Rapport final : Projet de Fin d'Études
2016-2017

(ex ChessProject)

Participants :
Tom Dall'Agnol
Arnaud Garnier
Lisa Joanno
Simon Paris

Encadrant :
Christophe Papazian

Table des matières

1. Résumé exécutif

Objectif

Le but de notre projet est de réaliser un Jeu de Plateau en Réalité Augmentée (RA) typé jeu d'échec (tour par tour et déplacement de pions sur différentes cases).

Nos objectifs principaux sont la conception d'un mode multijoueur, la possibilité d'interagir en réalité augmentée (RA), ainsi que l'adaptation de notre jeu au Google cardboard (casque de réalité augmenté lowcost pour smartphone). Ces trois objectifs sont prioritaires pour nous car ils sont nécessaires à l'obtention d'un jeu présentable au Festival International du Jeu (FIJ) de Cannes.

Cependant, ce ne sont pas les seuls objectifs que nous souhaitons nous fixer. En effet, nous envisageons également d'ajouter de nouvelles fonctionnalités, afin de rendre notre jeu unique et innovant. Nous voulons ajouter des comportements particuliers pour nos pièces, à savoir personnaliser leurs déplacements et déterminer les pièces à tuer pour remporter la partie. L'idée ne s'arrête pas là : le joueur adverse ne serait pas au courant de la pièce à éliminer dans le camp adverse, ce qui fait que n'importe quelle pièce serait suspecte. S'en suivrait donc un jeu d'esprit qui vise à bluffer pour induire son adversaire en erreur. Par la suite, nous comptons ajouter une nouvelle fonctionnalité de cases piégées. Ces cases, en apparence normales, auraient des comportements prédéfinis afin de rendre la partie plus difficile, en y ajoutant une dimension "aléatoire".

Résultat

Nous avons implémenté des interactions pour pouvoir jouer en réalité augmentée. Elles se font par l'intermédiaire d'un objet traqué que l'on met au doigt (Voir la partie Interaction en réalité augmentée). Nous avons fait tester ces interactions lors de nos démonstrations au FIJ (plus de 100 personnes ont essayé, de tout âge)et nombreux testeurs ont réussi à interagir, bien que ce moyen d'interaction ne soit pas encore très facile d'utilisation. Notre serveur et l'implémentation du mode multijoueur a parfaitement supporté les contrainte du FIJ : pas de connexion internet et un reboot simplifié et plus rapide.

Pour ce qui est du jeu, nous avons dû mettre de côté certaines règles (échange de face, certains pièges, choix des mouvements de chaque pièce au début de la partie) que nous voulions ajouter et nous n'avons finalement que des pions prédéfinis avec pour objectif de manger les pions adverses. Cependant, le public s'étant focalisé sur la réalité augmentée et ce qu'elle apporte de nouveau, notre stand a suscité beaucoup de curiosité.

2. Description du projet

Rappel

Au vu de l'essor de la réalité augmentée et virtuelle, nous faisons un jeu utilisant la RA. Nous voulons prendre le meilleur du jeu de société et du jeu vidéo pour faire un jeu de plateau au tour par tour et exploitant au maximum la RA, en proposant un plateau de jeu cubique. Ce plateau incite les joueurs à manipuler un cube réel, afin de consulter les différentes faces et de visualiser les multiples possibilités. Nous voulons des visuels convaincants et des règles pouvant séduire même en retirant la réalité augmentée. Nous voulons aussi des interactions avec la RA qui soient les plus intuitives et efficaces possible, tout en utilisant peu de matériel (l'idée étant de se limiter à un smartphone pour rendre le jeu accessible au grand public).

Risques

Le risque que nous avions identifié initialement concernant le matériel physique tel que les cardboards et smartphone a gagné de l'ampleur : pour pouvoir faire tester notre jeu et le présenter au festival du jeu il nous faut à minima deux smartphones avec des deux cardboards compatibles. Le fait que nous soyons amenés à solliciter énormément la batterie rend peu adaptée l'utilisation de nos téléphones, nous nous sommes donc tourné vers Polytech qui n'a pas répondu à nos attentes. Il nous était donc impératif de trouver une solution afin de pouvoir avancer sur nos phases de test.

Nous avons réussi à déployer notre jeu sur trois de nos téléphones, ce qui nous a permis d'avancer dans notre travail. Cependant, en raison d'un problème d'autonomie, nous avons dû cesser notre démonstration à de multiples reprises (le temps de mettre en charge nos téléphones). La surprise de dernière minute à laquelle nous avons dû faire face a été l'absence totale de connexion internet au FIJ. Nous avons créé en conséquence un réseau local, et avons rendu la modification des adresses IP pour le serveur et les clients dynamique, afin de faciliter le lancement.

Scénarios

Nous avons 3 scénarios principaux pour notre projet.

<u>Scénario #1 : Multijoueur</u>

Donald et Hillary sont des amis de longue date. Les deux sont très occupés par leur travail mais se souviennent de leur longues parties d'échec lorsqu'ils étaient à l'école. Très compétiteurs, ils aiment se retrouver autour d'une partie de jeu de plateau. Ils possèdent tous les deux une connexion internet et l'application installée (sur téléphone ou avec des lunettes de RA).

Ils lancent tous les deux l'application et s'équipent de leurs lunettes de réalité augmentée. Donald crée un salon et Hillary le rejoint. Une fois la connexion établie, ils peuvent commencer leur partie et ils voient les mouvements de l'autre en temps réel.

Critères d'acceptation :

- Un joueur a la possibilité de créer un salon et attendre un autre joueur
- Un joueur peut rejoindre un salon déjà créé
- Deux joueurs ont rejoint le salon : la partie commence
- Lorsqu'un joueur effectue un mouvement, l'autre joueur le voit et voit les effets du mouvement de l'autre joueur
- L'état du plateau de jeu est constamment le même chez les deux joueurs du salon

Scénario #2 : Jouer en réalité augmentée

Clémence possède des lunettes de réalité augmentée et aime les jeux de plateau. Elle lance l'application et place le cube de jeu en face d'elle. Une fois en jeu, elle voit le terrain de jeu là où se trouve le cube placé devant elle. Les pièces sont placées sur le terrain de jeu.
Elle sélectionne avec sa main une pièce, puis la déplace. Elle voit cette pièce changer de position sur le terrain de jeu. Elle souhaite visualiser le reste du terrain de jeu, donc elle tourne le cube physique, et elle voit le plateau de jeu et les pièces tourner avec lui.

Critères d'acceptation :

- Un joueur, muni de lunettes de réalité augmentée, lance la partie avec le cube devant lui. Il voit le plateau virtuel avec les pièces en relief.
- Le joueur bouge le plateau virtuel en bougeant le cube en face de lui.
- Le joueur voit les pièces bouger sur le plateau virtuel.

Scénario #3 : Economie de place et d'énergie

Olivier se déplace beaucoup et reçoit régulièrement des appels pour son travail surtout qu'il a un forfait limité. Il veut donc toujours avoir de la batterie, ne veut pas s'encombrer avec beaucoup de matériel quand il veut jouer, et ne pas consommer tout son forfait téléphonique avec beaucoup d'appels serveurs.

Critères d'acceptation :

- Avoir des temps de calculs plus faibles que 2ms.
- N'avoir qu'un appel serveur par action de joueur.
- N'avoir à connecter au téléphone aucun appareil.
- Les objets pour interagir avec la réalité augmentée doivent être portables.

3. Etat de l'art

Actuellement, nous pouvons voir plusieurs techniques mises en oeuvre pour interagir avec les objets virtuels de la RA. Comme dans *Pokémon Go*, l'écran tactile du téléphone peut permettre d'interagir avec. L'avantage est la mise en place aisée car il s'agit ni plus ni moins que la technologie de base du tactile comme pour n'importe quel jeu de smartphone classique. Le problème est que, en plus de voir des éléments virtuels se superposer au monde réel, nous nous retrouvons avec des interactions proche de ce que l'on peut déjà faire avec des jeux vidéos classiques, c'est-à-dire aucune interaction avec notre environnement réel.

Un moyen plus en lien avec la réalité augmentée est l'utilisation directe nos mains pour interagir avec les objets virtuels (plutôt que de passer par l'intermédiaire d'un écran). A l'aide d'une caméra à détection de mouvement telle que la Kinect de Microsoft, nous pouvons les retranscrire dans le monde virtuel et ainsi jouer avec les objets virtuels. Dans notre cas de jeu de plateau où seules les mains sont importantes (manipulation de pièces), nous pouvons aussi employer un dispositif de *Leap Motion*. Ce dispositif permet de capter seulement les mains et les avant-bras d'un utilisateur pour les retranscrire précisément dans le virtuel. Le problème de leur utilisation est qu'il force l'emploi de matériel en plus du plateau et du smartphone, et qui consomme la batterie du téléphone. Le *Leap Motion* présente tout de même l'avantage d'être portatif, ainsi que de pouvoir fonctionner directement connecté avec un smartphone, alors qu'une caméra de type Kinect nécessite un ordinateur.

Un dernier moyen est l'utilisation d'un autre objet que l'on suivrait grâce à la technologie de réalité augmentée. Servant de pointeur, nous aurions un objet physique qui permet d'interagir avec le virtuel, mais sans que l'on puisse toucher nous même le virtuel. Le problème est qu'en cas de problème de *tracking* (ie suivi) de l'objet utilisé, nous ne pouvons plus agir sur le virtuel.

4. Mise en œuvre

Lots et planning envisagés

Voici les lots que nous avions initialement prévu mais que nous n'avons pas pu réaliser :

- **Changement de face (3 hommes/semaine)**
  Notre plateau étant un cube, nous voulons ajouter la possibilité d'échanger les différentes faces. Il faut donc pouvoir sélectionner deux faces et pouvoir les échanger, que ce soit visuellement mais aussi au niveau du modèle de donnée.

- **Pièges (2 hommes/semaine) :**
  Notre idée est de pouvoir donner la possibilité au joueur de placer des pièges sur le plateau. Voici des idées de pièges :
  - Une case craquelée, qui se détériore dès qu'une pièce s'y déplace, jusqu'à ce que la case se casse et qu'on ne puisse plus s'y déplacer.
  - Une case incendiée, où les pièces ne peuvent pas se déplacer (on peut même imaginer que le feu se propage, rendant les cases adjacentes risquées...).
  - Une case convertisseuse, qui convertit en une pièce de la couleur du camp adverse (si on convertit la pièce à tuer, il faudra que le joueur qui perd sa pièce reconfigure une pièce à tuer).
  - Une case téléporteuse, qui déplace toute pièce la franchissant sur une autre case aléatoire non occupée.
  - Une case passeuse de tour, qui n'a pas d'effet sur le plateau mais permet à l'adversaire de jouer deux fois de suite.
  - ... plus d'idées à venir ...

- **Cases en relief (1 hommes/semaine)**
  Notre idée est d'introduire des cases en relief. Ces cases sont donc des obstacles. On associe aux pièces un niveau d'escalade, qui lui permet d'escalader les cases d'altitude inférieure ou égale à son niveau d'escalade.

La raison pour laquelle nous n'avons pas réalisé ces lots est le temps que l'interaction en RA nous a finalement pris. Nous avons sous-estimé ce temps, de recherche, tests, essais, et nous avons décidé de faire passer ce temps sur ces lots. Nous avons choisi de ne pas réaliser ces lots là parce qu'ils sont ceux qui ne mettaient pas en péril la jouabilité du jeu lors de l'échéance finale : le FIJ. Notre objectif constant était d'avoir un résultat final jouable. Ces lots étaient du bonus donc non primordiaux pour le FIJ.

A noter que le lot **Pièges** a tout de même été partiellement réalisé, car nous voulions avoir une preuve de concept que nous pouvons avec notre modèle de jeu "interagir" avec les cases. Nous avons donc un traitement des événements complexes : un *évènement* (comme le déplacement d'un pion sur une case) peut être la source d'un autre *évènement* (par exemple : le déclenchement d'un piège), qui peut lui-même être un *évènement* (par exemple : pour le cas d'un piège, téléportation de la pièce ?) et ainsi de suite.

Gestion du projet

Les interactions avec la réalité augmentée ont été compliquées à implémenter, d'autant plus qu'elles ont été implémentées au départ dans une situation qui ne reflétait pas le cas réel d'utilisation : nous filmions avec les webcams de nos ordinateurs des images tandis que l'utilisation finale était d'utiliser une cardboard avec un smartphone pour filmer et manipuler un cube dont toute les faces

seraient traquées. Ainsi il a fallu d'une part trouver au plus vite une solution pour les smartphones et collaborer avec le Fablab pour obtenir le cube dont nous avions besoin. La problématique des smartphones s'est en partie débloquée lorsque nous avons réussi, à force de recherches, à déployer notre jeu, non plus sur un téléphone mais trois. Pour ce qui est du travail avec le FabLab, nous avons utilisé les semaines non dédiés au PFE (une demie journée) pour travailler sur le cube et sa fabrication afin d'avoir tout le matériel nécessaire lors des semaines à temps plein. L'idée était la suivante : les horaires de disponibilité du Fablab sont restreintes et rendent le travail en urgence très compliqué, tandis que nous pouvons plus facilement écrire du code lors des derniers jours. Nous avions aussi levé les verrous techniques les plus importants et étions assurés à 3 semaines du FIJ de pouvoir montrer un projet crédible.

Les semaines non dédiées ont aussi été utilisées pour des démonstrations systématiques à notre encadrant afin de maintenir un bon rythme de travail et d'avoir un retour sur ce que nous avions fait. Nous n'avons pas hésité à faire de nombreuse réunions afin d'harmoniser nos objectifs et que l'ensemble du groupe soit en accord avec les orientations du projet.

Enfin nous avons appris la dernière semaine que nous n'aurions de connection internet lors du FIJ, un code maîtrisé de bout en bout et suffisamment modulaire nous a permis de trouver une solution dans les délais.

Ainsi pour résumer ce que nous estimons être des bonnes pratiques :

- Prendre le temps de s'accorder sur les ambitions de chacun, tout au long du projet ;
- Lever les verrous techniques au plus tôt ;
- Lever les verrous matériels au plus tôt ;
- Maîtriser son code et le rendre modulable autant que possible pour être réactif et pouvoir faire face à des imprévus de dernière minute.

5. Résultats

Livrables

Modèle de jeu

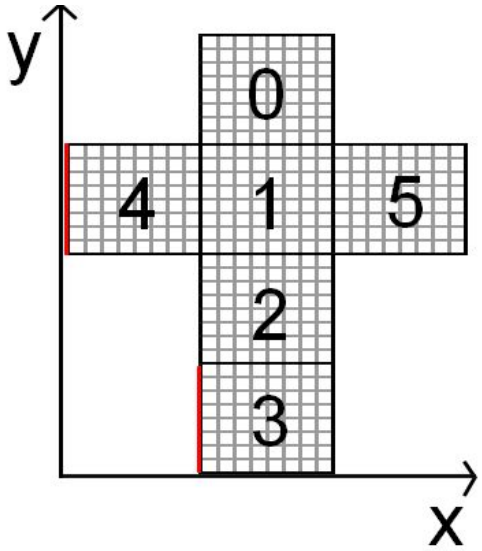
L'implémentation du modèle de jeu a commencé par un refactor du modèle que nous avions produit en PNSInnov. Le plateau est composé tout d'abord du modèle 3D de cases, regroupés en lignes, puis en plateau.

Pour chaque élément du plateau (case et pion), nous attribuons à leur représentation objet une *Position*, contenant un vecteur 2, entre (0,0) et (8,8) pour un plateau de 8x8 cases, mais aussi un *Board*, la face sur laquelle le pion est situé. À la création du plateau, on crée, pour chaque face :

- Un *Board*, classe qui connait la liste de ses voisins, les autres *Board* (up, down, left et right) ;
- Un dictionnaire des modèles 3D des cases qui constituent ce *Board*, reliés avec leur *Position.*

Les pions ont aussi des mouvements possibles qui leur sont attribué. Pour pouvoir initialiser des mouvements spécifiques pour chaque pion, nous avons choisi de les rendre génériques : à l'initialisation du script, nous leur attribuons une case, relative à la position du pion, et une possible extension. Explication : si on veut un mouvement possible de 5 cases vers le haut, on choisit la coordonnée (0,1) (vers le haut), puis on autorise une extension, qu'on place à 5. De cette manière, on peut aller en diagonale,

Pour qu'une pièce puisse être cohérente dans ses mouvements quand elle change de plateau, nous avons introduit des matrices de conversion entre chaque face des *Board*.



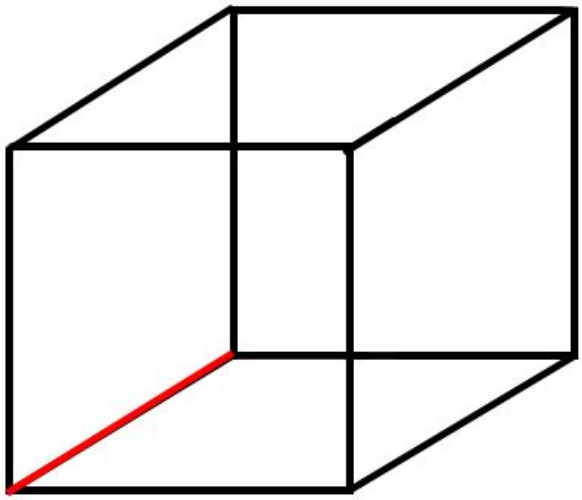


Figure 1.a : Patron du cube avec l'arrête entre les faces 3 et 4 en rouge

Figure 1.b : Cube avec l'arrête entre les faces 3 et 4 en rouge

Nous avons un modèle assez simple : un repère pour chaque face (voir la figure 1.a et la figure 1.b) et une matrice de conversion de vecteur ainsi qu'une lambda expression pour préciser la coordonnée de la case atteinte sur le nouveau plateau en cas de mouvement de la pièce.

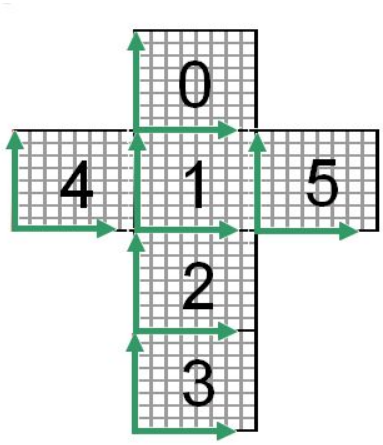


Figure 2 : Nouveau modèle du plateau avec un repère par face

Dans les faits, lorsque le mouvement de la pièce nous fait sortir du repère, on applique la matrice de conversion de vecteur sur le reste du mouvement (ainsi on continue notre mouvement dans ce nouveau plateau) et on donne comme nouvelle origine du vecteur reste le retour que l'on a de la lambda expression.

Par exemple, imaginons que nous soyons sur la face **4** dans la figure 2, en coordonnées (1,6). Nous avons un mouvement de (-2,1) (soit 2 cases à gauche et une en haut) ce qui nous ferait arriver en (-1, 7). Comme nous avons un dépassement, nous effectuons d'abord le mouvement pour dépasser le bord (soit le mouvement (-2,0)). Ensuite, nous appelons la lambda expression avec la coordonnée d'où l'on vient pour dépasser (soit (0,6)) ainsi que la taille du nouveau plateau. Celle-ci nous renvoie la coordonnée où on arrive après le dépassement et à partir d'où nous allons continuer le mouvement. Pour un cube, dépasser par la gauche la face gauche nous fait arriver à la gauche de la face **3** (voir figure 3), on arrive ainsi à la case (0,1). Le mouvement restant étant de (1,0), la matrice de conversion de vecteur va transformer ce (0,1) en (0,-1) pour que le mouvement reste logique.

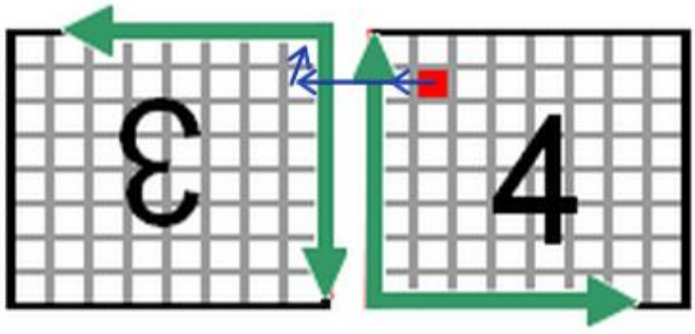


Figure 3 : Mouvement possible pour une pièce type cavalier aux échecs en mettant les faces du cube à plat

A noter que notre plateau final a une taille de 4 cases : en effet, nous avons réalisé que lors du jeu avec les lunettes de réalité augmentée, un plateau trop grand nuisait à la jouabilité. Ce n'est pas un problème : nous avons rendu la taille du plateau paramétrable pour pouvoir passer d'un jeu de 8 cases à un jeu de cases sans effort supplémentaire.

Serveur

La possibilité de jouer en multijoueur étant un de nos principaux objectifs, la synchronisation entre les joueurs est impérative. Pour ce faire, deux choix s'offraient à nous :

- Utiliser les outils fournis par Unity qui facilitent l'implémentation d'un jeu multijoueur ;
- Implémenter une solution complète pour un jeu multijoueur.

Nous avions déjà manipulé les outils d'Unity concernant le multijoueur lors du PNSinnov, ils ne sont rien d'autre qu'une librairie spécialisée. Mais nous n'avions pas réussi à implémenter un jeu multijoueur pour la raison suivante : l'utilisation de cette librairie induit des modifications dans l'ensemble de notre projet et il était clair qu'utiliser de tels outils devait être anticipé. De plus, l'échange de données entre les deux jeux est énorme : nous renvoyons la totalité de l'état de jeu courant ce qui est incompatible avec notre volonté de jeu portable, nous souhaitons que notre application consomme un minimum de batterie. Ainsi, cette solution souffre d'un manque de modularité et contraint complètement notre code. C'est pour cette raison que nous avons décidé de nous orienter vers une solution personnalisée, nous laissant en pleine maîtrise de notre code et la possibilité de nous adapter aux imprévus. De plus, il nous laisse une plus grande marge d'optimisation.

Nous avons donc implémenté un serveur en *Go*. Ce langage a été choisi pour les facilités qu'il offre à gérer la concurrence, et ainsi nous permettant d'implémenter un serveur pouvant synchroniser 2 joueurs. A terme il nous permet aussi d'héberger une multitude de parties en simultané. Grâce à un système de queue à messages et de *goroutine* (sorte de *micro thread*) permettant de lancer plusieurs processus en parallèle, nous avons en une semaine implémenté un serveur. Ce serveur répondait parfaitement à nos exigences pour le FIJ, et nous assurait des démonstrations sans incident.

Pour rentrer plus en détails dans le serveur, chaque face du cube était modélisée par un tableau avec autant d'entier que de cases sur la face. 0 quand la case est vide, 1 quand une pièce est présente dessus. Pas besoin de s'encombrer avec la nature de chaque pièce car les joueurs ont le même état initial, c'est-à-dire la même position pour l'ensemble des pions. Ainsi, la seule chose impérative est le mouvement de chacunes des positions au fil des tours. Le serveur va donc recevoir systématiquement un message contenant la position d'une pièce et sa nouvelle position, et mettre à jour ses tableaux pour rendre le mouvement effectif. Cela permet un stockage minimal d'information. Cependant, dans l'état actuel, il nous est impossible de sauvegarder une partie. Ceci dit, au vu de notre jeu, cela n'a jamais été une de nos exigences de part la rapidité des parties et de la nature du jeu.

Pour la montée en charge, nous avons aussi en place un serveur de Service Discovery comme indiqué dans la figure 4. Lorsqu'un serveur de jeu se lance, il envoie à ce deuxième serveur son adresse IP et les salons de jeu possible. Un joueur, lorsqu'il veut se connecter à une partie, fait d'abord une requête au Service Discovery pour récupérer la liste des salons de jeu atteignables, puis se connecte au serveur de jeu choisi. De cette manière, nous pouvons lancer ou enlever des serveurs en fonction de l'afflux de joueurs, ce qui nous permet de monter en charge suivant les évènements.

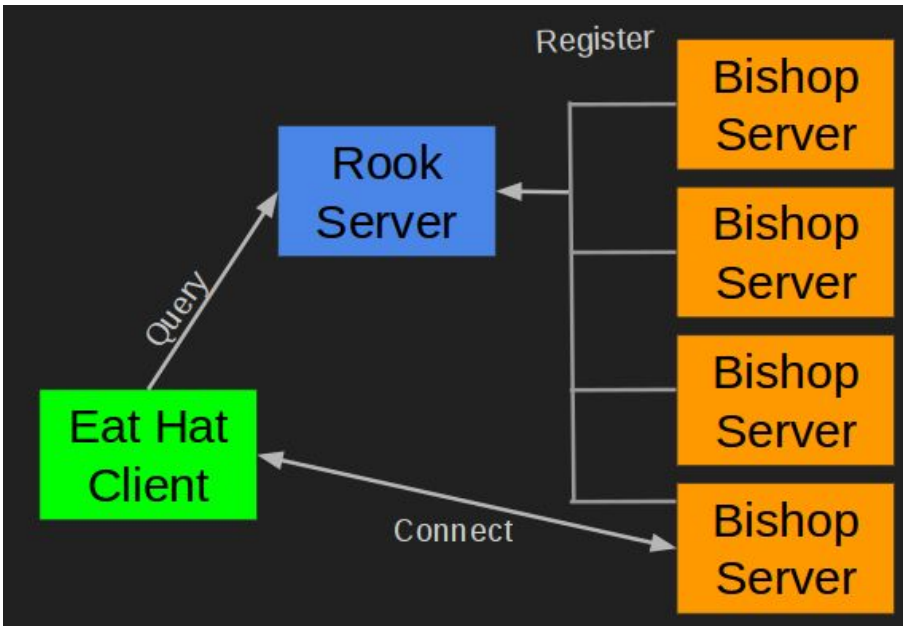


Figure 4 : Les Bishop Server sont les serveurs de jeux, tandis que le Rook Server est notre Service Discovery

Interaction en réalité augmentée

Afin d'interagir avec le jeu, nous avons utilisé un objet cubique, que l'on positionne sur le doigt. Cet objet fait apparaître un "laser" virtuel comme on peut le voir dans la figure 5. Nous avons utilisé le mécanisme de *box collider* interne à Unity : chaque objet possède un *box collider*, qui permet la gestion des collisions avec ce dernier. Ainsi, pour sélectionner une pièce, il suffit que le laser entre en collision avec un pion ou une case.

Au cours du FIJ, nous avons pu constater de l'utilisabilité de ce moyen d'interaction et collecter des retours. Ayant beaucoup expérimenté les moyens d'interactions, les membres de notre groupe de projet n'ont pas eu de difficulté pour interagir avec le dispositif. Cependant lors du festival, seule la moitié des utilisateurs qui ont essayé notre produit ont réussi à correctement interagir avec. La difficulté à sélectionner vient principalement du fait qu'il faut conserver l'image de l'objet cubique dans le champ de vision de l'utilisateur.

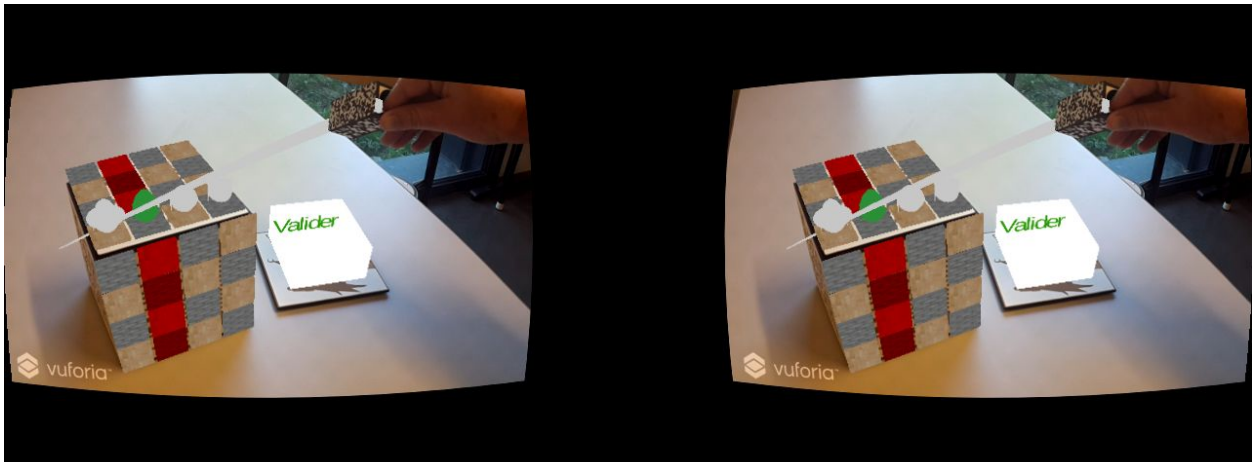


Figure 5 : Interaction avec le plateau en jeu

Résultat de recherche

L'une des principales difficultés de notre projet a été la partie concernant l'interaction en réalité augmentée. En effet, étant un domaine relativement récent, il n'existe pas encore de solution générique. Plus que de pouvoir interagir avec nos objets virtuels, il faut que notre produit soit jouable, et donc simple d'utilisation.

Les interactions à effectuer pour le joueur sont de pouvoir sélectionner un pion puis la case sur laquelle on souhaitait diriger le pion. Plusieurs pistes ont été creusées afin d'y parvenir.

La première est l'utilisation de la main pour sélectionner un pion et une case : le plugin Vuforia permet l'utilisation de boutons virtuels. Ces boutons sont des zones invisibles dans lesquelles on détecte le passage de la main de l'utilisateur. Le souci lié à l'implémentation de cette solution fut la précision. En effet, comme les cases sont relativement proches les unes des autres, il n'était pas toujours facile de sélectionner la case de notre choix. De plus, comme nous avons travaillé sur un cube, les boutons se superposent, ce qui entraîne des comportements indésirables (comme par exemple la sélection d'une case sur la mauvaise face).
La solution qui aurait été envisageable était d'agrandir le plateau virtuel, mais cela ne réglait pas le problème de superposition des boutons.

La seconde piste que nous avons développée a été le *tracking* d'objet physique. Nous avons en effet trouvé qu'un objet permettant de "pointer" sur un pion ou une case était plus précis que les boutons virtuels.
Afin de mettre en place cette solution, la première étape a été de scanner un objet physique (l'objet à *tracker* par la suite). Nous avons commencé par scanner un simple stylo. Le scan est possible en utilisant seulement un smartphone, grâce à l'application *Scanner* de Vuforia (disponible gratuitement sur le Play Store). Le soucis que nous avons rapidement détecté a été la complexité de l'objet : plus l'objet était simple (avec peu de relief) et plus il était difficile de le *tracker* dans la vidéo. Le second objet que nous avons scanné a été une télécommande. La reconnaissance de l'objet et son *tracking* furent bien meilleurs qu'avec le stylo. Ces deux objets sont montrés dans la figure 6.

Figure 6 : Les deux objets scannés pour le tracking.

Cependant, lorsque l'on orientait l'objet vers l'avant pour pointer vers le plateau virtuel, on perdait très facilement l'objet et cela affectait grandement l'utilisabilité du produit.

Nous avons par la suite expérimenté d'autres types d'objets à *tracker*. Vuforia propose plusieurs types de templates d'objets trackables : les images simples, les objets, les cubes et les cylindres. Nous avons donc par la suite tenté d'avoir un objet que nous pouvions mettre sur le doigt. Ce choix nous a semblé le plus cohérent, dans l'optique de pointer l'objet avec son doigt.
Le premier essai a été effectué sur le cylindre. Nous avions un tube à l'extrémité du doigt, sur lequel était dessiné un motif. Le *tracking* de ce type d'objet n'a pas été concluant, à cause du fait que l'objet était très petit.

Le second essai fut d'utiliser un objet cubique. En effet, nous avons remarqué que Vuforia arrive beaucoup mieux à suivre des images planes, et donc les faces d'un cube. Ce second choix fut retenu au final, car il était le mieux suivi. Nous avons utilisé de plus des QR codes, qui sont très bien *trackés* par Vuforia.

6. Conclusion

Sur le contenu

En ce qui concerne le jeu, nous n'avons pas rempli tous les objectifs que nous nous étions fixés en termes de règle et de contenu. Cependant, nous avons pu présenter un jeu innovant dans sa manière de jouer. Pour ce qui est de la technique : gestion du multijoueur, modèle du jeu et interaction nous livrons un projet convaincant qui malgré le manque d'approfondissement dans le jeu et le visuel a suscité une grande curiosité lors du FIJ.

L'interaction avec la réalité augmentée rend, du fait sa nouveauté, le travail très exploratoir et transforme en véritable challenge la livraison d'un produit testable par un public large. Nous y sommes arrivé même si ces interactions peuvent encore être largement améliorées.

Concernant le modèle et l'implémentation du jeu, de même que le serveur, la rigueur avec laquelle nous les avons développés nous a permis de réagir dans les délais à de nombreux contretemps et nouvelles contraintes : cela nous conforte dans l'idée que nous produisons un code maintenable et d'une certaine qualité.

Sur la manière de travailler

Ce projet a réellement débuté lors du PNSinnov. En allant le présenter au FIJ, c'est une ambition commune datant d'un an que nous réalisions. Si cela a été possible, c'est tout d'abord car

nous avions les mêmes ambitions : les nombreuses réunions n'avaient pas pour but de déterminer "que faire" mais "comment faire" pour atteindre nos objectifs. C'est une communication systématique entre nous qui nous a permis de garder notre cap durant tous ces mois.

Concernant notre travail chacun avait la charge d'une partie précise et a ainsi pu monter en compétence sur des points clés. Grâce à notre communication efficace, la maîtrise grandissante de chacun s'est immédiatement ressentie dans notre projet.

Nous avons su faire des compromis, autant sur les aspects matériels que sur les aspects de notre jeu, afin de limiter au maximum les risques pour notre projet.

Notre équipe n'a cessé de s'améliorer dans son organisation et sa communication : poussés par nos ambitions communes, nous n'avons pas hésité à mettre les problèmes sur la table et à y apporter des réponses. Nous avons notamment profité de la reprise de notre projet suite au PSNInnov pour ajuster notre fonctionnement au vu des problèmes rencontrés dans le passé.

Tous d'accord sur la réussite de notre projet, nous serions prêts à repartir dans un projet similaire en suivant la même organisation.

Annexe

Etat de l'art

Pour créer un jeu mobile employant des objets virtuels 3D (comme nous le souhaitons), plusieurs choix s'offrent à nous :

- Unreal Engine 4
- Unity3D

Les deux premiers sont des solutions assez similaires dans le fait qu'ils sont tous deux des moteurs de jeu mais aussi des outils complets (c'est à dire qu'ils proposent un outillage complet pour faciliter la création de jeu vidéo). Ainsi, sous la forme de "Scènes", un Concepteur de Jeu peut mettre en place les niveaux de jeux, paramétrer les différents éléments, mais aussi changer les modèles 3D des objets sans avoir à toucher au code. La logique de jeu proviendra par contre des développeurs. Ils devront coder les interactions et les comportements des différents objets. Un autre avantage est la présence d'un "marché" intégré, permettant à d'autres créateurs de proposer (gratuitement ou non) des modèles 3D ou des comportements pré faits. Tous deux permettent aussi de déployer notre jeu sur différents supports : android, IOS, Windows, OSX, Linux.

Le problème de ces deux outils étant que tous les deux produisent en sortie des jeux assez lourds car ils embarquent tout le moteur physique et tous les éléments "facilitateurs" directement dans l'exécutable final.

Au niveau des différences, Unreal Engine 4 possède une moins bonne communauté que Unity3D. Dans ce dernier, la documentation est à jour, précise et propose des exemples appréciables. Un simple tour sur les forums de Unity permettent d'avoir des informations lorsque l'on est bloqué lors de l'implémentation d'un comportement.

Par contre, nous n'avons pas accès à la base de code de Unity3D, certains comportements restent donc "cachés" voir "aléatoires", contrairement à Unreal Engine qui est ouvert. Autre avantage de l'Unreal Engine est de proposer des rendus très réalistes et donc plus beau que Unity.

Ainsi, nous avons opté pour Unity3D pour la communauté solide, qui est toujours la bienvenue lors de problème avec une technologie. Proposer de meilleurs graphismes (ou tout du moins plus réaliste) n'est pas à prendre en considération quand on sait que notre cible est un smartphone, support qui n'aurait pas les capacités de faire tourner un jeu avec des graphismes trop réalistes.

Côté Réalité Augmentée, on voit qu'il existe un certains nombre de cadriciels permettant de faciliter son utilisation. Parmi les plus utilisés, nous retrouvons Vuforia et ARToolkit, tous deux proposant de s'interfacer avec Unity3D. Ces technologies ont été choisies principalement par leur utilisation de "marqueurs", c'est à dire d'objets réels qui servent au logiciel de placer les objets virtuels dans le réel. Comme nous souhaitons proposer un jeu de plateau en Réalité Augmentée, permettre d'avoir un plateau réel sur lequel ajouter des pièces et des animations en 3D est exactement ce que l'on souhaite.

L'avantage de Vuforia est de toujours faire apparaître l'objet virtuel alors même que le marqueur est en grande partie caché : nous avons un meilleur tracking ce qui permet une utilisation poussée de l'objet qu'on souhaite suivre. Vuforia propose aussi un support direct des nouveaux équipements pour la réalité augmentée (que ce soit les Cardboard comme nous souhaitons utiliser, ou bien les traditionnelles webcam). Le problème viens du côté boîte noir de la technologie : aucun moyen de savoir ce que fait vuforia exactement, ce qui fait qu'au moindre problème il n'y aura pas d'aide directe. Un autre désavantage est le fait que cette technologie n'est disponible que sur IOS, Android, Windows 8 et 10 et OSX. Cela force un environnement de travail.

L'avantage de ARToolkit est sa communauté bien plus importante que Vuforia : s'il y a un problème nous pouvons facilement le résoudre à l'aide de la communauté. Un autre point positif à ce niveau là est que le code est open source, on peut donc y accéder et vérifier des comportements pour s'assurer que c'est bien ce qu'on attend. De plus, il permet de déployer sur des lunettes de réalité augmenté ce qui est un grand plus. Le gros point négatif est le tracking de marqueur, ARToolKit perdant facilement l'objet que nous souhaitons suivre.

Nous avons choisi d'employer Vuforia, car notre plateau sera manipulé régulièrement et fait parti de l'expérience utilisateur que nous souhaitons proposer : nous ne pouvons pas laisser

disparaître trop régulièrement les objets virtuels car nous cachons le marqueur nous servant de plateau de la main. L'expérience utilisateur étant un des points nous tenant à coeur, nous nous sommes donc dirigé vers cette technologie, et il nous reste entre autre à voir comment nous pouvons interagir avec.

Actuellement, nous pouvons voir plusieurs techniques traditionnellement mises en oeuvre pour interagir avec les objets virtuels de la RA.

Comme dans *Pokémon Go*, l'écran tactile du téléphone peut permettre d'interagir avec. L'avantage est la mise en place aisée car il s'agit ni plus ni moins que la technologie de base du tactile comme pour n'importe quel jeu "normal". Le problème est que, à part voir des éléments virtuels s'ajouter au réel, nous nous retrouvons avec des interactions proche de ce que l'on peut déjà faire avec des jeux vidéos "normaux".

Un moyen plus en lien avec la réalité serait d'utiliser directement nos mains pour interagir avec les objets virtuels, et non plus par l'intermédiaire d'un écran. A l'aide d'une caméra à détection de mouvement telle que la Kinect de Microsoft, nous pouvons les retranscrire dans le monde virtuel et ainsi saisir et même jouer avec les objets virtuels. Dans notre cas de jeu de plateau, seule les mains étant importantes nous pouvons aussi employer un Leap Motion. Ce dispositif permet de capter seulement les mains et les avant-bras d'un utilisateur pour les retranscrire précisément dans le virtuel. Le problème de leur utilisation est qu'il force l'emploi de matériel en plus du plateau et du smartphone. Le leap motion a tout de même l'avantage d'être portable et de pouvoir fonctionner directement connecté avec un smartphone, alors que le kinect nécessite un ordinateur.

Un dernier moyen est l'utilisation d'un autre objet que l'on trackerai grâce à la technologie de réalité augmentée. Servant de "baguette magique", nous aurions un objet physique qui permet d'interagir avec le virtuel, mais sans que l'on puisse toucher nous même le virtuel. Le problème est qu'en cas de problème de tracking de l'objet utilisé, nous ne pouvons plus agir sur le virtuel. De plus, on perd le côté manuel du jeu de plateau (bouger les pièces à la main par exemple), ce qui est une sensation que nous souhaitons conserver.

C'est pour cette raison que nous avons choisi l'utilisation du LeapMotion : pour garder le plaisir du jeu de plateau comme il existe normalement, nous souhaitons une interaction directe avec les mains.

Un autre des objectifs est de proposer un jeu à deux joueurs, bien que les deux soient extrêmement éloignés. Comme nous souhaitons proposer un service rapide et fluide (nous ne voulons pas attendre 10 minutes que l'action de notre adversaire soit finalement reçue), il nous faut pouvoir monter en charge et ainsi pouvoir se maintenir même avec un grand nombre de joueurs en même temps. Ce serveur permet aussi de vérifier qu'un des joueurs ne triche pas, et ne fait pas de coups non autorisés.

Une solution retenue est celle de Photon. Il s'agit d'une technologie qui peut s'interfacer avec Unity3D. Le serveur peut être hébergé sur le Cloud, et un grand nombre d'optimisations automatiques sont faites pour permettre une montée en charge conséquente. Le désavantage principal est qu'il s'agit d'une technologie centrée sur le synchrone, sur le jeu en temps réel. Or nous souhaitons faire un jeu au tour par tour, donc asynchrone. Une grosse partie des fonctionnalités et des optimisations ne nous concernent pas voir nous gênent (la transmission de donnée serait régulière et fait donc chuter la batterie de nos smartphones avec l'utilisation).

Une seconde solution est celle de Go, langage conçu par Google. Un de ses intérêts est un déploiement assuré sur le support de notre choix : le logiciel produit embarque avec lui tout ce dont il a besoin pour tourner. On peut donc déployer sur un serveur sans avoir un installer un grands nombre de dépendances et/ou de prérequis : la simple ligne "go run monProgramme" suffit au lancement. De plus, le concept de goroutines permettent de passer à l'échelle : il a été fait pour paralléliser des programmes (exécuter en parallèle des programmes pour éviter un trop grand temps d'attente alors qu'un grand nombre d'utilisateur essayent d'accéder à notre serveur). Le problème va être de devoir établir un protocole de communication avec notre serveur, ainsi que des messages datés (il ne faudrait pas qu'un des joueurs reçoivent des actions qui ont lieu après d'autres qu'il n'a pas encore reçu) mais aussi établir le modèle de jeu à la fois côté serveur dans un langage, et côté client dans un autre.

Nous avons donc choisi d'utiliser le langage Go et faire notre serveur nous même, pour permettre une montée en charge certaine pour notre cas, celui d'un jeu au tour par tour.

Une autre problématique liée au déploiement sur smartphone est la consommation de donnée. Dans le cas où nous souhaitons jouer en extérieur, il faut penser à réduire les échanges au maximum (pour les biens de la batterie mais aussi du forfait data du joueur). Il est donc inimaginable d'envoyer un état de jeu complet au serveur et que celui ci le transmettre aux joueurs. Dans l'industrie du jeu, ce qui est souvent fait est d'envoyer l'action effectuée (par exemple aux échecs, transmettre seulement "cavalier part de C7 pour arriver en D5"). Ainsi, nous n'aurons qu'à exécuter l'action transmise pour se retrouver dans le même état de jeu que l'adversaire. Nous pouvons aussi sauvegarder côté serveur les différents coups joués pour les renvoyer au joueur s'il a perdu la connexion. On parle ainsi de Programmation Basée sur des Évènements. Bien qu'il faille exécuter du coup trois fois le coup (une fois pour le joueur qui fait le coup, une fois pour le serveur pour vérifier la validité du coup et une fois par le joueur subissant le coup pour mettre à jour son côté), nous y gagnons en termes de nombre de données échangées.