubsRunner board (\*.crb)",  
 'color': (255 / 255, 211 / 255, 133 / 255, 1), # de 0 a 1  
 'edge\_color': (105 / 255, 105 / 255, 105 / 255, 1), # de 0 a 1  
  
 'element\_name': "Plateau",  
 'appearance\_translation\_x': -1500,  
 'appearance\_translation\_y': 1000,  
 'appearance\_translation\_z': -1,  
  
 'window\_title': "Propriétés du plateau",  
  
 'color\_name': "Choisir la couleur du plateau",  
 'color\_cursor': QtCore.Qt.PointingHandCursor,  
 'color\_default': False,  
 'color\_dialog\_title': "Couleur du plateau",  
 'edge\_color\_name': "Choisir la couleur des arrêtes",  
 'edge\_color\_default': False,  
 'edge\_color\_dialog\_title': "Couleur des arrêtes du plateau",  
  
 'axis\_rotation\_x\_name': "x",  
 'axis\_rotation\_y\_name': "y",  
 'axis\_rotation\_z\_name': "z",  
 'angle\_rotation\_min': -180,  
 'angle\_rotation\_max': 180,  
 'gb\_name': "Mise en place du plateau",  
 'angle\_lbl\_name': "Angle : ",  
 'axis\_lbl\_name': "Axe de rotation : ",  
  
 'offset\_lbl\_name': "Hauteur : ",  
 'offset\_sb\_min': -3000,  
 'offset\_sb\_max': 3000,  
  
 'close\_btn\_name': "Fermer",  
 'close\_cursor': QtCore.Qt.PointingHandCursor,  
 'close\_default': True,  
 'reset\_btn\_name': "Reset",  
 'reset\_cursor': QtCore.Qt.PointingHandCursor,  
 'reset\_default': False,  
 'remove\_btn\_name': "Supprimer",  
 'remove\_default': False,  
 'remove\_cursor': QtCore.Qt.PointingHandCursor,  
  
 'remove\_message\_box\_type': QtWidgets.QMessageBox.Question,  
 'remove\_message\_box\_title': "Suppression d'élément",  
 'remove\_message\_box\_message': "Etes-vous sûr de vouloir supprimer le plateau ?\n"  
 "Cette action est irréversible.",  
 'remove\_message\_box\_buttons': QtWidgets.QMessageBox.No | QtWidgets.QMessageBox.Yes,  
 } # end self.board  
  
 self.vinyl = { # Contient toutes les donnees pour le vinyl  
 'vinyl\_message\_box\_type': QtWidgets.QMessageBox.Information,  
 'vinyl\_message\_box\_title': "Information",  
 'vinyl\_message\_box\_message': "Choisir le tapis du plateau",  
 'vinyl\_dialog\_open\_title': "Choisir le tapis du plateau",  
 'vinyl\_dialog\_open\_extensions': "All files (\*.png \*.pdf \*.jpg \*.jpeg) ;; PNG (\*.png) ;; PDF (\*.pdf)"  
 ";; JPEG (\*.jpg \*.jpeg)",  
 'remove\_message\_box\_type': QtWidgets.QMessageBox.Question,  
 'remove\_message\_box\_title': "Suppression d'élément",  
 'remove\_message\_box\_message': "Etes-vous sûr de vouloir supprimer le tapis ?\n"  
 "Cette action est irréversible.",  
 'remove\_message\_box\_buttons': QtWidgets.QMessageBox.No | QtWidgets.QMessageBox.Yes,  
 } # end self.vinyl  
  
 # Contient toutes les donnees pour le robot principal et les donnees communes aux deux robots  
 self.main\_robot = {  
 'type': "robot",  
 'name': "Robot principal",  
  
 'new\_message\_box\_type': QtWidgets.QMessageBox.Information,  
 'new\_message\_box\_title': "Information",  
 'new\_message\_box\_message': "Choisir le robot principal",  
 'file\_dialog\_open\_title': "Choisir le robot principal",  
 'file\_dialog\_open\_extensions': "All files (\*.stl \*.obj \*.3mf \*.crr) ;; STL (\*.stl) ;; OBJ (\*.obj) ;; "  
 "3MF (\*.3mf) ;; CrubsRunner robot (\*.crr)",  
 'color': (29 / 255, 144 / 255, 18 / 255, 1), # de 0 a 1  
 'edge\_color': (12 / 255, 73 / 255, 10 / 255, 1), # de 0 a 1  
 'save\_extension': "CrubsRunner robot (\*.crr)",  
  
 'element\_name': "Robot principal",  
 'appearance\_translation\_x': 0,  
 'appearance\_translation\_y': 0,  
 'appearance\_translation\_z': 0,  
  
 'window\_title': "Propriétés du robot principal",  
 'color\_name': "Choisir la couleur du robot",  
 'edge\_color\_name': "Choisir la couleur des arrêtes",  
 'color\_dialog\_title': "Couleur du robot",  
 'edge\_color\_dialog\_title': "Couleur des arrêtes",  
 'remove\_message\_box\_message': "Etes-vous sûr de vouloir supprimer le robot principal ?\n"  
 "Cette action est irréversible.",  
 'axis\_rotation\_x\_name': "x",  
 'axis\_rotation\_y\_name': "y",  
 'axis\_rotation\_z\_name': "z",  
 'angle\_rotation\_min': -180,  
 'angle\_rotation\_max': 180,  
  
 'gb\_name': "Mise en place du robot sur le plateau",  
 'angle\_lbl\_name': "Angle : ",  
 'axis\_lbl\_name': "Axe de rotation : ",  
 'offset\_lbl\_name': "Hauteur : ",  
 'offset\_sb\_min': -3000,  
 'offset\_sb\_max': 3000,  
 'invisible\_coef': 1000,  
  
 'speed\_lbl': "Vitesse (mm/s) : ",  
 'speed\_min': 1,  
 'speed\_max': 1000,  
 'speed\_rotation\_lbl': "Vitesse de rotation (degrés/s) : ",  
 'rotation\_min': 1,  
 'rotation\_max': 360,  
 'gb\_speed\_name': "Vitesses",  
 'track\_visible\_cb\_name': "Voir la trace",  
 'track\_visible\_cb\_checked': True,  
  
 'sequence\_btn\_name': "Creer la séquence du robot",  
 'sequence\_btn\_default': False,  
 'sequence\_btn\_cursor': QtCore.Qt.PointingHandCursor,  
  
 'import\_gcrubs\_btn\_name': "Importer le fichier séquentiel",  
 'import\_gcrubs\_btn\_default': False,  
 'import\_gcrubs\_btn\_cursor': QtCore.Qt.PointingHandCursor,  
 'import\_gcrubs\_title': "Choisir le fichier séquentiel",  
 'import\_gcrubs\_extension': "Fichier séquentiel (\*.gcrubs)",  
  
 'sequence\_dialog\_title': "Séquence du robot principal",  
 'sequence\_text': "{comment} Séquence gcrubs générée par CrubsRunner le {date} pour le robot principal.\n",  
 'sequence\_save\_btn\_name': "Enregistrer",  
 'sequence\_save\_btn\_default': True,  
 'sequence\_save\_btn\_cursor': QtCore.Qt.PointingHandCursor,  
 'sequence\_cancel\_btn\_name': "Fermer",  
 'sequence\_cancel\_btn\_default': False,  
 'sequence\_cancel\_btn\_cursor': QtCore.Qt.PointingHandCursor,  
 'sequence\_new\_btn\_name': "Nouveau fichier",  
 'sequence\_new\_btn\_default': False,  
 'sequence\_new\_btn\_cursor': QtCore.Qt.PointingHandCursor,  
 'save\_sequence\_title': "Sauvegarder le fichier généré",  
 'date\_format': "dd/MM/yy",  
  
 'list\_sorting\_order': QtCore.Qt.AscendingOrder,  
 'sequence\_speed\_lbl\_text': "Vitesse du robot : ",  
  
 'sequence\_origin\_lbl\_text': "Placer le robot sur l'origine\n"  
 "(axe z (en bleu) au niveau de\n"  
 "l'axe de rotation).",  
 'sequence\_origin\_lbl\_text\_start': "Placer le robot en position de départ.",  
 'start\_sequence\_text': "{comment} Position de depart : x = {x} mm, y = {y} mm, angle = {angle} degres\n",  
 'position\_text': "Position de depart :",  
 'sequence\_origin\_btn\_name': "Fait",  
 'sequence\_origin\_btn\_default': True,  
 'sequence\_origin\_btn\_cursor': QtCore.Qt.PointingHandCursor,  
  
 'track\_width': 20, # mm  
 'out\_limits': (2000, 1500) # mm avant de considerer hors du plateau  
 } # End self.main\_robot  
  
 self.second\_robot = { # Contient les donnees pour le robot secondaire  
 'type': "robot",  
 'name': "Robot secondaire",  
  
 'new\_message\_box\_type': QtWidgets.QMessageBox.Question,  
 'new\_message\_box\_title': "Question",  
 'new\_message\_box\_message': "Y a-t-il un second robot ?",  
 'new\_message\_box\_buttons': QtWidgets.QMessageBox.No | QtWidgets.QMessageBox.Yes,  
 'file\_dialog\_open\_title': "Choisir le robot secondaire",  
 'color': (40 / 255, 49 / 255, 255 / 255, 1), # de 0 a 1  
 'edge\_color': (11 / 255, 33 / 255, 180 / 255, 1), # de 0 a 1  
 'save\_extension': "CrubsRunner robot (\*.crr)",  
  
 'element\_name': "Robot secondaire",  
 'appearance\_translation\_x': 0,  
 'appearance\_translation\_y': 0,  
 'appearance\_translation\_z': 0,  
  
 'window\_title': "Propriétés du robot secondaire",  
 'remove\_message\_box\_message': "Etes-vous sur de vouloir supprimer le robot secondaire ?\n"  
 "Cette action est irréversible.",  
  
 'sequence\_dialog\_title': "Séquence du robot secondaire",  
 'sequence\_text': "{comment} Sequence gcrubs generee par CrubsRunner le {date} pour le robot secondaire.\n\n"  
 } # End self.second\_robot  
  
 self.view = { # Contient les donnees pour la gestion de la vue du widget central  
 'start\_view\_position\_rotation': QtGui.QQuaternion.fromEulerAngles(QtGui.QVector3D(-45, 0, 0)),  
 'start\_view\_position\_distance': 4000,  
 'start\_view\_position\_pos': QtGui.QVector3D(0, 0, 0),  
  
 'top\_view\_position\_elevation': 90,  
 'bottom\_view\_position\_elevation': -90,  
  
 'moving\_cursor': QtCore.Qt.ClosedHandCursor,  
 'orbit\_cursor': QtCore.Qt.DragMoveCursor,  
 'rotation\_view\_key': QtCore.Qt.MouseButton.LeftButton,  
 'moving\_view1': QtCore.Qt.KeyboardModifier.ControlModifier, # & ev.modifiers()  
 'moving\_view\_middle\_button': QtCore.Qt.MouseButton.MiddleButton,  
 'moving\_view\_middle\_button1': QtCore.Qt.KeyboardModifier.ControlModifier, # & ev.modifiers()  
 'moving\_view2': QtCore.Qt.MouseButton.RightButton,  
  
 'coord\_sys\_name': "Repere",  
 'coord\_sys\_file': "3d\_files/coord\_sys.stl",  
 'coord\_sys\_visible': True,  
 'coord\_sys\_x\_color': (1., 0., 0., 1.),  
 'coord\_sys\_y\_color': (0., 1., 0., 1.),  
 'coord\_sys\_z\_color': (0., 0., 1., 1.),  
  
 'min\_zoom': 3000,  
 'max\_zoom': 7000  
 } # End self.view  
  
 self.grid = { # Contient les donnees de la grille dans le widget central  
 'element\_name': "Grille",  
 'height': 2000,  
 'width': 3000,  
 'spacing\_height': 100,  
 'spacing\_width': 100,  
 'color': (255, 255, 255, 75), # de 0 a 255  
 'transparency': 75,  
 'visible': True,  
  
 'coord\_sys\_name': "Voir le repère",  
 'coord\_sys\_visible': True,  
  
 'see\_name': "Voir la grille",  
 'width\_name': "Largeur du carreau (mm) :",  
 'width\_min': 1,  
 'width\_max': 2000,  
  
 'height\_name': "Hauteur du carreau (mm) :",  
 'height\_min': 1,  
 'height\_max': 3000,  
  
 'color\_name': "Choisir la couleur",  
 'color\_default': False,  
 'color\_cursor': QtCore.Qt.PointingHandCursor,  
  
 'transparency\_name': "Transparence : ",  
  
 'close\_name': "Fermer",  
 'close\_default': True,  
 'close\_cursor': QtCore.Qt.PointingHandCursor,  
  
 'reset\_name': "Reset",  
 'reset\_default': False,  
 'reset\_cursor': QtCore.Qt.PointingHandCursor,  
  
 'window\_name': "Propriétés de la grille",  
 'window\_modal': True,  
  
 'color\_dialog\_name': "Choisir la couleur de la grille",  
  
 'group\_box\_name': "Grille"  
 } # End self.grid  
  
 self.gcrubs = { # Contient les donnees pour toute la partie fichier sequentiel  
 'extension': "Sequence CrubsRunner (\*.gcrubs)",  
 'edit\_action\_name': "Éditer les commandes gcrubs",  
 'edit\_action\_status\_tip': "Éditer les commandes gcrubs",  
  
 'edit\_window\_title': "Éditer les commandes gcrubs",  
 'description\_lbl\_text': "Remplir à gauche la description de ce que fait l'action et à droite "  
 "la séquence correspondante.\n"  
 "Mettre entre crochets ce qui peut varier. Les variables 'dist', 'angle' et 'temps'"  
 " sont connues et sont à utiliser\n"  
 "(les unites sont respectivement 'millimètre', 'degré' et 'seconde').",  
 'apply\_btn\_name': "Appliquer",  
 'apply\_btn\_default': True,  
 'apply\_btn\_cursor': QtCore.Qt.PointingHandCursor,  
 'cancel\_btn\_name': "Annuler",  
 'cancel\_btn\_default': False,  
 'cancel\_btn\_cursor': QtCore.Qt.PointingHandCursor,  
 'add\_btn\_name': "Ajouter une commande",  
 'add\_btn\_default': False,  
 'add\_btn\_cursor': QtCore.Qt.PointingHandCursor,  
  
 'key\_btn\_tip': "Choisir la touche associée",  
 'del\_btn\_tip': "Supprimer la commande",  
 'btn\_cursor': QtCore.Qt.PointingHandCursor,  
  
 'sa\_height': 430,  
 'sa\_width': 730,  
 'window\_modal': True,  
  
 'key\_dialog\_title': "Définir le mouvement associé",  
 'key\_lbl\_text': "Appuyer sur la touche correspondant à l'instruction\n"  
 "'{instruction}'\n"  
 "Touche : ",  
 'key\_close\_name': "Fermer",  
 'key\_close\_cursor': QtCore.Qt.PointingHandCursor,  
 'key\_close\_default': True,  
 } # End self.gcrubs  
  
 self.run = { # Contient les donnees pour la simulation  
 'run\_action\_name': "Lancer une simulation",  
 'run\_action\_shortcut': QtGui.QKeySequence(QtCore.Qt.CTRL | QtCore.Qt.Key\_R),  
 'run\_action\_tip': "Lancer une simulation",  
  
 'stop\_run\_action\_name': "Arrêter la simulation",  
 'stop\_run\_action\_shortcut': QtGui.QKeySequence(QtCore.Qt.CTRL | QtCore.Qt.Key\_R | QtCore.Qt.SHIFT),  
 'stop\_run\_action\_tip': "Arrêter la simulation",  
  
 'dialog\_title': "Choix du robot à simuler",  
 'dialog\_modal': True,  
 'main\_robot\_cb\_name': "Robot principal",  
 'main\_robot\_cb\_checked': False,  
 'second\_robot\_cb\_name': "Robot secondaire",  
 'second\_robot\_cb\_checked': False,  
 'cancel\_btn\_name': "Annuler",  
 'cancel\_btn\_default': False,  
 'ok\_btn\_name': "Valider",  
 'ok\_btn\_default': True,  
  
 'window\_title': "Simulation",  
 'cmd\_lbl\_main': "Commande robot principal : {cmd}",  
 'cmd\_lbl\_second': "Commande robot secondaire : {cmd}",  
 'time\_lbl': "Chrono : {time} s",  
 'theoretical\_time\_lbl': "Temps théorique : {time} s",  
 'theoretical\_time\_accuracy': 2, # Nombre de chiffres apres la virgule  
 'accuracy\_timer': None, # None pour ne pas voir les chiffres apres la virgule  
 'time\_before\_start': 2000, # ms  
 'timer\_refresh': 1000, # ms  
 'time\_for\_refresh\_estimation': 2, # s  
 # 15 parce que c'est ce qu'il fallait apres plusieurs tests, peut varier selon le pc  
 # mais aucun lien direct trouve avec le reste donc bon...  
 'added\_time\_refresh\_time': 15 # ms  
 } # End self.run  
  
 self.extensions = { # Contient toutes les extensions ouvrables par l'application  
 'project': ".crp",  
 'board': ".crb",  
 'robot': ".crr",  
 'sequence': ".gcrubs",  
 '3d\_file': (".stl", ".obj", ".3mf"),  
 'vinyl': (".png", ".pdf", ".jpg", ".jpeg")  
 } # End self.extensions  
  
 def get\_window(self, key: str):  
 *"""  
 Renvoie la donnee de la fenetre principale qui correspond a la cle.* ***:param*** *key: Cle pour obtenir la valeur correspondante* ***:return****: Valeur correspondant a la cle  
 """* return self.window.get(key)  
  
 def get\_board(self, key: str):  
 *"""  
 Renvoie la donnee du plateau qui correspond a la cle.* ***:param*** *key: Cle pour obtenir la valeur correspondante* ***:return****: Valeur correspondant a la cle  
 """* return self.board.get(key)  
  
 def get\_vinyl(self, key: str):  
 *"""  
 Renvoie la donnee du tapis qui correspond a la cle.* ***:param*** *key: Cle pour obtenir la valeur correspondante* ***:return****: Valeur correspondant a la cle  
 """* return self.vinyl.get(key)  
  
 def get\_main\_robot(self, key: str):  
 *"""  
 Renvoie la donnee du robot principal qui correspond a la cle.* ***:param*** *key: Cle pour obtenir la valeur correspondante* ***:return****: Valeur correspondant a la cle  
 """* return self.main\_robot.get(key)  
  
 def get\_second\_robot(self, key: str):  
 *"""  
 Renvoie la donnee du robot secondaire qui correspond a la cle.* ***:param*** *key: Cle pour obtenir la valeur correspondante* ***:return****: Valeur correspondant a la cle  
 """* return self.second\_robot.get(key)  
  
 def get\_view(self, key: str):  
 *"""  
 Renvoie la donnee de la gestion de la vue qui correspond a la cle.* ***:param*** *key: Cle pour obtenir la valeur correspondante* ***:return****: Valeur correspondant a la cle  
 """* return self.view.get(key)  
  
 def get\_grid(self, key: str):  
 *"""  
 Renvoie la donnee de la grille qui correspond a la cle.* ***:param*** *key: Cle pour obtenir la valeur correspondante* ***:return****: Valeur correspondant a la cle  
 """* return self.grid.get(key)  
  
 def get\_extension(self, key: str):  
 *"""  
 Renvoie l'extension qui correspond a la cle.  
 Si key == 'value': renvoie toutes les valeurs* ***:param*** *key: Cle pour obtenir la valeur correspondante* ***:return****: Valeur correspondant a la cle  
 """* if key == 'value':  
 return self.extensions.values() # Renvoie toutes les extensions  
 else:  
 return self.extensions.get(key)  
  
 def get\_gcrubs(self, key: str):  
 *"""  
 Renvoie la donnee de la partie sequentielle qui correspond a la cle.* ***:param*** *key: Cle pour obtenir la valeur correspondante* ***:return****: Valeur correspondant a la cle  
 """* return self.gcrubs.get(key)  
  
 def get\_run(self, key: str):  
 *"""  
 Renvoie la donnee de la partie simulation qui correspond a la cle.* ***:param*** *key: Cle pour obtenir la valeur correspondante* ***:return****: Valeur correspondant a la cle  
 """* return self.run.get(key)