RAGo

Rapport PAPPL 2013 - 2014

Nicolas DAVID & Sylvain PALOMINOS

Encadré par Myriam Servières & Jean-Marie Normand

Contenu

Préambule

Dans le cadre de notre troisième et dernière année d'étude en cursus ingénieur à l'École Centrale de Nantes, nous avons réalisé à 2 un projet de groupe encadré par Myriam Servières et Jean-Marie Normand.

Ce projet portait sur le développement d'une **application de réalité augmenté** permettant de **jouer au Go**.

Ce rapport présente l'ensemble de notre travail : le cahier des charges, la conception de l'application, et le fonctionnement. Il sera également question des éventuelles pistes d'améliorations possibles.

Les deux dépôts utiles pour ce projet sont disponibles sur github aux adresses suivantes :
https://github.com/SPalominos/qgo
https://github.com/SPalominos/RAGo

Une vidéo de démonstration non commentée est présentée ici :
https://www.youtube.com/watch?v=_OM6P49FXWg

I – Le contexte

I-1 – Lancement du projet

Sujet :
Le jeu de Go est un jeu de stratégie chinoise qui se joue sur un plateau de 19x19 cases avec des pions noirs et blancs posés sur les intersections. Le principe de cette application est de permettre de jouer avec un joueur distant à l'aide d'une interface tangible. C'est à dire, un des joueurs jouera directement sur un plateau avec une des deux couleurs et sera filmé par une caméra. Les coups de son adversaire seront directement vidéo-projetés sur ce même plateau. Cette application pourra être interfacée avec un logiciel de jeu de go en ligne du type de KGS.
Cette application peut aussi être développée pour un jeu de dame voire un jeu d'échecs.

Cahier des charges :
La rédaction par l'équipe projet du cahier des charges, validé par Myriam Servières et Jean-Marie Normand a dégagé les grands aspects principaux suivants :

- Calibration caméra projecteur
- Détection des pièces et de la fin du tour
- Interfaçage avec un client de Go

Le cahier des charges en version intégrale est disponible en **annexe 1**.

I-2- la Réalité Augmentée

La réalité augmentée (à ne pas confondre avec la réalité virtuelle) consiste en des systèmes informatiques qui rendent possible la superposition d'un modèle virtuel 3D ou 2D à la perception que nous avons naturellement de la réalité et ceci en temps réel.

La RA a donc 2 apports principaux :

- Améliorer les interactions avec objets usuels
- Aider à leur utilisation

Pour notre projet, la réalité augmentée passe par un système projecteur-caméra permettant donc une double interaction :

- D'une part par interaction avec l'objet via détection de mouvement
- D'autre part par lecture d'informations supplémentaires non propre à l'objet qui seront projetées sur lui

I-3 – le jeu de Go

Ici, la connaissance des règles précises du jeu n'était pas nécessaire pour réaliser le projet, puisque les règles sont directement implémentées par le client que nous utilisons. Toutefois, les principes élémentaires de jeu sont tout de même à connaitre pour s'assurer que les règles puissent bien s'appliquer à notre modèle.

Ainsi, le GO se joue sur une grille dont la taille dépend du modèle de goban choisi. De plus, les pierres se positionnent aux intersections. Enfin, des pierres qui sont dites « prises » doivent être retirées du plateau. La notion de prise est traduite dans notre application par le fait que le déplacement d'une pierre à sa propre position est non valide. Il convient alors de retirer la dite pierre du goban.

II – Présentation du Projet

A partir du cahier des charges établi, nous avons dégagé une organisation pour l'application RAGo. A termes, celle-ci devra proposer une interface graphique et un tutorial plus complet permettant aux utilisateurs une meilleur appréhension de notre application. Faute de temps, ces points n'ont toutefois pas pu être développés comme nous le souhaitions. Nous reviendrons donc en détail sur ces éléments dans la partie IV.

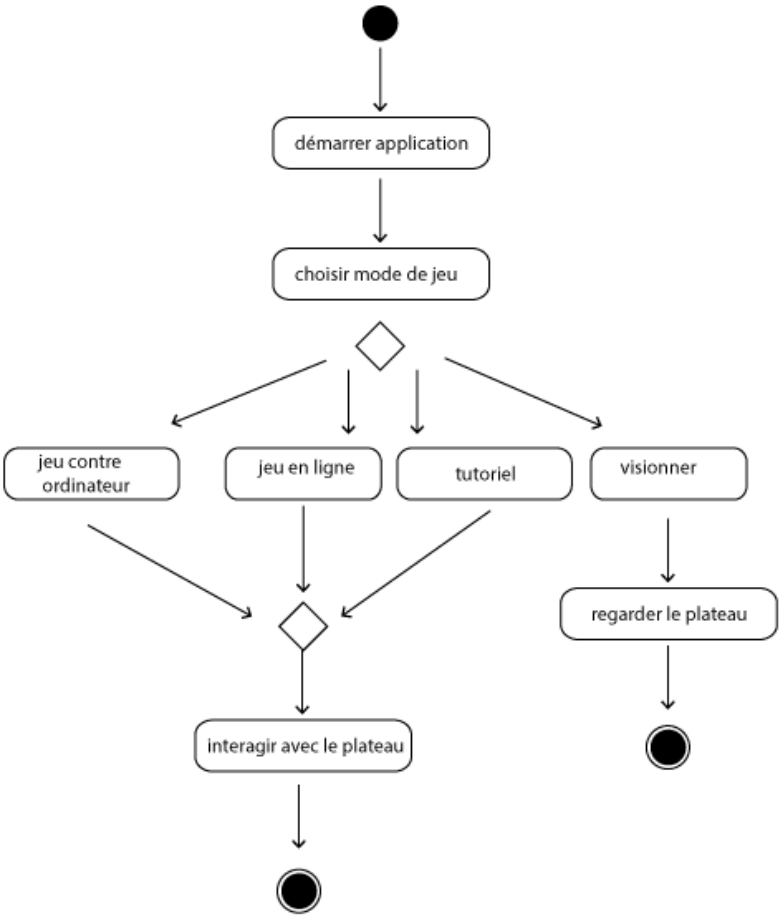


Diagramme d'activité du fonctionnement général de RAGO

Les 4 fonctionnalités sont présentées plus en détail dans le guide utilisateur, voir l'**annexe 3**.

II-1 – Environnement de travail

Ce projet a été réalisé sous environnement UNIX. Le développement a été effectué en C++ avec l'IDE code::blocks. Le guide développeur a été réalisé grâce à la documentation du code générée par Doxygen.

II – 2 – Installation et prise en main open CV

OpenCV est une bibliothèque graphique libre (sous licence BSD), initialement développée par Intel, spécialisée dans le traitement d'images en temps réel.

Pour l'installer, tous les fichiers sont disponibles ici :
http://opencv.org/downloads.html
Ainsi qu'un tutorial d'installation pour linux à l'adresse suivante :
http://docs.opencv.org/doc/tutorials/introduction/linux_install/linux_install.html#linux-installation

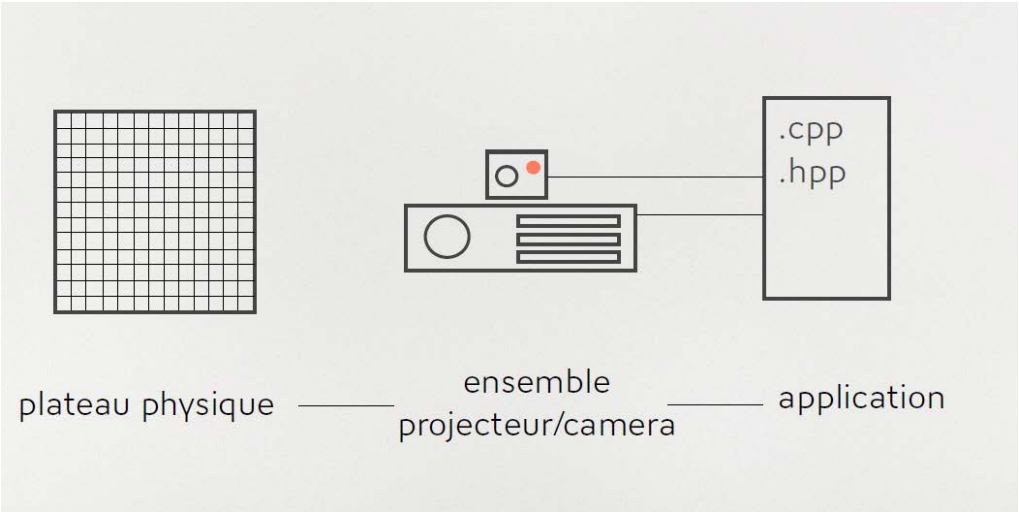
Une fois compilée, lors de la création d'un projet sous code blocks, il faudra alors ajouter les librairies à votre projet.
Pour cela, clic droit sur le projet, « build options », onglet « linker settings », puis ajouter toutes les librairies de openCV.

Le site d'OpenCV présente par ailleurs un grand nombre de tutoriaux pour l'utilisation des fonctions qui sont disponibles dans cette bibliothèque.

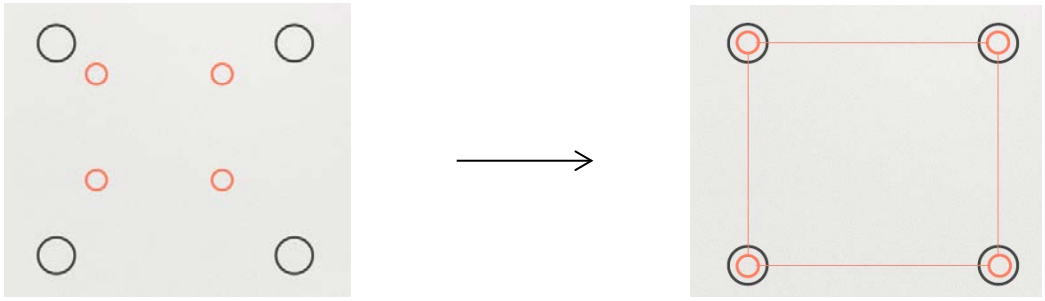
Dans notre cas, les fonctions utilisées sont celle de traitement d'image, pour améliorer la détection des pierres par l'utilisation de fonction comme l'érosion ou le flou.
Nous utilisons aussi des fonctions de calcul matriciel pour permettre le passage entre les différents repères comme nous le présentons dans la partie suivante.

II – 3 – Ensemble Camera Projecteur (calibration)

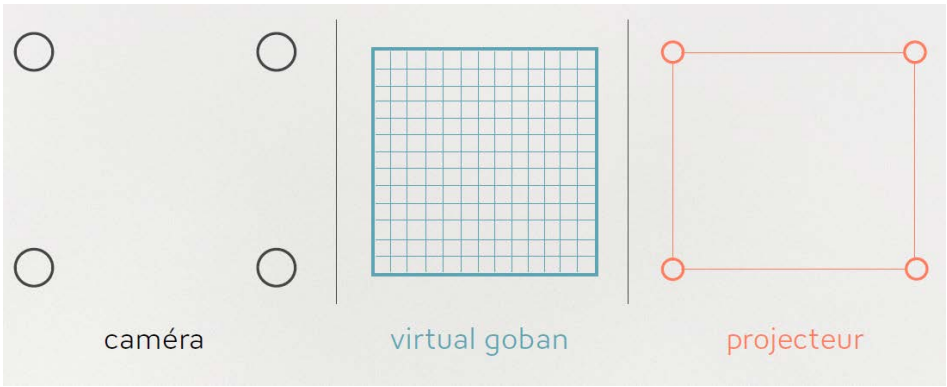
Le premier point de l'application consistait à calibrer le projecteur & la camera pour faire coïncider leurs repères.
En effet, à partir de l'application RAGo, nous récupérons des positions par l'intermédiaire de la caméra (ayant son propre repère) et projetons des éléments grâce au projecteur (qui a son propre repère).



Cette calibration se réalise de manière automatique en proposant de détecter des pièces placées aux 4 coins du goban, puis de les projeter et de déplacer ces projections (en orange sur le schéma ci-dessous) jusqu'à les faire coïncider avec les pierres initialement détectées dont la position a été gardée en référence (en noir ici).



Un autre repère a de plus été introduit dans le code : le **VirtualGoban**. C'est lui qui représente le jeu en lui-même et qui communiquera les positions au client. La fonction **findHomography** nous permet alors de trouver les 4 homographies pour changer de repère (ces matrices s'utilisent avec la fonction **warpPerspectiv**)
A noter que les textes ou les ronds sont dessinés sur le virtualGoban et que toute l'image subit alors l'homographie pour s'assurer que, malgré les déformations dues à la projection, l'information reste lisible en « vue aérienne ».



Les 3 repères utilisés dans RAGo

Ce qu'on peut noter sur cette fonctionnalité c'est que seuls les coins sont utiles pour la calibration, ainsi, on peut très bien imaginer à fortiori jouer sans goban !

II - 4 – Fonctions de détection

La calibration effectuée, il faut maintenant pouvoir interagir avec le plateau. Pour cela, des fonctions de différences entre 2 images acquises par la caméra vont permettre :

- De détecter la nouvelle pièce posée
- De signifier que le joueur à fini son tour par l'utilisation d'une zone de détection, appelée « horloge »

Ainsi, avant le début du tour humain, on capture une image qui correspond à la situation initiale. On détecte si le signal de fin de tour est envoyé. Lorsque celui-ci est envoyé, on recapture une image pour effectuer la différence et trouver la nouvelle pièce.

Ici, on voit évidemment apparaître 2 problèmes :
- Comment signifier la fin du tour ?
- Comment ne détecter la différence que sur le plateau ?

Au final, ces 2 problèmes sont résolus par l'emploi de masque binaire sur nos images. Les masques binaires permettent en effet de masquer une partie d'une image (partie en noire) pour ne considérer que la zone blanche.

Ainsi, connaissant les coins du plateau, on peut aisément appliquer un masque sur notre plateau pour comparer au début et à la fin du tour les pions qui s'y trouvent.

Pour la zone de détection, il nous suffit de projeter un cercle, sur lequel nous faisons une détection de différence en continue. Lorsque l'utilisateur y présente sa main, un compteur s'incrémente et ce tant que sa main reste dans la zone. Lorsque ce compteur atteint la valeur demandée, le tour est terminé. Si l'utilisateur enlève sa main, le compteur retombe à 0.

II- 5 – Interfaçage avec le client (Qgo) avec Qt

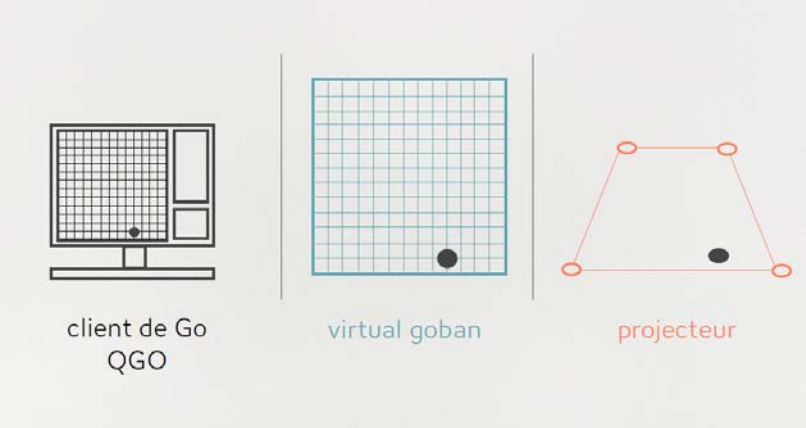
Il reste maintenant à présenter les fonctionnalités permettant d'acquérir & traiter l'information en vue de la projeter sur notre goban. Pour cela, RAGo communique avec le client de GO QGo.

Nous acquérons alors les informations relatives aux positions des pierres à projetées (adverses ou partie regardée) ainsi que les temps d'horloge mis à jour. Ces informations sont alors positionnées sur le virtualGoban puis projetées.

Les informations qui sont récupérées de QGo sont :
- La position des pièces jouées en ligne/par l'ordinateur
- Les temps d'horloge
- Les pièces capturées

Les informations envoyées à QGo sont :

- La position des pièces acquises par la caméra
- Le signal de fin de tour



En vue de réaliser cette communication, nous implémentons dans le client un serveur de sockets et par RAGo un client de sockets, que nous faisons communiquer par le port 5001.

L'avantage de cette technologie est :

- la simplicité de mise en œuvre
- la possibilité de faire communiquer des applications totalement indépendantes
- la possibilité de réaliser une communication en locale (cf actuellement) comme à distance

III – Difficultés

III-1 - Prise en main Open CV & Réalité augmentée

La principale difficulté a été l'adaptation à openCV qui est une bibliothèque très riche et très précise dont les nombreuses fonctionnalités sont difficiles à appréhender pour des personnes débutantes en réalité augmentée (face à la quantité importante de contenu disponible).

Par ailleurs, la réalité augmentée nus a sensibilisé à une problématique dont nous n'avions plus véritablement l'habitude qui n'est autre que l'aléa des conditions physiques d'un projet.
En effet, loin de nos tps de thermodynamique, les TPs d'informatiques de code « pur » ne sont pas sensibles aux conditions extérieures non paramétrables. Dans ce projet, il nous a toutefois fallut prendre en compte les différences de luminosité qui pouvaient venir perturber la détection et mettre en défaut nos tests ou encore l'angle d'incidence de la projection sur le plateau.

III – 2 – Temps

La création de RAGo nous est apparue comme un projet complexe et très intéressant mais aussi chronophage, peut être en grande partie à cause des difficultés cités précédemment.
Le temps a donc aussi été une difficulté importante qui a fait obstacle à certaines de nos tentatives d'améliorations de fonctions.

IV – Améliorations

IV – 1 – Amélioration des conditions de projection

Pour effectuer nos tests, nous avons dû incliner le goban et laisser le projecteur horizontal, en effet, incliner le projecteur suffisamment s'avérait dangereux pour ce dernier. Ainsi, l'utilisation d'un picoprojecteur (mais attention à la luminosité) ou la réalisation d'un support pour le projecteur constitueraient des atouts non négligeable pour l'avancement de RAGo.

IV – 2 – Amélioration de l'interface

Pour afficher les éléments de RAGo, nous projetons une déformation du goban virtuel. Toutefois, celui-ci est une unique matrice sur laquelle nous venons dessiner des formes.

Pour faciliter la réalisation des fonctions d'affichage, il pourrait être intéressant de s'intéresser à la composition de matrice par zone qui semble être prise en charge par openCV mais à laquelle nous n'avons pas eu le temps de prêter plus d'attention.

De plus, RAGo ne présente pas d'interface graphique. Le choix de la fonctionnalité voulu s'effectue dans le terminal ce qui constitue un facteur négatif quant à son utilisation. L'idéal serait donc de réaliser une interface en fenêtre permettant le choix du mode et un affichage plus « userfriendly » des données.

Dernier point concernant l'interface, il s'agit du tutorial que nous avons mis en place. Les instructions d'interactions ne s'effectuent que dans le terminal, il faudrait afficher celle-ci sur le goban pour que l'utilisateur suive directement ces instructions sans regarder son ordinateur.

IV – 3 – Mode de calibrage manuel pour perfectionner la correspondance

Nous souhaitions réaliser une calibration automatique se mettant en place par la détection des coins du goban. Ainsi, 3 améliorations liées à ce point sont à étudier :

- permettre de coordonner les repères même si les conditions de luminosité de sont pas suffisantes grâce à un cliquer déposer des points projetés sur les extrémités du goban

- permettre une recalibration en temps réel si un déplacement du système caméra projecteur a lieu.

- Permettre de jouer sur un goban totalement virtuel, c'est-à-dire sans grille, en plaçant uniquement 4 extrémités d'un carré sur le sol par exemple et projeter le goban complet.

IV – 4 – Affichage des temps

Les temps sont actuellement mis à jour en fin de tour de chaque joueur par transmission des temps de partie du client. Nous ne pouvons en effet pas imaginer transmettre le temps à chaque seconde. Toutefois, il serait intéressant d'implémenter un timer dans RAGo qui se chargerait de faire varier et afficher ce temps en temps réel et le resynchroniser à chaque fin de tour.

Conclusion

Le projet RAGo s'est bien déroulé du point de vue de l'équipe de développement et le projet s'est avéré riche et instructif. Nous avons entièrement conçue une application de réalité augmentée de façon à pouvoir être améliorée par des développeurs en respectant les fonctionnalités prévues dans le cahier des charges. Nous avons gardé à l'esprit de faciliter la tâche à de futurs développeurs durant tout ce processus. D'un point de vue technologique travailler sur la réalité augmentée offre des possibilités et une expérience intéressante qu'il est important de mettre en avant, ainsi que des difficultés physique souvent absentes des projets d'informatiques « pur ».

Il est à présent possible de jouer à une partie de go en ligne ou contre un ordinateur, ainsi que de visionner une partie grâce à la communication entre RAGo et qgo.

Nous pouvons sans nul doute conclure que ce sujet était très intéressant et a constitué une réelle montée en compétence pour l'ensemble de l'équipe notamment par la complexité de son approche.

Source

http://www.gnu.org/software/gnugo/
http://www.youtube.com/watch?v=AENJxqR0g48

Reconnaissance damier :
http://en.wikipedia.org/wiki/Blob_detection
http://remi.coulom.free.fr/kifu-snap/

Création de compte :
http://pandanet-igs.com/

Développement :
opencv.org
stackoverflow.com
qt-project.org
https://github.com/SPalominos/RAGo

QGo :
https://github.com/SPalominos/qgo

ANNEXES

ANNEXE 1 – Cahier des charges

RAGO

Cahier des charges

Nicolas DAVID, Sylvain PALOMINOS

5 février 2014

Encadré par Myriam SERVIERES & Jean-Marie NORMAND

Résumé

Cahier des charges du projet d'application (PAPPL) visant à l'élaboration d'une application de jeu de GO à distance en réalité augmentée.

Table des matières

1 Présentation du sujet

1.1 Préambule

Le présent document traite des spécifications fonctionnelles et techniques concernant la réalisation du projet "RAGO". Il est élaboré suite à la réunion du mercredi 8 janvier 2014 entre l'équipe projet et l'équipe pédagogique. Ce document établit l'ensemble du travail qui sera réalisé sous forme d'un accord entre les étudiants du projet et l'équipe pédagogique.

La situation du contexte de ce projet ainsi que la description des objectifs vont permettre de dégager les besoins du projet et de spécifier ses lignes directrices.

Outre ces points, ce document permet de cadrer les conditions et modalités d'éxécution des missions, de communiquer en amont du projet et de formaliser et ordonner l'expression des besoins et des objectifs. Il doit donc être validé par les étudiants ainsi que par l'équipe pédagogique.

1.2 Contexte

Dans le cadre du cours de Projet d'application de l'option disciplinaire Informatique, les étudiants sont amenés à travailler par binôme sur un sujet proposé par différents encadrants.

Ce sujet propose la mise en place d'une application de réalité augmentée permettant de jouer au GO sur un plateau réel avec un adversaire distant dont les images des pions seront projetés sur le plateau. L'application doit permettre de détecter les mouvements du joueur en présence et de jouer à distance en se connectant à des serveurs de jeu de GO.

2 Environnement

2.1 Licence et technologies

L'ensemble du projet sera écrit en langage C++ sous environnement UNIX. Le code sera écrit et commenté en langue anglaise à des fins de partage. Les commentaires serviront à générer une documentation par l'intermédiaire de l'outil doxygen. L'application sera développée à l'aide de la librairie openCV mise à disposition sous licence BSD *(licence libre pour réutiliser tout ou partie du logiciel)*. Toutes les fonctionnalités du jeu de GO seront reprise duclient q4go sous licence GPL.

cf document d'analyse des clientsGO à détailler

Le programme final sera sous licence CeCILL. (http ://www.cecill.info/)

2.2 Environnement de développement

L'ensemble du code et des documents seront stockés sur un dépôt Github. Chaque membre de l'équipe aura accès à ce dépôt en tant que collaborateur.

https ://github.com/Mendroyxx/RAGO

2.3 Méthodes de suivi et déroulement du projet

- rapport hebdomadaire à l'équipe pédagogique envoyé le vendredi de chaque semaine (ou sous forme de réunion avec l'équipe encadrante)
- suivi du temps passé sur chaque tâche (tableur googledoc)
- suivi du cours du projet (tableur googledoc)
- réunions à planifier suivant l'avancement du projet

2.4 Livrables

- Prototype final de l'application avec démonstration
- Rapport d'avancement hebdomadaire
- Rapport final
- Code source du programme
- Manuels utilisateur et développeur

3 Spécification

3.1 Fonctionnalités de l'application

Le but est la réalisation d'une application "RAGO" contenant :
– les différentes fonctionnalités d'un client de GO à savoir :

1. se connecter à un serveur de Go
2. jouer avec un adversaire distant.

   Le joueur physique utilisant l'application doit pouvoir défier un joueur distant par l'intermédiaire du serveur. C'est ce joueur dont les pièces seront projetées sur le plateau. Le joueur physique utilisera quant à lui de vraies pierres pour jouer la partie.
3. regarder une partie de Go

   Le joueur physique utilisant l'application doit pouvoir demander à l'application la diffusion totalement en réalité augmentée (projection des pierre sur le plateau) d'une partie de GO stockée sur les serveurs liés (panda go) ou en local sur le plateau de go.
4. jouer seul face au logiciel

   Le joueur physique utilisant l'application doit pouvoir défier une intelligence artificielle gérée par l'application. *Cette fonctionnalité ne rentre pas dans le cadre de ce projet. Elle constitue une évolution qui se baserait sur l'utilisation d'un client Go.*
5. enregistrer la partie pour permettre son visionnage

   Au début de la partie, l'application doit proposer au joueur d'enregistrer l'enchainement des mouvements qui la constituent pour permettre de la stocker en locale (voire de l'envoyer au sevreur) pour pouvoir la visionner ultérieurement (cf fonctionnalité 3)
6. interface d'aide et d'analyse de la partie

   L'application doit afficher sur le plateau les interractions possibles entre les pièces par analyse des coups effectués/en cours *Cette fonctionnalité n'est pas primordiale. Elle constitue une évolution qui se baserait sur l'utilisation d'un client Go.*

– couplées à des fonctionnalités de réalité augmentée :

1. détecter les mouvements des pièces sur le plateau d'un joueur physiquement présent

   Une fois la caméra en place, l'application doit permettre au joueur physique de réaliser une partie de go en plaçant simplement ses pièces sur le plateau et en reconnaissant automatiquement ses mouvements (début/fin du tour) afin d'envoyer les consignes de jeu au serveur.

2. vidéo-projeter les pièces du joueur distant ou de la partie visionnée

   Une fois le projecteur en place, l'application doit permettre au joueur physique d'observer les coups joués par son adversaire distant (ou au cours d'une partie visionnée) et ce dans des conditions de luminosité normales (en tenant compte des caractéristiques du vidéoprojecteur)

3. assurer un calibrage caméra-projecteur en cas de déplacement de la caméra, du projecteur ou du plateau

   L'ensemble caméra/projecteur/plateau se doit d'effectuer une calibration "automatique" et régulière une fois ceux-ci mis en place. La synchronisation se réptera à intervalles de temps réguliers afin de s'adapter en cas de mouvement de l'un des constituants de ce système. *Le calibrage dynamique constitue une évolution, dans un premier temps, une calibration se fera en considérant chaque élément comme immobile au cours de la partie.*

4. proposer l'affichage d'une pendule interactive

   Au cour de la partie, le temps influe sur les tours des joueurs, ainsi l'affichage d'une horloge avec laquelle le joueur peut interagir (pour signifier la fin d'une action/ de son tour) doit être pris en compte. 2 horloges seront considérées : une horloge laissant un certain temps à chaque joueur pour éxécuter son mouvmeent, ou un mode rapide forçant le joueur à jouer un cetain nombre de coup dans un temps donné.

5. proposer au joueur une interface pour choisir un type de partie : jouer contre un adversaire ou regarder une partie

   Une fois l'application lancée sur l'ordinateur, une interface ordinateur doit constituer l'interface principale de l'utilisateur et lui permettre de choisir quel mode de jeu, celui-ci souhaite lancer. Le jeu prend alors place sur le plateau et un message invite l'utilisateur à regarder ce plateau.

3.2 Documentation

L'ensemble du code sera commenté et documenté en anglais. De plus un document détaillant les fonctionnalités et l'architecture du programme sera rédigé en parallèle au développement auquel s'ajouteront des diagrammes type UML afin d'expliquer le travail de conception.

4 Réalisation

La réalisation de l'application de réalité augmentée suppose l'accès au matériel nécessaire par l'équipe projet. Ainsi, un vidéo-projecteur, une caméra USB et un jeu complet de Go avec les poins de couleurs noir et blanc et un plateau seront fournis par l'équipe pédagogique pour toute la durée du projet.

ANNEXE 2 – Etat de l'Art

Serveur de go :

Les principaux serveurs utilisés par les joueurs sont :
- KGS : serveur multilingue, doté notamment d'une salle en français : http://www.gokgs.com
- Pandanet-IGS : un des plus vieux serveur. International et en anglais, des joueurs de tous les niveaux (débutant à professionnel) s'y affrontent : http://pandanet-igs.com/

Les applications clientes utilisent le protocole GTP pour communiquer avec ces serveurs :
http://www.lysator.liu.se/~gunnar/gtp/
Pour stocker les informations sur les parties, un format spécifique est utilisé :
http://fr.wikipedia.org/wiki/Smart_Game_Format

Clients go :

Liste non exhaustive de clients : http://senseis.xmp.net/?GoClient
Des clients libres et open-source en C++ permettent déjà de communiquer avec ces serveurs de jeu. Leur code sera partiellement réutilisé dans l'application :

Client	ccGo	q4Go	Cgoban3
Langage	C++	C++	Java
Serveur	IGS	IGS, CyberORO, Tygem, eWeiqi etc	KGS
GNUgo	Compatible		

Projets contenant des fonctions a implémenter :

Il existe des projets libres et open-source contenant des fonctionnalités que nous aurons à mettre en place et dont nous pourrons nous inspirer.

Nom projet	ARGoTrainer	GoCam	Kifu	**ofxcv2**
Source	ftp://ftpa.ec-nantes.fr/user/jmnorman/RAGo	http://koti.kapsi.fi/thirsima/gocam/	http://wiki.elphel.com/index.php?title=Kifu%3a_Go_game_record_%28kifu%29_generator	http://www.youtube.com/watch?v=pCq7u2TvlxU
Technologie	Opencv Opengl GTP Fuego Kombilo	CImg Library ImageMagick	OpenCv	OpenCv
Spécificité	Utilisation de marqueurs Head mounted display (no projector) Calibrage en temps réel. Finger tracking Network (KGS)	Analyse d'un flux vidéo pour en sortir les position des pions	Détection du goban	Analyse et calibrage du plateau et de la projection en temps réel.
Langage	C, C++, Objective-C	C++	C++	C++

ANNEXE 3 – Guide utilisateur RAGO

Sur la page suivante, vous trouverez le guide d'installation et d'utilisation de l'application RAGo. Attention, son utilisation nécessite l'installation de qgo.

Vous pouvez aussi le trouver sur le dépôt github dans le dossier /docs.

Lien : https://github.com/SPalominos/RAGo

Nicolas DAVID & Sylvain PALOMINOS

Projet APPLication RAGO

GUIDE UTILISATEUR RAGO

Merci d'avoir installé RAGo! Cette application vous permettra de jouer sur votre goban une partie que vous feriez en temps normal sur un client de go installé sur votre ordinateur grâce à une caméra et un vidéo projecteur.

Notre application suit un certain nombre d'étapes pour se calibrer et permettre une utilisation optimale de ses fonctionnalités :

- La mise en place du matériel.
- L'*initialisation* des coins du goban à l'aide de pierres noires placées sur les coins du goban.
- La *calibration* de l'ensemble caméra/projecteur.
- L'*utilisation* de l'application en parallèle du client de go qgo.

Les différentes opérations inhérentes à ces étapes vous seront détaillées dans la suite de cet ouvrage. Nous vous souhaitons un agréable jeu à l'aide de notre application.

Nicolas DAVID & Sylvain PALOMINOS

Projet APPLication RAGO

PREAMBULE :

L'application RAGo repose sur la détection d'éléments sur un goban. Or dans la version actuelle, les résultats des différentes étapes d'exploitation (*initialisation*, *calibration*, *utilisation*) peuvent grandement varier en fonction de l'environnement. Afin d'obtenir les meilleures conditions d'utilisation, nous conseillons de suivre les recommandations suivantes :

- placer le goban sur une surface plane ayant un fond uni comme par exemple une nappe blanche afin d'éviter la détection de formes parasites lors de *l'initialisation* et de *l'utilisation*.
- Procéder à la *détection* avec une faible luminosité ambiante, ainsi la détection sera plus rapide et plus précise. Le reste des étapes peuvent se dérouler sans restriction de luminosité.
- Ne jamais déplacer l'ensemble caméra/projecteur/goban une fois la détection faite sous peine de devoir recommencer toute la détection.
- Ne jamais éclairer le plateau par la lumière directe du soleil, car les formes alors dessinées seraient détectées comme des éléments posés sur le plateau et pourraient interférer.

L'application RAGo est constituée de trois fenêtres :

- *Vidéo Projecteur,* qui se placera sur le vidéo projecteur, servira à afficher les informations et interactions sur le goban.
- *Interface* qui servira pour les échanges directs entre l'application et l'utilisateur pour que celui-ci contrôle les résultats des procédures de détections effectuées.
- Un terminal afin que l'utilisateur valide auprès de l'application des différentes étapes de configuration et qu'il puisse choisir les modes d'utilisation.

L'application RAGo n'implémente pas directement les fonctions d'un client de go. Pour cela une version modifiée de qgo est mise à disposition afin que l'utilisateur profite pleinement de fonctionnalités de jeu tel que les parties face à l'ordinateur, le visionnage de parties en ligne ou encore jouer contre un adversaire distant. Il est important d'utiliser cette version de qgo, car celui-ci est le seul capable de s'interfacé avec RAGo.

Nicolas DAVID & Sylvain PALOMINOS

Projet APPLication RAGO

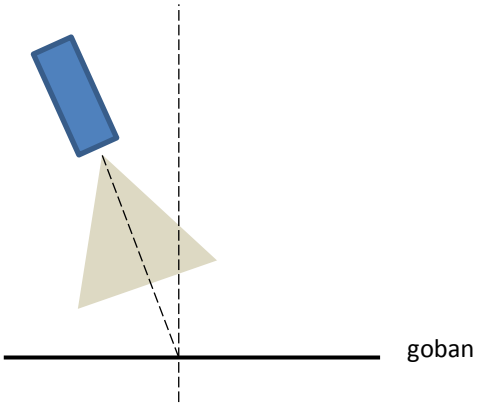
UTILISATION DE RAGO :

L'utilisation de l'application se fait en quatre étapes pour mettre dans un premier temps le matériel en place, puis la configuration du matériel en deux temps pour enfin l'utiliser.

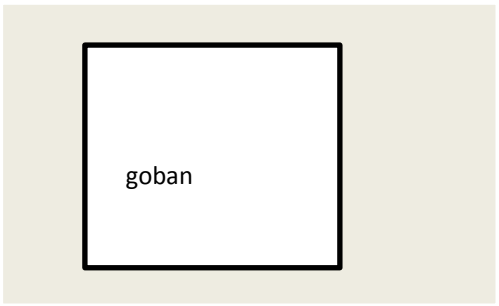
1 - MISE EN PLACE DU MATERIEL :

Avant de pouvoir utiliser l'application il est nécessaire de mettre en place le matériel. Outre le respect des conseils précédents, il est important que le trio projecteur/caméra/goban ait une disposition correcte.

D'une manière générale, l'angle d'incidence du vidéo projecteur avec le goban ne doit pas être inférieur à 30° afin de garder de bonnes performances.



De plus celui-ci doit pouvoir éclairer l'ensemble du plateau tout en laissant des marges autour du plateau correspondant à environ un quart du goban.



La caméra quant à elle peut être placée n'importe où tant qu'elle peut capturer l'ensemble du faisceau lumineux du projecteur. Il est tout de même préférable de la positionner proche du projecteur afin que les déformations dues à l'angle d'incidence de celui-ci n'influent pas sur la caméra.

Une fois le matériel positionné, vous pouvez lancer RAGo.

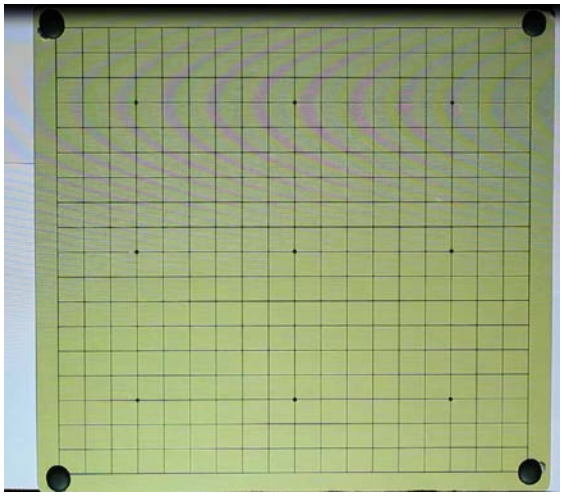
Nicolas DAVID & Sylvain PALOMINOS

2 - INITIALISATION :

Cette première étape est essentielle à l'utilisation de l'interface RAGo. Elle va permettre au système d'obtenir l'emplacement des coins du point de vue de la caméra. Elle peut s'exécuter dans des conditions normales de luminosité. Afin d'obtenir des résultats optimaux, faites en sorte que le plateau soit posé sur un fond uni, sans motifs.

Juste après que l'application soit lancée, vous êtes invités à placer la fenêtre nommée *Vidéo Projecteur* en plein écran sur le vidéo projecteur.

Une fois cela fait, placez quatre pions **noirs** sur les quatre coins du goban. Ces pions serviront de références.



Retournez sur la fenêtre *Interface* et validez la configuration sur le terminal. En retour celle-ci affichera une photo du plateau en entourant les pions détectés. Deux cas se présentent :

- Soit quatre pions ont été détectés. Vous pouvez valider les coins ou demander à l'application de recommencer cette détection.
- Soit plus ou moins de quatre coins ont été détectés. Vous pouvez soit relancer cette détection, soit quitter l'application.

Une fois l'acquisition des coins validée, l'*Interface* vous invite à enlever les pierres et à valider pour passer à l'étape suivante.

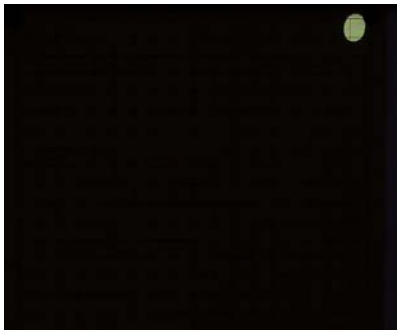
Il est possible que cette étape soit exécutée plusieurs fois avant de donner un résultat juste, car les conditions de luminosité ainsi que les imperfections des surfaces peuvent influencer les résultats.

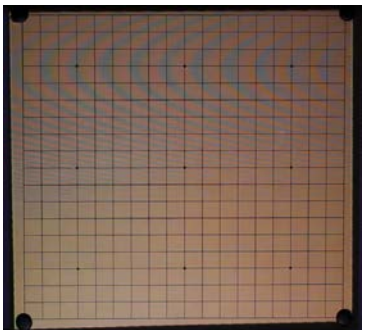
Nicolas DAVID & Sylvain PALOMINOS

3 - CALIBRATION :

Le plateau maintenant débarrassé des pierres est prêt pour la calibration. Les conditions optimales de luminosité pour cette étape sont l'obscurité. Mais cela fonctionne correctement tant que le plateau n'est pas directement éclairé par le soleil. Pour lancer la calibration, validez le retrait des pièces dans le terminal.

Le plateau va maintenant afficher une suite de rond blanc qu'il va déplacer pour les faire se superposer aux coins préalablement détectés. Cela peut prendre du temps, mais il est important de ne modifier ni la luminosité ambiante, ni la position de l'ensemble caméra/projecteur.





Une fois la calibration terminée, l'application va afficher le goban détecté et vous demande si le résultat est bon.

Il est important d'avoir une détection du goban précise (les coins détectés ne doivent pas être décalés de plus d'un cinquième de case) afin d'assurer un fonctionnement précis de l'interface.

Si les résultats ne sont pas bons, la calibration recommencera le balayage avec les cercles blancs.

4 - UTILISATION :

La calibration de l'ensemble caméra/projecteur étant fait, il est maintenant possible d'utiliser l'application. Trois modes sont disponibles :

- Partie contre un ordinateur ou en réseau
- Visionner une partie sur un serveur
- Découverte du fonctionnement

La découverte du fonctionnement prend l'aspect d'un tutoriel. Il vous suffit de suivre les indications affichées par le terminal.

Les autres modes nécessitent l'utilisation du client de go qgo. Pour son installation veuillez suivre le guide d'installation et d'utilisation de qgo modifié.

Une fois qgo lancé et le mode de jeu mis en place, sélectionnez le mode choisit sur le terminal de RAGo.

Dans le cas du visionnage d'une partie en ligne, aucune interaction n'est requise de la part de l'utilisateur.

Pour une partie, l'application demandera dans un premier temps la couleur avec laquelle va jouer l'utilisateur. Ensuite, la partie démarre.

Lors de son tour de jeu, le joueur est invité à jouer sa pierre. Pour valider la fin de son tour, il faut placer sa main dans le cercle gris et la conserver à cet endroit. Parallèlement, un cercle rouge apparaît au-dessus de la main, se réduisant au fil des secondes. Une fois celui-ci figé, le joueur doit enlever sa main pour que l'application confirme le coup et le communique au client qgo.

ANNEXE 4 – Guide utilisateur QGO

Sur la page suivante, vous trouverez le guide d'installation de de lancement de notre version modifiée de qgo.

Vous pouvez aussi le trouver sur le dépôt github dans le dossier /docs.

Lien : https://github.com/SPalominos/RAGo

Nicolas DAVID & Sylvain PALOMINOS
Projet APPLication RAGO

GUIDE D'UTILISATION DE QGO

Initialement projet de pzorin (https://github.com/pzorin/qgo) ce client de go a été choisi car il contient l'ensemble des fonctionnalités nécessaires à une bonne expérience de jeu : parties en ligne, visionnage de parties, jeu contre l'ordinateur …

Ce client a donc été modifier pour permettre la communication avec l'application RAGo et est disponible à cette adresse : https://github.com/SPalominos/RAGo.

PRE-REQUIS

Le client qgo est basé sur Qt 5. S'il n'est pas déjà installé, téléchargez le depuis la page http://qt-project.org/downloads et installez le.

INSTALLATION

Tout d'abord, importez le dépôt avec la commande :

git clone https://github.com/SPalominos/qgo

Ensuite en allant dans le dossier qgo :

```
cd qgo
```

Lancez la compilation du projet :

```
cmake
make
make install
```

Le client sera alors installé et prêt à être exécuté.

UTILISATION

Lorsque vous lancez le client, sur le menu de gauche, un onglet RAGo est présent et vous permet de spécifier au client que celui-ci doit interagir avec RAGo.

Pour jouer sur un serveur ou regarder une parties, allez dans l'onglet *Server*, faites ctrl+C pour vous connecter. Ensuite le logiciel charge les parties et les joueurs. Vous pouvez alors soit regarder une partie en cliquant dessus ou défier un joueur.

Pour jouer contre l'ordinateur, allez dans l'onglet *Go Engine*, configurez la partie et cliquez sur *new game*.

Une fois cela fait, une fenêtre vide s'affiche et vous pouvez retourner sur RAGo et sélectionner le mode de jeu correspondant à celui sur le client de go. Vous n'aurez alors plus besoin de retourner sur qgo.

ANNEXE 5 – Guide développeur RAGO

Le guide développeur est généré par Dxygen et est présent sur le dépôt github dans le dossier /docs.

Lien : https://github.com/SPalominos/RAGo