# Schéma global du projet : Tableau virtuel interactif

Bollini Kevin, Mélia Geoffrey, Pagès Julien, Saleil Baptiste

23 avril 2012

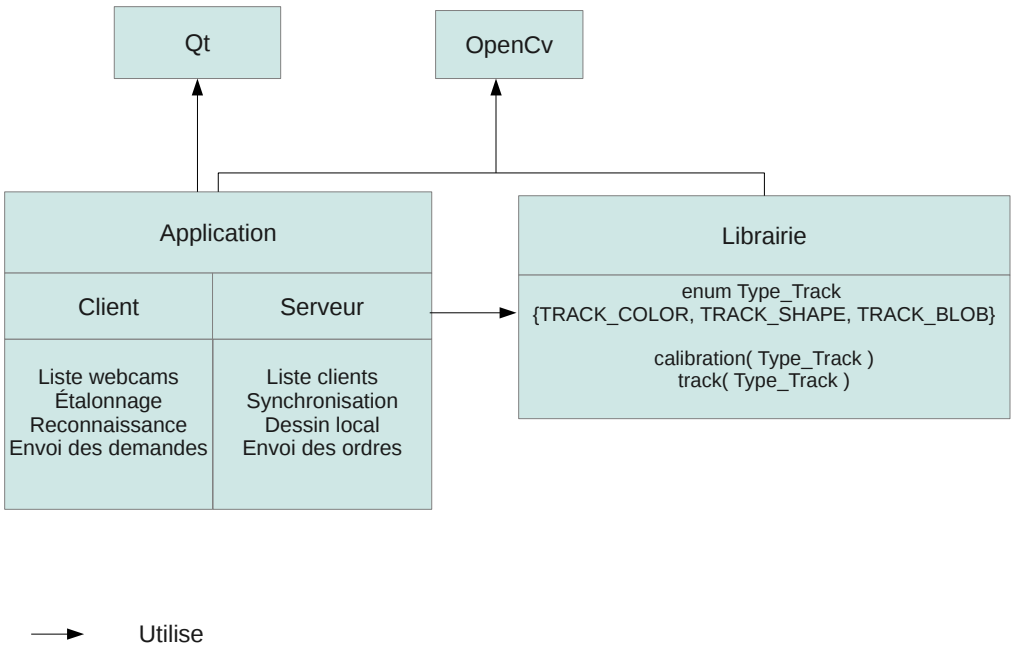
## 1 Introduction

Avant propos : Ce document présente notre projet et donne un aperçu du fonctionnement de notre application.

Le but de l'application est de simuler une écriture ou un dessin sur un tableau virtuel. Ceci en se basant sur une reconnaissance de mouvements via une webcam. Le projet comporte deux parties distinctes : la librairie -que nous nommeront libtrack- permettant de réaliser le suivi des objets dans le flux vidéo acquis par la webcam ; et l'application proposant une interface graphique permettant à un ou plusieurs utilisateurs de dessiner.

## 2 Schéma Global

Le schéma ci dessous présente de façon globale les différents modules pour une utilisation en réseau de l'application.

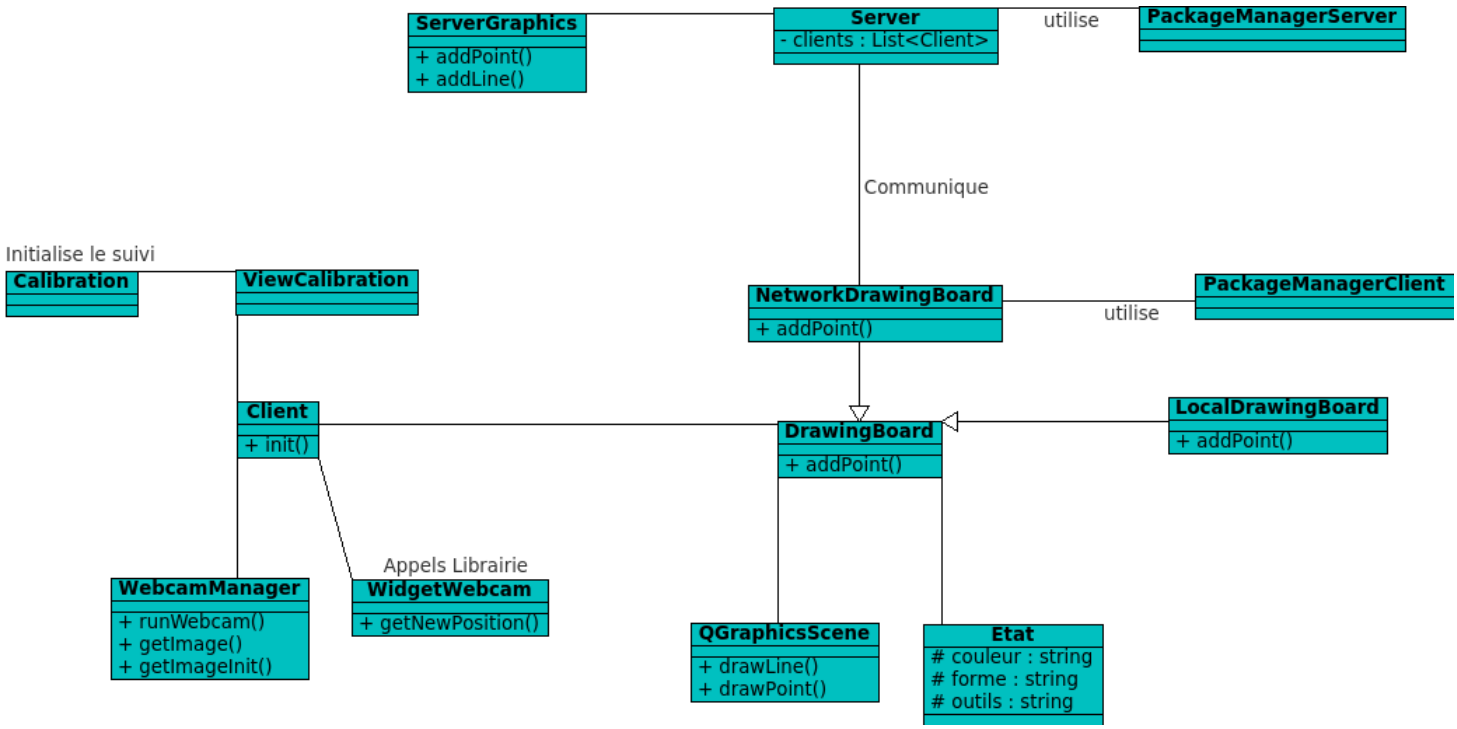



Le client et le serveur sont écrits à l'aide de la librairie Qt. Le client fait appel à la librairie libtrack pour reconnaître les mouvements.
Cette dernière utilise OpenCV pour le traitement de l'image et se veut réutilisable d pour'autres projets. Elle contient principalement une fonction d'étalonnage qui permet de signifier l'objet à suivre, ainsi que le type de suivi (Couleur, forme, avec cvBlob). La librairie possède ensuite un seul point d'entrée permettant d'effectuer le suivi.

# 3 Architecture de l'application

L'application est conçue entièrement par dessus la librairie en essayant de minimimer le nombre de fonctions appelées. L'étalonnage du suivi de l'objet est effectué dans la classe *Calibration*, ensuite les appels pour suivre les mouvements sont faits via une seule fonction dans la classe *WidgetWebcam* qui transmets ses mouvements au *Client* (classe principale).

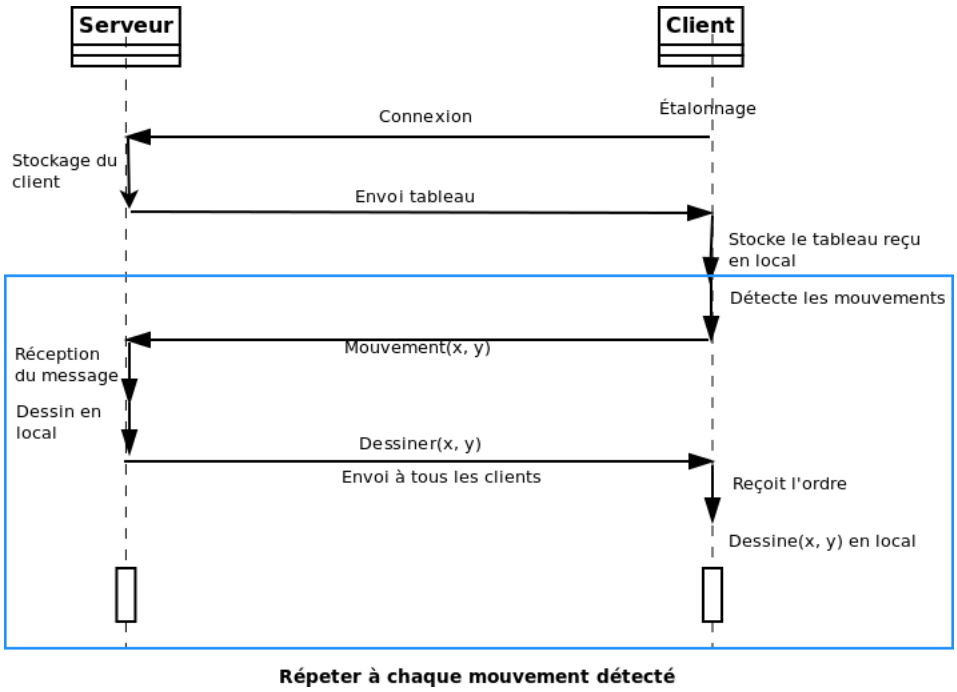
De plus l'application est conçue pour fonctionner de manière similaire pour une utilisation en réseau, ou en local, les classes représentant le tableau en local ou en réseau ont donc la même apparence (mêmes points d'entrées) vu de l'extérieur.




# 4 Schéma lors d'une utilisation en réseau

Ce schéma présente un déroulement des opérations lors d'une connexion et d'un dessin en utilisant les fonctionnalités réseaux.

- Le client se connecte
- Le client effectue son étalonnage
- Le serveur l'enregistre, et lui envoie le tableau courant (vide ou non)
- Le client stocke le tableau et commence à détecter un mouvement
- Il envoie le mouvement détecté
- Le serveur reçoit le mouvement, dessine en local et renvoie à tous les clients l'ordre de dessin
- Le client reçoit l'ordre et dessine en local




# 5 Principe de la librairie

Comme expliqué précédemment, la librairie libtrack doit permettre de réaliser le suivi d'un objet (au sens large du terme) en temps réel.
Pour cela, elle propose une structure contenant toutes les informations nécessaires à la reconnaissance de l'objet dans l'image. C'est la struct Cursor. Celle-ci doit être utilisée par l'utilisateur dans les deux fonctionnalités principales de la librairie :

- la méthode de calibration (ou étalonnage), qui permet d'initialiser la structure de suivi.
- la méthode de suivi à proprement parler, qui met à jour les données de la structure, ce qui permet à l'utilisateur de récupérer les données actualisée.

Il faut commencer par étalonner suivant le type de suivi voulu. L'étalonnage permet de déterminer l'objet que l'on souhaite suivre afin de dessiner. Pour cela, l'utilisateur doit fournir deux points qui permettront de définir un cadre autour de l'objet.

## 5.1 Différentes méthodes de suivi

La librairie propose différentes manières pour suivre l'objet désiré. L'utilisateur peut choisir l'une ou l'autre en le spécifiant simplement à l'aide de l'énumération suivante :
TYPE_TRACK { TRACK_COLOR, TRACK_SHAPE, TRACK_BLOB} ;

Il est donc possible de choisir entre 3 types de suivi :

- **par forme** : Ici, c'est la forme de l'objet qui est utilisée. On recherche une correspondance forte entre un pattern de l'objet recherché, et une éventuelle zone solution dans le flux vidéo.
  Pour cela, on pose des calques du 'modèle' à intervalle régulier sur l'image entrante, et on compare le taux de ressemblance afin d'évaluer la correspondance.

- **par couleur** : On s'intéresse ici aux valeurs des couleurs (des pixels) de l'objet plutôt qu'à leur disposition spatiale. Des méthodes d'échantillonage et de seuillage permettent de proposer un résultat plus fin et plus souple qu'une moyenne de couleur seule. Cette dernière ne peut en effet supporter des variations de teinte fortes au sein de l'objet.

- **par blob** (binary long object) : On fait ici appel à une librairie externe construite avec OpenCv : CvBlob. On utilise les composantes connexes pour déterminer quel ensemble de pixels forme ou non un "objet blob". Il faut ensuite mettre en correspondance les blobs trouvés dans le flux vidéo avec l'objet issu de notre initialisation.