SmartProjector

BRANGER Mathias, HABLOT Jules

5 avril 2016

Table des matières

1 Introduction

De nos jours, les présentations sur vidéo-projecteur sont de plus de plus courantes. Néanmoins, la technologie sans fils dans ce domaine n'est pas encore très développée. Par exemple dans notre école, Polytech Grenoble, les vidéo-projecteurs sont présents dans toutes les salles mais il faut encore s'y connecter par cable VGA. Ce type de port, était très répandu sur les anciens et volumineux ordinateurs portables mais désormais il est trop gros pour être mis sur nos ultras-books. Il nous font donc acheter des adaptateurs et ne pas les oublier. De plus, nous sommes parfois confrontés à des problèmes de connectiques : les cables VGA sont mal branchés à la prise murale, les embouts sont détruits par des élèves malveillants...

2 Notre projet

Nous apportons la solution pour régler tous ces problèmes. Nous proposons une application qui sera disponible sur toutes les plateformes et qui permettra une connection en wifi entre les ordinateurs portables des utilisateurs et le vidéo-projecteur. Les utilisateurs pourront se connecter seul ou à plusieurs en même temps sur le vidéo-projecteur. Ils peuvent être des simples élèves qui partagerons l'écran, mais aussi un professeur qui aura accès à plus de fonctionnalité comme de choisir l'écran à afficher sur le vidéo-projecteur.

3 Technologies utilisées

Nous avons commencé par rechercher une solution déjà faite sur internet pour ne pas reproduire un projet déjà abouti et fonctionnel.

Nous nous sommes donc interrogé sur les technologies qui existent et qui peuvent nous être utiles. Nous avons aussi demandé de l'aide à des professeurs pour voir ce qu'ils connaissaient pour qu'ils puissent nous orienter vers certains type de technologies. Ainsi nous avons entendu parler de **WebRTC**, **HTML5**, **GStreamer** concernant les frameworks de gestions de flux.

Après analyse de **HTML5** et de **WebRTC** nous avons conclu que cela ne répondait pas à nos attentes, en effet nous devions passer par un navigateur

pour partager l'écran du navigateur or les applications de présentations de diaporama ne sont pas encore intégrées dans tous les navigateurs, et l'on voulait aussi projeter autre chose que des diaporamas.

Nous devrons aussi choisir une librairie graphique compatible avec toutes les plateformes et qui nous convienne.

3.1 GStreamer

Nous nous sommes intéressés par le biais de Jérôme Maisonasse au framework GStreamer. Il s'agit d'une librairie C/C++ pour faire du transfert de flux. Sa prise en main était simple : nous pouvions tester les différentes fonctionnalité en ligne de commandes. Nous nous sommes donc très vite décider à prendre ce framework car il semblait correspondre à toute nos attentes et nous arrivons à faire beaucoup de chose en ligne de commandes : splitter un écran en 4 et recevoir 4 flux en simultané, donc certains provenant du réseau. Néanmoins nous avons était confronté à un problème : la gestion dynamique des flux. En effet lors de la connection d'un nouveau client ou la déconnection, il fallait changer d'affichage et donc de ligne de commande. Nous avons donc décider de passer par un langage permettant une gestion dynamique des clients, orienté objet pour une simplicité et utilisé par le reste de la communauté de GStreamer pour avoir des réponses à nos questions. Nous sommes donc passer sur le développement d'un projet en C++ avec l'API de GStramer.

3.2 Qt5

Maintenant que nous avions défini notre langage, nous pouvions regardé plus en profondeur les différentes librairies graphiques. Nous avons commencé par regarder **GTK**. Mais à cause de problème d'installation et de compatibilité avec Mac OSX et un manque d'utilisation par la communauté de GStreamer, nous avons du changer.

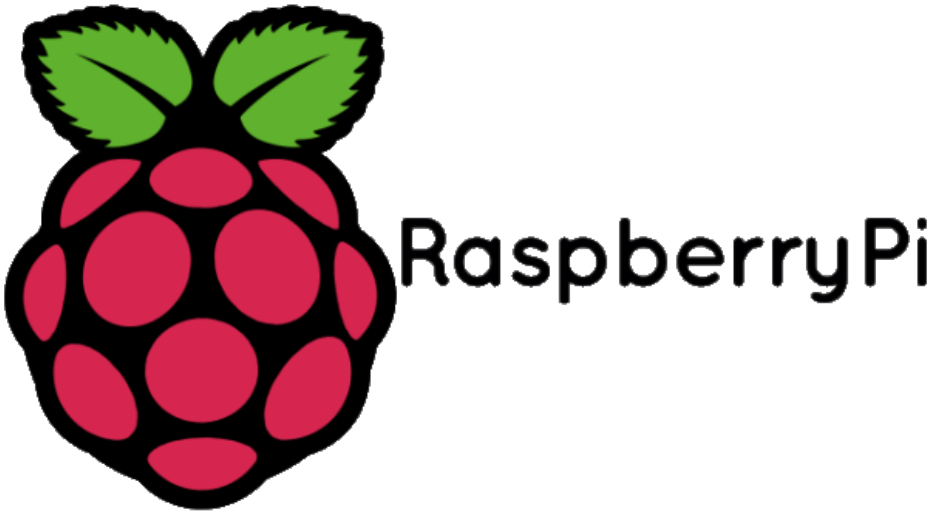
Nous avons cherché parmi les multiples exemples que nous avions trouvé sur le net durant notre veille technologique et il est remonté un utilisation

fréquente de **Qt** comme librairie graphique. Nous avons aussi fait la découverte d'une IDE, **Qt-Creator**, pour cette librairie graphique, ce qui nous facilitait la tache pour l'interface graphique. Restait néanmoins à intégrer les deux frameworks en même temps.

Cette partie n'est toujours pas fonctionnelle sous Mac OSX à cause d'un problème d'inclusion de GStreamer. Malgré des questions à la communauté et une recherche approfondie sur internet, nous n'arrivons toujours pas à compiler sous **Qt-Creator**, ni même en ligne de commande.

3.3 Stick

À nos premiers essais nous étions sur nos machines pour faire le server et le client de notre application. Par la suite nous avons voulu changer cela pour avec quelque chose qui ressemble plus à un produit fini. Nous sommes donc allé à la FabLab pour demander une carte capable de faire ce que nous voulions. M. Maisonasse nous a prêté un **Raspberry Pi 2** avec un dongle wifi (il ne permet que de recevoir et pas d'émettre).



Notre première idée a été de faire de la cross-compilation pour compiler notre programme pour Raspberry sous Ubuntu. Mais il a été trop difficile d'installer le nécessaire et nous sommes passés à l'idée de compiler directement le projet sur le raspberry. Ceci a impliqué encore une grande phase d'installation : Qt5, Qt-Creator et GStreamer. Malgré quelques difficultés nous avons finis par réussir cette série d'installation et même la compilation. Nous voilà donc avec une application serveur tournant sur le raspberry. Lors du lancement d'un client, le raspberry affiche l'écran reçu et lorsque plusieurs clients se connectent, l'affiche est diviser pour laisser la place à tous les utilisateurs. Nous sommes pour le moment obliger de passer par un téléphone

portable faisant office de routeur car **Wifi Campus** ne nous autorise pas à l'utiliser comme routeur.

4 Avancement actuel

Nous allons récapituler ici ce qui marche pour le moment. Nous avons deux projets Qt-Creator distincts : un client et un serveur. Tous deux compilent avec GStreamer 1.0 et Qt 5, sous Ubuntu et Raspberry.

Notre serveur écoute sur un port et lorsqu'il reçoit un message TCP sur son port d'écoute il ouvre une nouvelle socket sur un autre port pour commencer la communication avec le client et l'échange de flux. Lorsque plusieurs clients se connectent en même temps le serveur est capable de séparer la fenêtre pour afficher le nouveau client. Néanmoins ceci n'est pas fait dynamiquement dans le sens ou l'on doit relancé la fenêtre dans son ensemble.

Côté client, lors du lancement il vous propose de se connecter à un serveur et ensuite envoie le flux de votre écran. De plus nous avons mis en place en tchat distribué qui à la base servait d'exemple mais qui reste pour le moment en attendant les autres fonctionnalités. Notre interface graphique se limite au minimum car ce n'était pas dans nos objectifs de départ mais une séparation MCV a été faite pour permettre une amélioration facile ultérieurement.

5 Évolution

Il nous reste à implémenter différentes fonctionnalités pour avoir quelque chose de complet et de fonctionnel. À titre d'exemple, la différenciation entre un élève et un professeur n'est pas faite, il faudrait avoir une base de donnée sur le serveur pour que lors de la connection, le serveur puisse indiquer au client de quel type il est. Une fois ceci réaliser nous pourront implémenter les différentes fonctionnalités qui nécessitent la distinction entre les deux types d'utilisateurs.

La fonctionnalité la plus facile à rajouter et le zoom/dezoom sur un écran, il faudra par ailleurs regardé pour trouver une meilleure répartition des écrans lors de l'ajout et de la suppression d'un client.

6 Conclusion

Il nous reste beaucoup de travail à faire pour réaliser tous les objectifs initiaux, mais la base a été posée et reste fonctionnelle dans la plus part du temps.