

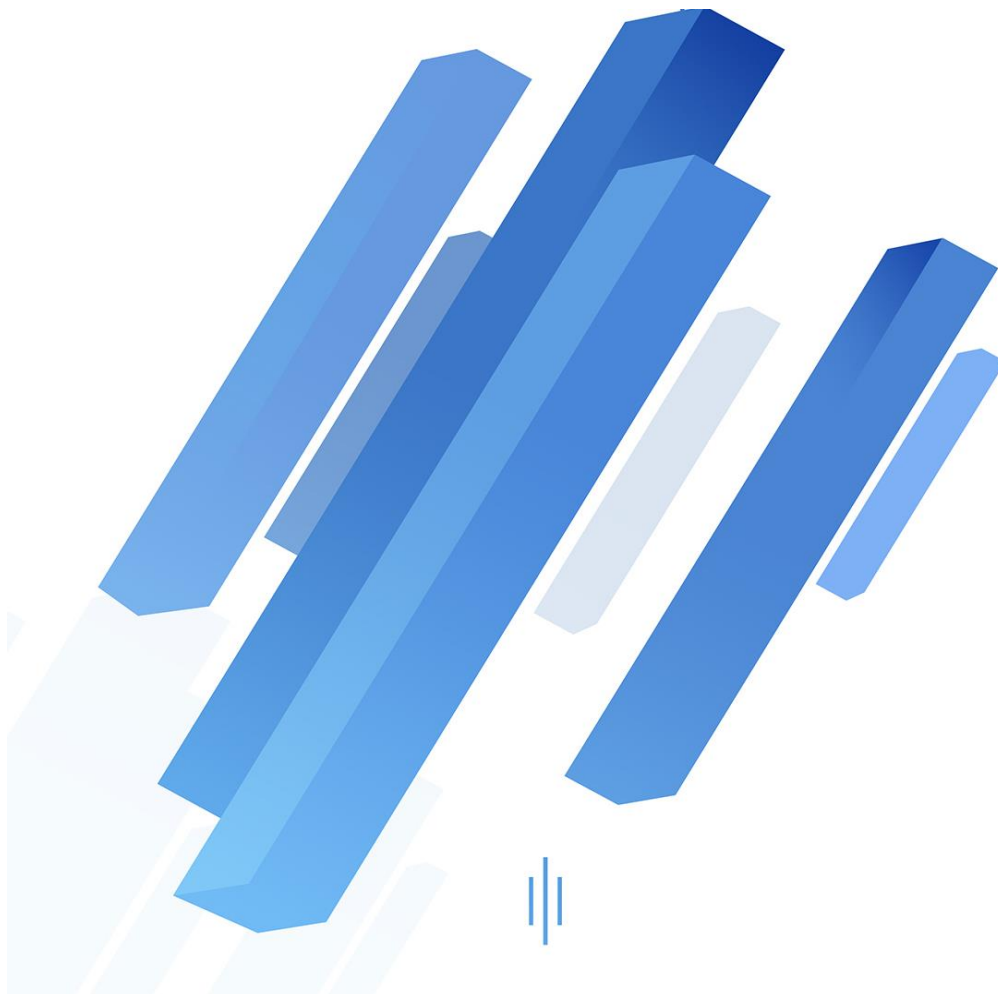
UTBM

SEMESTRE P2021

HM40

TP2 – Loi de Fitts

Enseignant : Jean-Charles CREPUT



GROUPE : Paul GOUBET / Aurélien GOUY / Cyril OBRECHT

1) Critiques du programme et évolutions proposées.

Après des tests et une analyse approfondie du code fourni, nous en avons déduit que certaines choses n'allaient pas.

Le premier point qui nous a sauté aux yeux est la grande quantité de warnings dans le code. En effet, un code propre ne peut pas contenir un nombre excessif de warnings sinon il risque de ne pas marcher comme prévu.

Le deuxième point correspond à des bugs graphiques lors de l'exécution de l'application. Par exemple, si l'on redimensionne la fenêtre de l'application en taille minimale avec la souris, lorsqu'on va lancer le programme en cliquant sur le bouton *Démarrer*, certains cercles où l'on doit cliquer vont apparaître en dehors de la zone visible à l'écran, ce qui va avoir pour conséquence de ne rien afficher dans le rectangle prévu à cet effet. Pour résoudre ce problème à l'instant T, il faut agrandir la fenêtre avec la souris, et donc perdre du temps sur le test en cours.

En troisième point, nous avons remarqué des possibilités assez intrigantes par rapport au réglage des valeurs sur l'écran d'accueil. En effet, il est possible de mettre le nombre de cibles à 0, chose assez incongrue puisque cela signifie que le logiciel nous affiche directement les résultats d'un test non passé, donc finalement des résultats vides. De plus, il est possible de régler la taille maximale des cibles inférieure à la taille minimale des cibles et inversement, ce qui est techniquement impossible et devrait être vérifié dans le code.

En quatrième point, nous avons remarqué que, dans le controller, il y a des bouts de code qui devraient normalement être présents dans le model, notamment la conception des ellipses, le calcul des points à afficher sur le graphique et le calcul de la prochaine cible. De plus, certaines devraient être dans la view, comme l'affichage des ellipses ou encore la fonction permettant de faire le graphique de résultats. Globalement, une grande majorité des fonctions présentes dans le controller ne devraient pas se situer là, nous avons donc dû tout réorganiser.

Finalement, pour résoudre tous ces problèmes (bien sûr en prenant en compte le fait que ce soit un projet pas très conséquent) nous avons décidé de repartir de zéro en créant un nouveau projet. En effet, nous avons pensé qu'il serait plus productif de tout recréer, comme cela nous avons l'organisation du projet en tête et pouvons réorganiser tranquillement le système MVC comme nous le souhaitons sans être perdu dans la réorganisation d'un projet inconnu. Bien évidemment, il ne s'agit pas de refaire tout le projet, nous avons donc copié-collé certains bouts de code qui marchaient très bien, notamment la plupart des éléments du controller qui avaient juste besoin d'être déplacés à leur place.

2) Dictionnaire des données et nomenclature.

Dictionnaire des classes

Nom	Description	Type	MVC
controller	Classe du controller	class	Controller
model	Classe du model	class	Model
GraphicWidget	Classe qui gère les clics de la souris et l'affichage des ellipses	class	View
MainWindow	Classe de la fenêtre principale du programme	class	View
Ui_MainWindow	Classe de l'interface graphique	class	View

Dictionnaire des méthodes

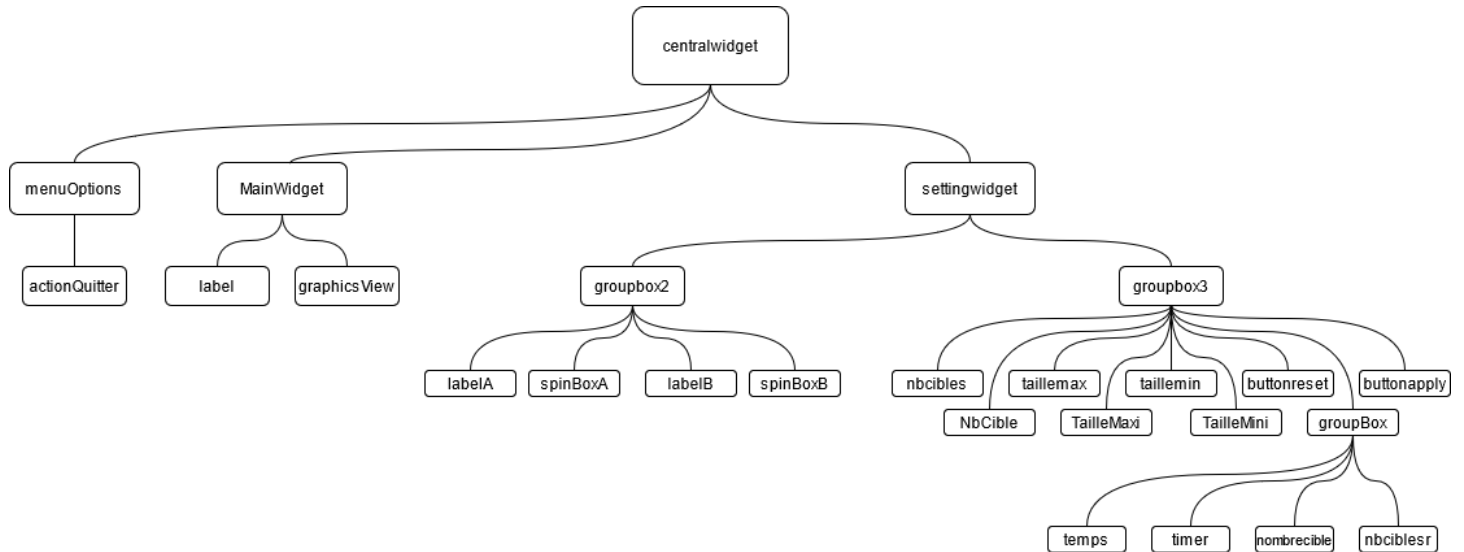
Nom	Encapsulation	Type de retour	Type des paramètres	MVC
onClick	public slots	void	int, int	Controller
onAChange	public slots	void	double	Controller
onBChange	public slots	void	double	Controller
onNbCibleChange	public slots	void	int	Controller
onMinSizeChange	public slots	void	int	Controller
onMaxSizeChange	public slots	void	int	Controller
nextTarget	private	void	void	Model
onCircleClick	public	void	QPointF	Model
resetTest	public	void	void	Model
init	public	void	void	Model
computeGraph	public	void	void	Model
getCirclesLeft	public	int	void	Model
isTestStarted	public	bool	void	Model
isInLastCircle	public	bool	QPointF	Model
mouseClicked	public signals	void	int, int	View
mousePressEvent	protected	void	QMouseEvent*	View
initWindows	public	void	controller*, model*	View
updateTestMsg	public	void	model*	View
getTestSceneHeight	public	int	void	View
getTestSceneWidth	public	int	void	View
drawCircle	public	void	QPoint, int	View
PrintResults	public	void	QList<QPointF>, QList<QPointF>	View
getNbCibleSpin	public	QSpinBox*	void	View
getTailleMinSpin	public	QSpinBox*	void	View
getTailleMaxSpin	public	QSpinBox*	void	View
getASpin	public	QSpinBox*	void	View
getBSpin	public	QSpinBox*	void	View
on_actionQuitter_triggered	private slots	void	void	View
setupUi	public	void	QMainWindow*	View
retranslateUi	public	void	QMainWindow*	View

Dictionnaire des attributs

Nom	Encapsulation	Type	MVC
pmodel	private	model*	Controller
view	private	MainWindow*	Controller
clickPoints	private	QList<QPointF>	Model
circleCenters	private	QList<QPointF>	Model
circleSizes	private	QList<int>	Model
times	private	QList<qint64>	Model
timer	private	QElapsedTimer*	Model
circlesLeft	private	int	Model
a	public	double	Model
b	public	double	Model
nbCircles	public	int	Model
minSize	public	int	Model
maxSize	public	int	Model
view	public	MainWindow*	Model
scene	public	QGraphicsScene*	View
ui	public	MainWindow*	View
plot	public	QChartView*	View
actionQuitter	public	QAction*	View
centralwidget	public	QWidget*	View
MainWidget	public	QWidget*	View
graphicsView	public	GraphicWidget*	View
settingwidget	public	QWidget*	View
groupBox_2	public	QGroupBox*	View
spinBoxA	public	QDoubleSpinBox*	View
spinBoxB	public	QDoubleSpinBox*	View
labelA	public	QLabel*	View
labelB	public	QLabel*	View
groupBox_3	public	QGroupBox*	View
NbCible	public	QSpinBox*	View
TailleMini	public	QSpinBox*	View
TailleMaxi	public	QSpinBox*	View
nbcibles	public	QLabel*	View
taillemin	public	QLabel*	View
taillemax	public	QLabel*	View
buttonapply	public	QPushButton*	View
buttonreset	public	QPushButton*	View
groupBox	public	QGroupBox*	View
timer	public	QTimeEdit*	View
temps	public	QLabel*	View
nombrecible	public	QLabel*	View
nbciblesr	public	QLabel*	View
menubar	public	QMenuBar*	View
menuOptions	public	QMenu*	View
statusbar	public	QStatusBar*	View

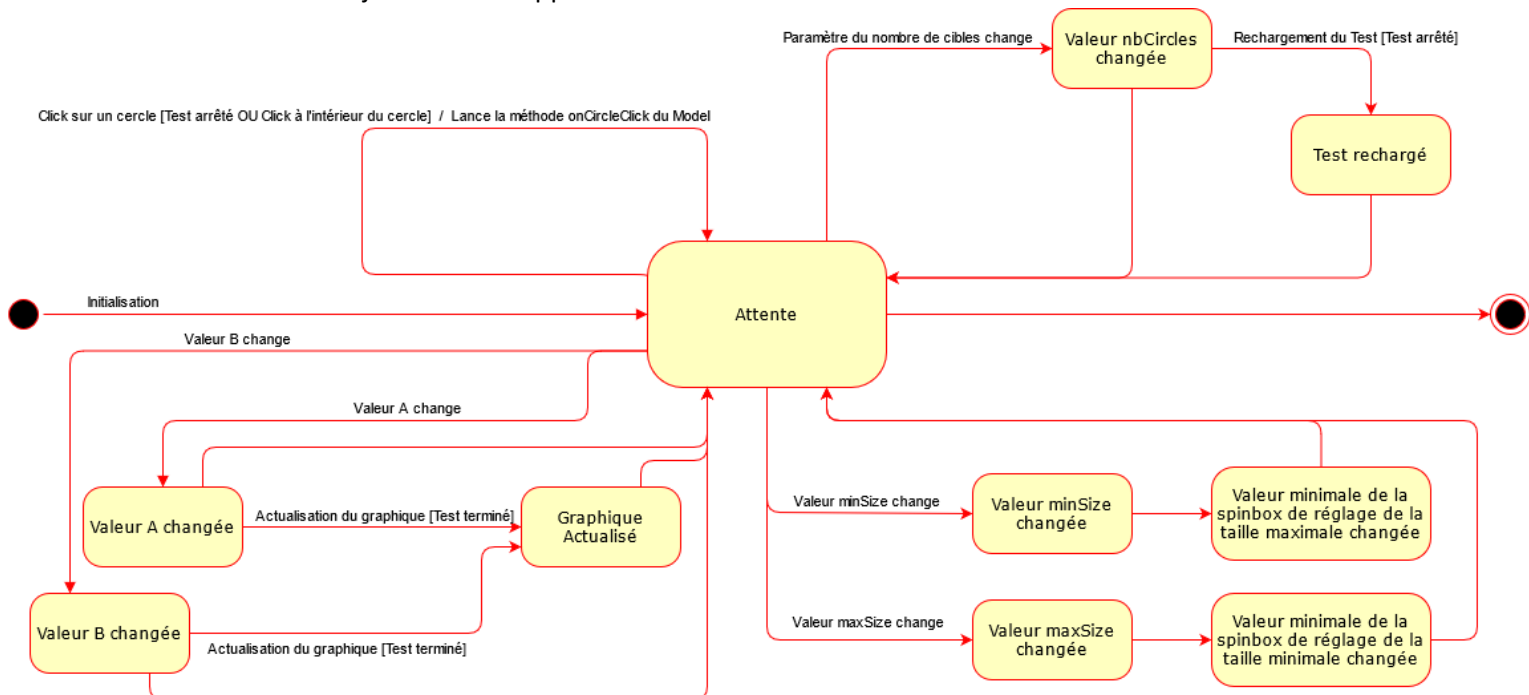
3) Diagramme de structure de l'interface proposée.

Voici ci-dessous le diagramme de structure de l'interface de notre application, avec les noms des objets identiques aux noms des objets de notre application.



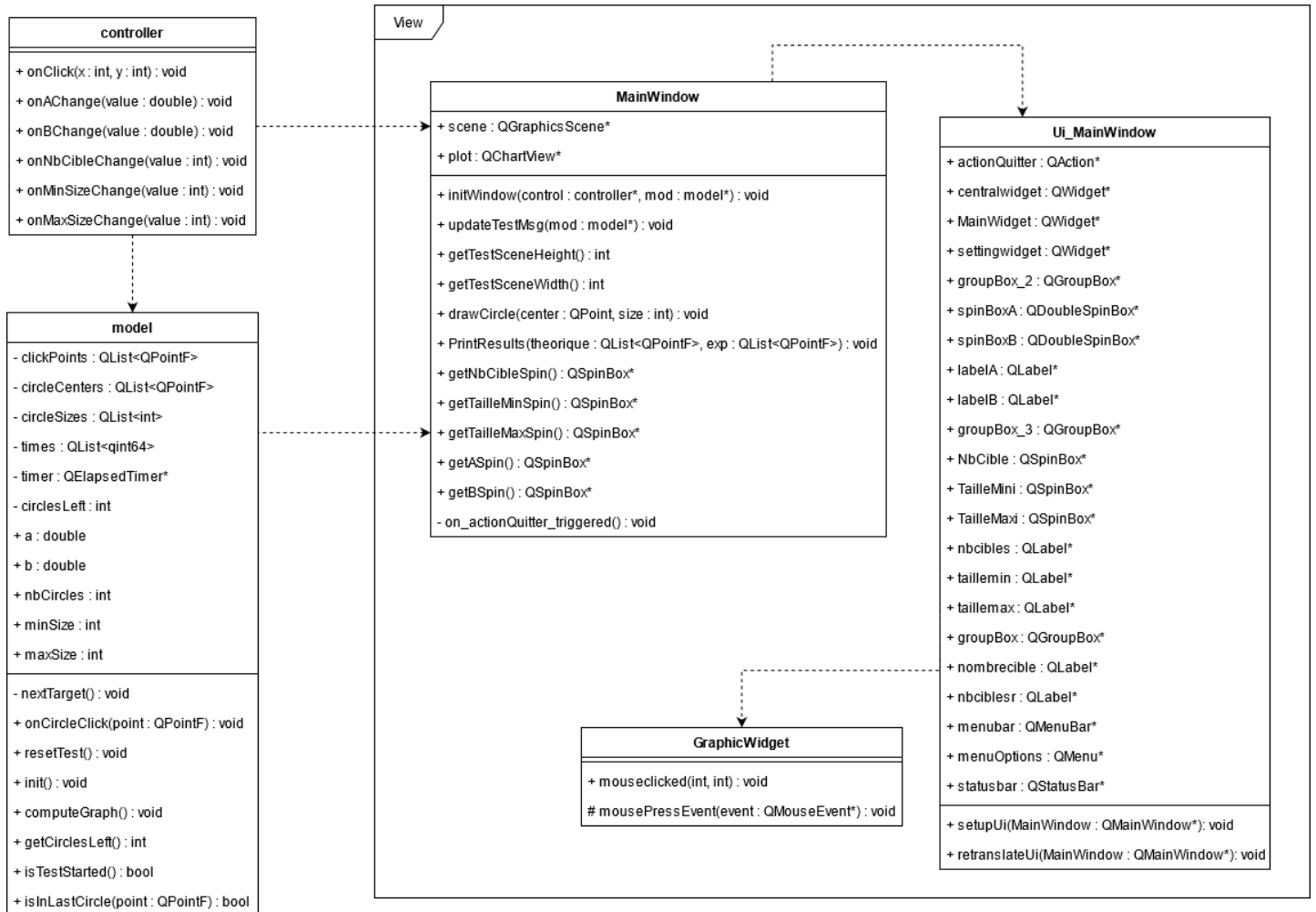
4) Modèle État/Transition du contrôle.

Voici ci-dessous le diagramme État/Transition de notre controller, avec les noms des objets identiques aux noms des objets de notre application.



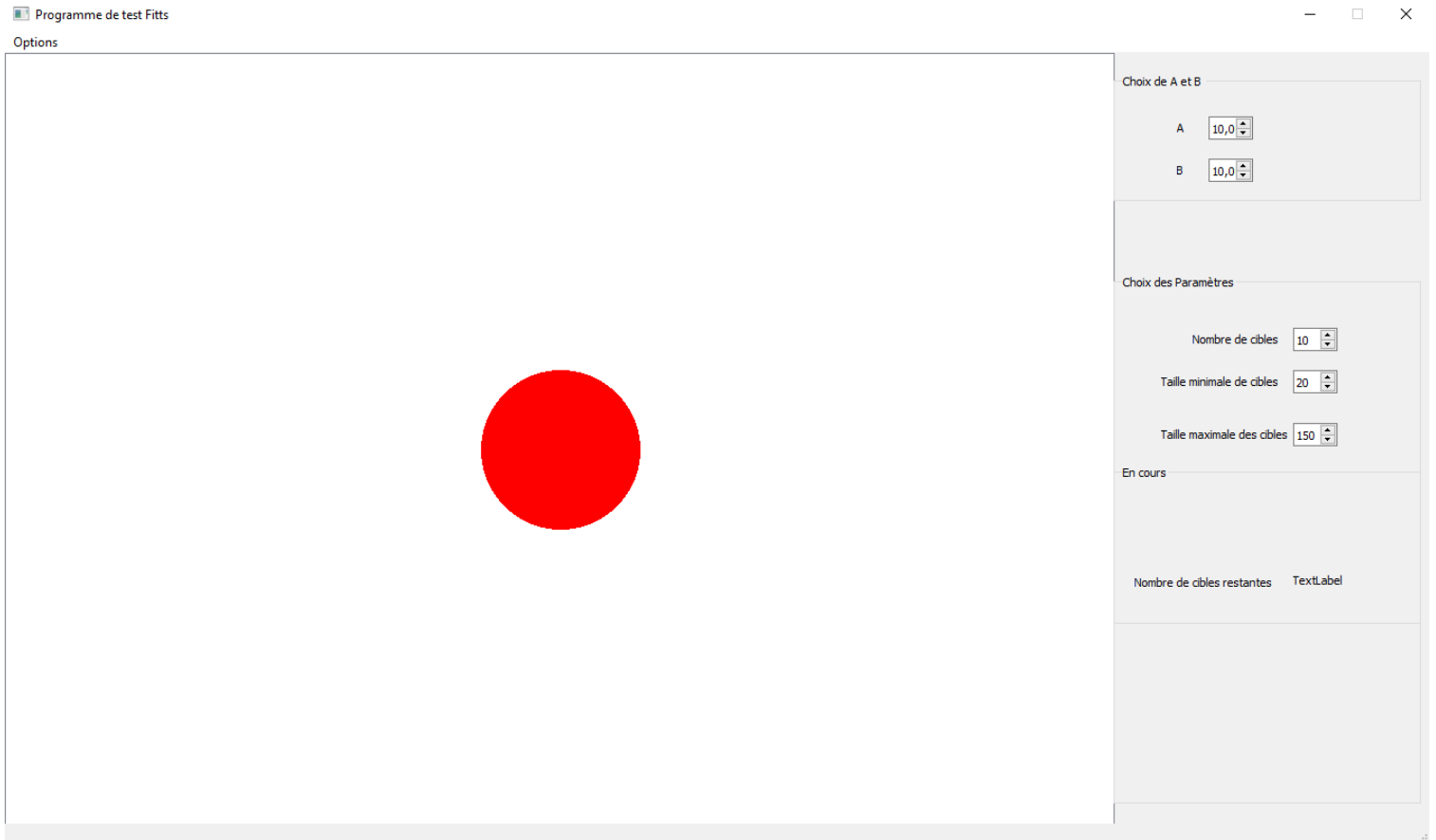
5) Modèle de classe incluant tous les objets Qt et les dépendances.

Voici ci-dessous le diagramme de classe de notre projet, avec les noms des objets identiques aux noms des objets de notre application.



6) Évaluation de la nouvelle interface.

Après avoir fini de reprendre le projet, nous pouvons passer à la phase de test de notre nouveau programme.



Premièrement, l'interface principale a changée, en effet nous tombons dorénavant directement sur un cadre qui va permettre de lancer le test directement. Nous avons choisi de laisser le paramétrage sur le côté droit du cadre de test car cela nous paraissait plus pertinent de pouvoir modifier le nombre de cibles, ainsi que leur taille minimale et maximale avant de commencer le test. En rapport avec ces paramètres, il n'est dorénavant plus possible de mettre une taille minimale de cible plus grande qu'une taille maximale et inversement. Les cibles ont désormais une taille minimale limite de 10 pixels de diamètre et une taille maximale limite de 1000 pixels de diamètre. De plus, le problème du réglage du nombre de cibles a également été résolu, le nombre de cibles minimum étant de 1 et maximum étant de 100.

Ensuite, malheureusement tout ce qui est en rapport avec un redimensionnement (redimensionnement de la fenêtre du programme, redimensionnement du graphique à la fin, etc...) ne sont pas parfaits. En effet, c'est la première fois que nous utilisons le logiciel Qt mais aussi que nous travaillons avec des interfaces graphiques, c'est donc dur à appréhender, qui plus est quand on ne sait pas d'où viennent les beugs à cause du manque d'expérience. Nous avons donc bloqué la possibilité de redimensionner la fenêtre du programme, cela évitera des beugs si l'utilisateur décide de faire n'importe quoi. De même, les boutons *Démarrer* et *Réinitialiser* ne sont plus présents. En effet, cela est normal pour le bouton *Démarrer* qui n'a plus lieu d'être car nous arrivons directement sur la fenêtre

du test, mais pour le bouton *Réinitialiser* nous avons eu des problèmes d'implémentation, nous avons donc préféré le supprimer tout simplement. Il faudra donc redémarrer le programme pour recommencer un nouveau test.

Enfin, nous avons essayé de coder le plus proprement possible pour obtenir le moins de warnings possible à la compilation. Nous en dénombrons actuellement 6 qui sont tous liés à l'utilisation d'une fonction considérée comme obsolète par Qt : *qrand*. Ce codage « propre » nous a également amené à réorganiser les méthodes et fonctions qui étaient auparavant quasiment toutes condensées dans le controller. Nous avons bien réparti notre code en suivant le principe MVC (Model, View, Controller).

Nous tenons à préciser que, malgré nos tests, certains bugs nous ont certainement échappé.