GAUTIER Lucas
LIU Yuhao
GAUTIER Baptiste
Encadré par GUERRIOT Didier et
VINCENT Johann
À destination de l'équipe pédagogique du projet développement

Rapport de projet développement : Jeu MTSGo – Application mobile

7 juin 2017

Rapport Version n°2
Projet n°15

TELECOM Bretagne
Institut Mines-Télécom

Résumé (english version below)

Rédacteurs : GAUTIER Lucas et GAUTIER Baptiste Relecteur : LIU Yuhao

Les jeux mobiles en réalité augmenté (AR, cf glossaire) ont connu un succès grandissant ces dernières années, grâce, notamment, à la sortie de Pokemon Go de Niantic. Nous avons développé, sous Android, une application similaire, à deux différences majeures près. Les utilisateurs peuvent ici répondre à des questions portant sur le cours de Mathématiques et Traitement du Signal (MTS) de Telecom Bretagne et le terrain de jeu se limite au campus brestois de l'école. L'application est donc nommée MTSGo.

La réalisation de MTSGo a nécessité la formation de quatre groupes de projet : serveur, géolocalisation, supervision et application, dont nous nous sommes chargés.

Nous avons commencé par réaliser les interfaces utilisateur (écran de connexion, d'inscription…), et à coder les requêtes entre le serveur et l'application (récupération des questions de MTS, connexion…). Nous avons ainsi créé un prototype fonctionnant sous Android, permettant à un utilisateur de s'inscrire, se connecter, répondre à des questions et détecter sa position GPS (cf glossaire). Cependant, Android Studio ne permettant pas de générer la carte du campus sur lequel l'avatar de l'utilisateur se déplace, nous avons continué notre travail sur Unity3d.

La plateforme de développement Unity3d nous a permis de réaliser un second prototype de MTSGo, permettant de valider l'ensemble des critères spécifiés dans le cahier des charges de notre projet. Nous avons réalisé de nouvelles interfaces utilisateur, obtenu une carte précise du campus, permis aux joueurs d'y déplacer leur avatar, et de répondre à des questions pouvant constamment être mises à jour.

Virtual Reality mobile games (VR) have recently known an increasing success, mostly thanks to the release of Pokémon Go developped by Niantic last summer. We have developed a similar application though there are two major differences: Users may answer to questions on Telecom Bretagne 's Mathematics and Signal Processing (MTS) course and the playing field is limited to the campus of Brest. The application is therefore named MTSGo.

The development of the application required the formation of four project groups: server, geolocation, supervision and application, that we took care of.

We started by creating the user interfaces (login screen, registration ...) and to encode the requests between the server and the application (retrieval of MTS questions, connection ...). We thus created a prototype running on Android, allowing a user to sign up, log in, answer questions and detect his GPS position. However, Android Studio does not allow generate the campus map on which the user's avatar moves, so we continued our work on Unity3D.

The development platform Unit3D allowed us to realize a second prototype of MTSGo, validating all the criteria specified in the specification sheet of our project. We have created new user interfaces, get a precise map of the campus, allowing players to move their avatar, and answer questions online.

Sommaire

1. INTRODUCTION

Rédacteur : GAUTIER Lucas Relecteurs : LIU Yuhao et GAUTIER Baptiste

1.1 CONTEXTE GLOBAL

La démocratisation de la réalité augmentée (AR) grâce à Pokemon Go est fortement corrélée avec la naissance du projet sur lequel nous travaillons. Le succès mondial qu'a rencontrée la pionnière des jeux en AR est la preuve de la viabilité de ce modèle. C'est pourquoi MTSGo est fortement inspirée de Pokemon Go, tant dans le traitement des règles qui régissent le jeu, que par la façon dont il a été développé.

MTSGo est donc un jeu en AR, fonctionnant sous Android. Le terrain de jeu se limite ici au campus brestois de l'IMT Atlantique, sur lequel les joueurs sont repérés par GPS. Se déplacer sur le campus permettra de repérer des questions, placées à différentes positions du campus, auxquelles les joueurs seront libres de répondre. Celles-ci portent sur les cours de Mathématiques et Traitement du Signal enseignés à Telecom Bretagne. Le public visé est donc restreint aux personnes fréquentant le campus brestois de Telecom Bretagne. À chaque réponse correcte donnée, le joueur gagne un certain nombre de points, il peut ensuite accéder à tout moment au classement des utilisateurs de l'application.

MTSGo répond à deux besoins : celui d'une part de rendre le campus brestois de l'IMT Atlantique plus interactif, et d'autre part, celui reposant sur la pérennité pédagogique, en diversifiant les méthodes d'enseignement au sein de l'école.

1.2 4 GROUPES DE PROJET POUR UNE APPLICATION : DELIMITATION DU TRAVAIL A REALISER

La réalisation de l'application MTSGo a nécessité une répartition du travail à quatre équipes composées chacune de 4 personnes :

- L'équipe serveur qui se charge de traiter les requêtes permettant aux utilisateurs de s'inscrire, de se connecter, et de répondre aux réponses. Il s'agit aussi de l'équipe qui nous envoie une liste de questions à afficher sur la carte du campus. De plus, cette équipe interagit avec l'équipe supervision.
- L'équipe géolocalisation qui a pour but de concevoir une fonction permettant de repérer les utilisateurs avec plus de précision que celle qu'offre actuellement le GPS des smartphones. Leur travail repose notamment sur la cartographie des réseaux wifi du campus.
- L'équipe supervision qui crée une interface permettant aux développeurs d'ajouter des questions au jeu, et de modifier leur emplacement. Ces informations sont envoyées au serveur qui les relaie à l'application.
- Et notre équipe, qui se charge de l'application. Notre rôle est de concevoir les interfaces utilisateurs (connexion, inscription, score…), de représenter l'avatar du joueur sur la carte du campus, lui permettre de se déplacer en réalité augmentée (grâce au GPS), de répondre aux questions que nous envoie le serveur, et donc de traiter les échanges avec le serveur (le lecteur pourra se reporter au cahier des charges en annexe 1 pour plus de précision concernant les critères de validation).

Ce rapport détaille tout d'abord la première phase de développement de MTSGo, sous Android Studio, la création des interfaces et des requêtes permettant de communiquer avec le serveur y sont spécifiés. S'en suit une brève introduction à Unity3d permettant d'appréhender toute la partie technique réalisée grâce à lui. Puis nous allons détailler les phases d'inscription/connexion, du déplacement du joueur, du positionnement des questions et de la gestion du score et de l'historique. Le fonctionnement du jeu sera donc expliqué en détail, et ce par le biais de trois axes : une explication de ce que perçoivent les utilisateurs, la façon dont nous avons implémenté les différentes fonctionnalités de MTSGo, et une partie explicitant les critères de validation.

2. PREMIER PROTOTYPE DE MTSGO

2.1 DEVELOPPEMENT DE MTSGO SOUS ANDROID STUDIO

Rédacteur : LIU Yuhao Relecteur : GAUTIER Lucas

2.1.1 Création de l'interface utilisateur

Figure 1a : écran d'accueil de MTSGo v1 sous Android Studio

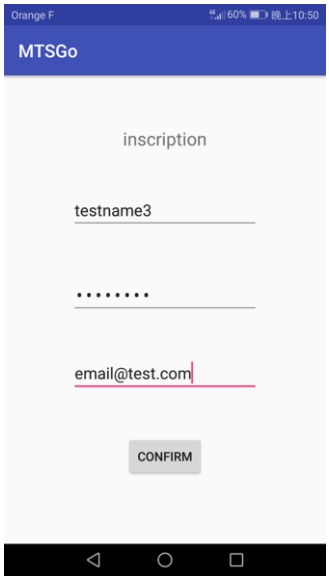

Figure 1b : écran d'inscription de MTSGo v1 sous Android Studio

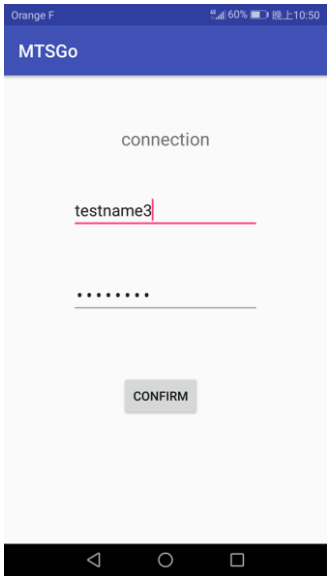

Fig 1c : écran de connexion de MTSGo v1 sous Android Studio

Nous avons tout d'abord privilégié Android Studio [1] pour réaliser nos interfaces graphiques. Pour ce faire, nous avons gardé le fond proposé par défaut par Android Studio. Puis nous avons créé des «widgets » que nous avons agencé pour créer les différentes interfaces utilisateurs. Les « widgets » sont, par exemple, les

boutons d'inscription et de connexion (fig. 1a), ou bien encore les zones de texte (fig. 1b et fig. 1c).

De plus, nous avons, à l'aide d'une classe nommée « LocationManager », obtenu la position GPS des utilisateurs, que nous avons pu afficher à l'aide d'un « widget » (fig. 2).

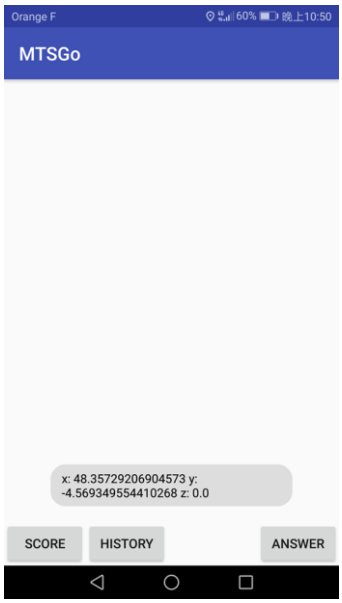

Figure 2 : écran principal de MTSGo v1 sous Android Studio

Après envoi d'une requête au serveur, nous obtenons les questions et leurs positions, auquel nous pouvons répondre (fig. 3)

Figure 3 : écran de réponse à une question de MTSGo v1 sous Android

Enfin, après nouvelles requêtes envoyées au serveur, et traitement des données, nous affichons le score (fig. 4a) et l'historique (fig. 4b).

Figure 4a : écran de score et de classement de MTSGo v1 sous Android Studio

Figure 4b : écran d'historique de MTSGo v1 sous Android Studio

2.1.2 Réalisation des requêtes serveur-application

La plus grande partie du travail que nous avons réalisé sur Android Studio consiste en la réalisation des requêtes entre le serveur et l'application. Celles-ci permettent aussi bien de communiquer au serveur les identifiants rentrés par l'utilisateur lors de son inscription que de recevoir les informations concernant les questions à afficher sur le jeu.

Afin de communiquer avec le serveur, il faut construire une requête en utilisant un objet JSON [2]. JSON (JavaScript Object Notation, cf glossaire) est un format de données textuelles dérivé de la notation des objets du langage JavaScript. Un document JSON a pour fonction de représenter de l'information accompagnée d'étiquettes permettant d'en interpréter les divers éléments, sans aucune restriction sur le nombre de celles-ci. Dans un objet JSON, il y a des ensembles de paires nom / valeur. Les valeurs peuvent être de type tableau, objet, booléen, nombre, chaîne ou null. Pour échanger les données, l'application Android et le serveur doivent se mettre d'accord sur le format de données dans l'objet JSON. Finalement, pour envoyer des requêtes, une librairie est utilisée. Elle s'appelle Okhttp [3] et il s'agit un client http (cf

```
{
    "glossary"
        "title": "example glossary",
                "GlossDiv": {
            "title": "S",
                        "GlossList": {
                "GlossEntry": {
                    "ID": "SGML",
                                        "SortAs": "SGML",
                                        "GlossTerm": "Standard Generalized Markup Language",
                                        "Acronym": "SGML",
                                        "Abbrev": "ISO 8879:1986",
                                        "GlossDef": {
                        "para": "A meta-markup language, used to create markup languages such as DocBook.",
                                                "GlossSeeAlso": ["GML", "XML"]
                    },
                                        "GlossSee": "markup"
                }
            }
        }
    }
}
```

Figure : exemple de chaîne json

glossaire).

Dans l'exemple de chaîne json ci-dessus « glossary » est une clé qui renvoie tout ce qui suit entre les accolades entourées en rouge. De cette façon, nous extrayons

les informations que nous envoie les serveurs, et nous imbriquons les informations que nous avons à envoyer dans des chaînes similaires.

Les requêtes réalisées dans l'application seront explicitées en détail dans les parties « point de vue développeur » concernant le fonctionnement de MTSGo.

Bien que nous n'ayons pas pu réaliser MTSGo sous Android Studio, la prise en main de cet outil de développement nous a servi grandement quant à la réalisation du jeu. En effet, la réalisation des interfaces utilisateurs est similaire sous Unity3d et nous avons pu réutiliser les requêtes codées sous Android Studio sous forme de plugin que nous avons implanté sous Unity3D.

2.2 PRESENTATION DE UNITY3D : DEVELOPPEMENT DU SECOND PROTOTYPE

Rédacteur : GAUTIER Lucas Relecteur : LIU Yuhao

Unity3D [4] est un moteur de jeu multi-plateforme développé par Unity Technologies. Il dispose d'une licence gratuite, et est très répandu dans l'industrie du jeu vidéo, ce qui en fait un outil très documenté. Ces particularités en font un outil de choix pour le développement de MTSGo[3].

Nous avons choisi de disposer l'interface graphique de la façon suivante :

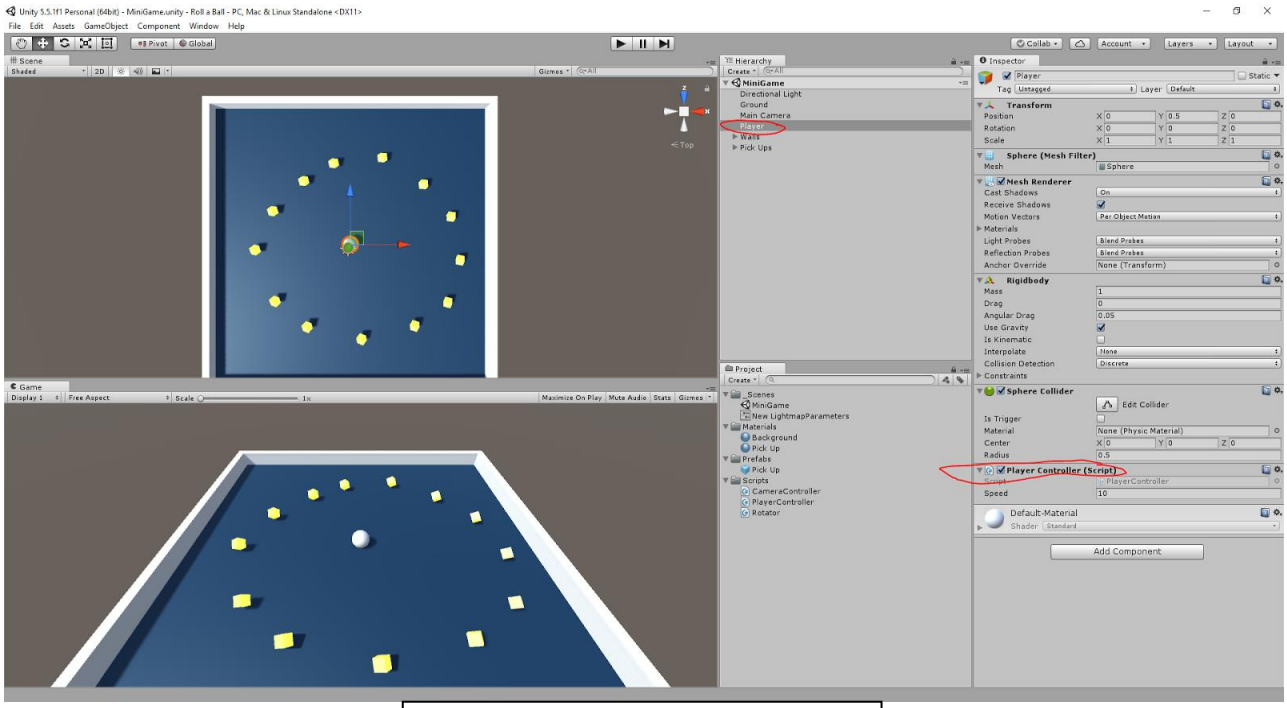

Figure 5 : interface graphique du Unity3d

Cette interface (fig. 5) est composée de 5 fenêtres principales. En haut à gauche, la fenêtre représente l'aire du jeu développé et ce qui s'y trouve, le développeur peut s'y déplacer à l'aide de sa souris. En bas à gauche, la vue représentée est celle que le joueur aura au lancement du jeu. L'onglet au milieu à droite liste les objets (cubes, sphères…) les éclairages, et les caméras utilisés pour concevoir le jeu. Au milieu en bas, il s'agit de dossier dans lequel sont organisés les matériaux, et les scènes régissant le jeu développé. À droite, l'onglet donne les détails concernant l'objet sur lequel clique le développeur dans les deux onglets précédents.

Un projet est notamment composé, sous unity, des assets (c'est ce qui nous intéresse), contenant une ou plusieurs scènes, des matériaux, des éléments préfabriqués et des scripts. Il faut charger la scène pour travailler sur le jeu, où l'on peut alors modifier les « GameObjects », lumières et caméras et leur attacher des scripts qui définiront leur comportement. Ici, on a attaché le script « Player Controller » à « Player », la sphère au centre de la scène, ce qui permet au joueur de la déplacer, sur le plateau bleu (qui n'est autre qu'un deuxième GameObject), dans la direction qu'il souhaite, via les touches du clavier.

Pour tester un jeu développé via unity3D, sur un smartphone, il suffit de télécharger les kits de développement d'Android et de Java, de les insérer dans ses préférences sous unity3D, puis de changer la plateforme à Android. Il ne reste ensuite plus qu'à construire la scène et à l'exécuter (build&run).

Une fois la prise en main de Unity3d réalisée, notre équipe a su utiliser ses similarités avec Android Studio pour réaliser le second prototype de MTSGo, que nous détaillons par la suite.

3. FONCTIONNEMENT DE MTSGO

Après téléchargement (dont la plateforme reste à définir) et installation du jeu, sur smartphone fonctionnant sous Android, le joueur -prenons par exemple un étudiant de l'IMT Atlantique que nous appellerons Jean- peut lancer l'application.

3.1 INSCRIPTION/CONNEXION

Rédacteur : LIU Yuhao Relecteur : GAUTIER Baptiste

3.1.1 Point de vue utilisateur

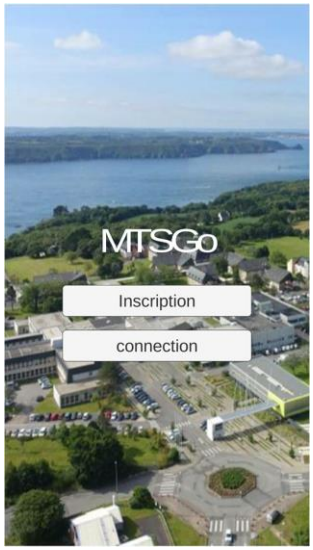

Figure 6a : écran d'accueil de l'application

Figure 6b : écran d'inscription de l'application

Jean arrive directement sur l'écran d'accueil de l'application (fig. 6a) où il peut choisir de se s'inscrire, ou de se connecter s'il possède déjà un compte. Dans le premier cas, il appuie sur le bouton « inscription » et arrive sur l'écran d'inscription (fig. 6b) où il doit saisir un nom de compte, un mot de passe et une adresse mail. Jean appuie ensuite sur le bouton d'inscription, ce qui le renvoie vers l'écran d'accueil, où il peut se connecter. Si Jean appuie sur le bouton « connection » (fig. 6a) au lieu du bouton « Inscription », il peut revenir en arrière à l'aide du bouton « Back » (fig. 6b)

3.1.2 Point de vue développeur

Quand un joueur ouvre l'application, ce qu'il voit est l'interface de menu sur laquelle il y a deux boutons (connexion et inscription) et un logo “MTSGo”. Les deux boutons sont liés à une méthode qui permet d’attendre que le joueur effectue une action pour la capturer (ici, cette méthode se nomme Onclicklistener). Si l’utilisateur appuie sur l’un des boutons, son action va être enregistrer dans le corps d'une classe. Dans cette classe, le programme va distinguer l'identité du bouton et effectuer une transition vers l’écran associé. L’écran précédent est gardé en mémoire auquel il peut accéder en appuyant sur « back » [5].

Dans l’écran d’inscription, il y a 3 zones de texte à remplir (fig. 6b). Quand le joueur clique sur le bouton de confirmation, « inscription » les informations dans les zones de texte vont être collectionnées. À ce moment-là, un nouveau processus est ouvert. A noter que, ce n'est pas bien d’envoyer les requêtes ou traiter les grandes données dans le processus principal, parce qu’il va interagir directement avec l’utilisateur : on ne peut pas attendre la réponse du serveur avant d’effectuer un processus car s’il ne répond pas, l’utilisateur se trouve bloqué sur le même écran. Dans ce nouveau processus, il faut utiliser un URL (cf glossaire) pour ouvrir une connexion et configurer la méthode de requête à “POST” car il s'agit d’envoyer quelque chose au serveur et attendre une réponse.

Après, il faut construire un objet JSON (fig. 7) dans lequel il y a toutes les données que le serveur demande. Par exemple, quand le joueur se connecte, la requête (objet JSON) doit avoir un nom d'utilisateur et un mot de passe sous forme de paires clé /valeur. Si la requête est envoyée avec succès, le serveur va rentrer une user_id et un token que l’application doit bien garder. S'il y a un problème, le serveur va rentrer d'autres codes de réponse. Par exemple, le code 403 indique une erreur au niveau du login ou du mot de passe.

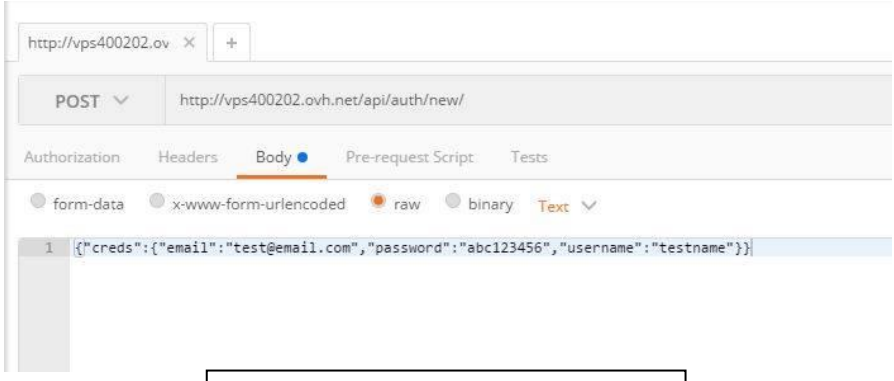

Figure 7 : exemple de chaîne json envoyée au serveur

Que ce soit succès ou l'échec, le joueur doit être informé. Mais, à ce moment-là, les informations envoyées par le serveur sont contenues dans le processus enfant, il a donc besoin d’utiliser un autre outil pour échanger les données entre les processus différents. Il s'appelle Handler. Cet outil a pour but d’envoyer un message contenant un objet et les données associées au processus principal. C'est dans le processus principal que l’on retire les informations et affiche le résultat au joueur.

Le diagramme de séquence suivant permet de résumer le fonctionnement de la connexion à MTSGo :

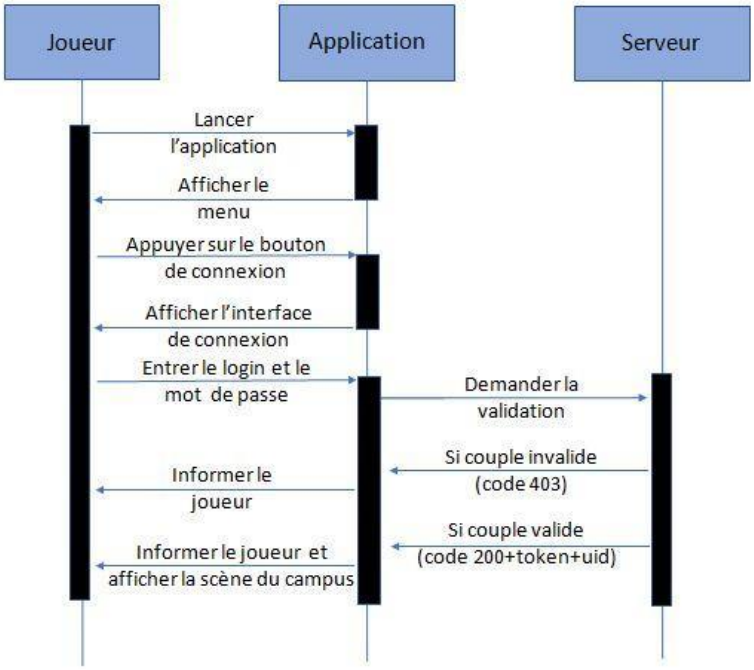

Figure 8 : diagramme de séquence de la connexion à MTSGo

3.1.3 Critère de validation

Le cahier des charges (cf annexe 1) spécifie que l'application doit recevoir un code de confirmation 200 du serveur lorsqu'un utilisateur s'inscrit ou se connecte à MTSGo de façon correcte. Ce critère est donc validé. Dans le cas d'une erreur l'application reçoit un code 403 pour la connexion et 401 pour l'inscription.

L'utilisateur peut donc s'inscrire et se connecter à MTSGo correctement.

3.2 DEPLACEMENTS DU JOUEUR ET POSITIONNEMENT DES QUESTIONS

Rédacteur : GAUTIER Lucas Relecteur : LIU Yuhao

3.2.1 Point de vue utilisateur

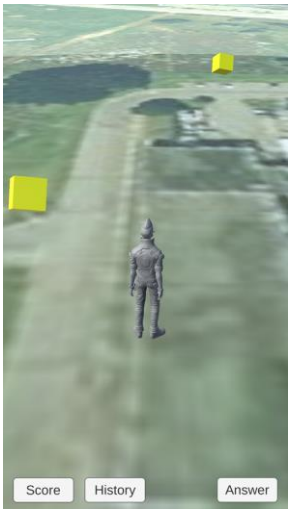

Figure 9 : écran principal de MTSGo

Une fois connecté, Jean a accès à un écran (fig. 9) représentant, au centre, son avatar, positionné sur la carte du campus brestois de l'IMT Atlantique. Lorsque Jean se déplace, la position GPS de son smartphone se modifie, ce qui résulte en le déplacement de l'avatar de Jean à l'écran. Jean peut appuyer sur les boutons « score » et « history » pour accéder aux écrans correspondant (cf 3.4). Quant aux cubes jaunes, ceux-ci représentent les questions. Jean peut appuyer sur le bouton « answer » à proximité de l'une d'elle pour accéder à l'écran de réponse à une question.

3.2.2 Point de vue développeur

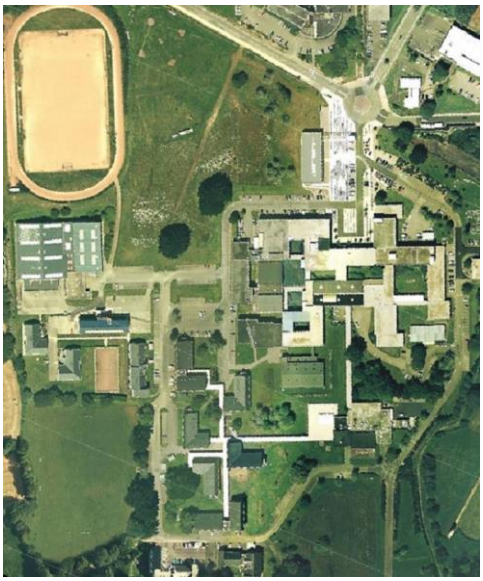

Figure 10 : carte du campus utilisée dans MTSGo

Le terrain de jeu de l'application -la carte du campus- a été obtenu en transformant une image (fig 10) obtenue à partir de google map, en texture sur Unity3D. Cette texture a ensuite été utilisée sur un GameObject parallélépipède rectangle.

Au niveau du programme, il y deux boucles principales : l'une pour mettre à jour la position au serveur et l'autre pour récupérer les questions du serveur régulièrement.

Dans la boucle (processus) de localisation, il est prévu d'utiliser éventuellement le module du groupe géolocalisation. Cependant, n'étant pas encore disponible, il est substitué à la localisation GPS qui permet à l'application de recevoir la position GPS du smartphone utilisé chaque trois secondes et demi. Pour utiliser la localisation GPS, il faut demander une permission à l'utilisateur. En utilisant un Timeout, le programme va créer une requête avec la position qui vient d'être mise à jour par la méthode getLastKnownLocation() et va l'envoyer au serveur sans arrêt [6]. Bien sûr, cette position doit aussi être envoyée au processus principal pour faire bouger le joueur sur la carte.

Dans la boucle (processus) de question, Il faut demander au serveur d'envoyer les questions autour du joueur dans un rayon de cent mètres (le groupe supervision est susceptible de le modifier) chaque demi-seconde. Cette fois-ci, ce n'est pas que l'application qui envoie quelque chose au serveur et attend la réponse, mais l'application demande quelque chose envoyée par le serveur. Donc, même si l'objet URL est encore utilisé, il faut intégrer les paramètres que le serveur demande (le User_id et le Token généralement) et configurer la méthode de requête à "GET" sans oublier d'indiquer l'adresse du domaine ("http://vps400202.ovh.net/api/questions/" par exemple)". Ce que l'application reçoit est encore un objet JSON, mais les questions autour du joueur sont déjà intégrées. Après avoir envoyé cet objet JSON au processus principal, il s'agit de retirer les questions. Sachant que le format de l'objet est connu, il suffit de faire une boucle et obtenir les valeurs en présentant les clés. Comme ça, Les questions enregistrées dans le processus principal sont mises à jour et représentées sur la carte par Unity 3D.

Le diagramme de séquence suivant résume le fonctionnement de la géolocalisation :

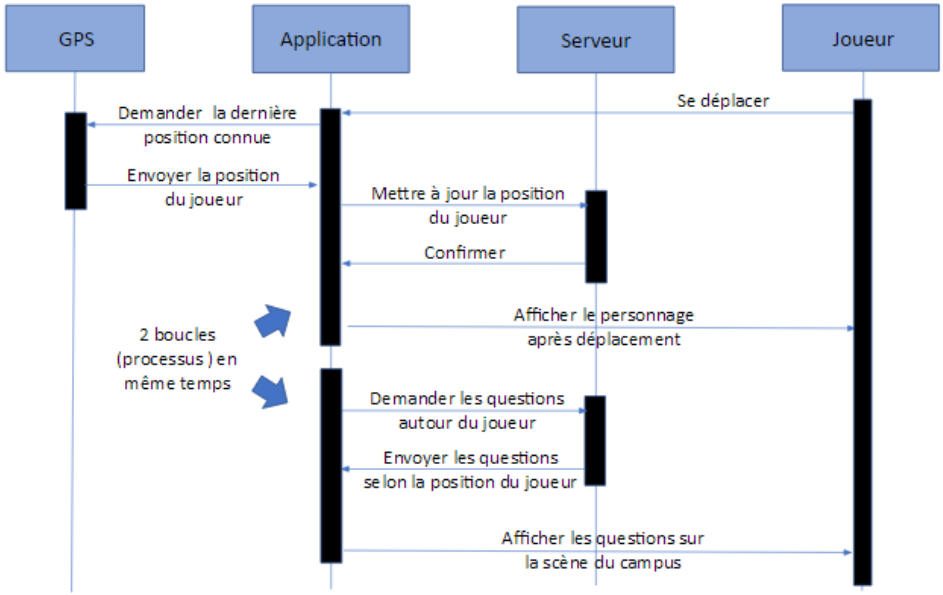

Figure 11 : diagramme de séquence de géolocalisation

3.2.3 Critère de validation

Le cahier des charges (cf annexe 1) spécifie quatre critères à valider :

- Récupérer la position : le GPS du smartphone de l'utilisateur nous envoie sa position toutes les 3,5 secondes. Le premier critère est donc validé.
- Mettre à jour la position du joueur et mettre à jour les questions : Le délai de récupération des données sont validées, ils sont de 3,5 secondes. Nous recevons bien un code de confirmation 200 lorsque le serveur reçoit la position et un code d'erreur 401 dans le cas contraire.
- Déplacer l'avatar : d'après les tests que nous avons réalisé sur notre application, l'écart entre la position GPS et celle de l'avatar est de moins de 25 mètres, ce qui valide le critère associé.

3.3 REPONDRE A UNE QUESTION

Rédacteur : Yuhao LIU Relecteur GAUTIER Lucas

3.3.1 Point de vue utilisateur

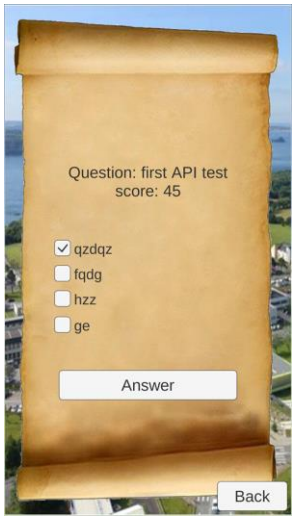

Figure 12 : écran de réponse à une question

S'il y a une question dont la position est assez proche de celle du joueur, l'action de répondre à la question est valide. Sinon, l'application va afficher un message d'erreur pour informer le joueur qu'il n'y a pas de question autour de lui. Considérons le premier cas : Jean arrive donc sur l'écran de réponse à une question (fig. 12). Il a ensuite le choix entre quatre réponses différentes. Si Jean choisit la bonne réponse, le message « bingo » s'affiche en bas de l'écran et son score augmente. La question n'est plus accessible par la suite, sur la carte, pour tous les utilisateurs de MTSGo. Dans le cas contraire, le message « What a pity » s'affiche, le score de Jean reste le même et la question ne disparaît pas : seul Jean n'y aura plus accès. Dans les deux cas, le joueur revient ensuite vers l'écran principal.

3.3.2 Point de vue développeur

Quand le joueur rencontre une question, il peut appuyer sur le bouton « answer » pour y répondre. À ce moment-là, l'application va appeler une classe dans laquelle la position du joueur est comparée avec celle de toutes les questions enregistrées dans le processus principal. Quand le joueur est suffisamment proche d'une question, appuyer sur le bouton va l'emmener à l'interface d'événement. Avant de passer à cette interface, il faut mettre le User_id, le Token, l'identité de la question, le corps de la question et les quatre réponses possibles dans une variable.

Dans l'interface d'événement, au lieu d'utiliser le CheckBox qui permet au joueur de choisir plusieurs réponses d'une question, on utilise le RadioBotton et RadioGroupe pour limiter les choix du joueur à une seule réponse parmi les quatre proposées. Quand le joueur choisit une réponse et appuie sur le bouton, son choix va être collectionné et enregistré par un numéro. Ce numéro doit être inclus dans la requête que l'application envoie au serveur (fig. 13). Du côté du serveur, il va trouver la bonne réponse avec l'identité de la question que l'application lui fournit et vérifier si la réponse que le joueur choisit est correcte ou pas. Après, il va envoyer le code de réponse correspondant à l'application pour informer le joueur.

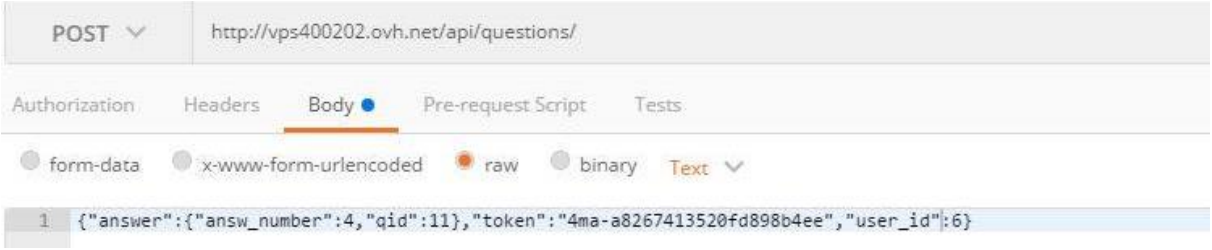

Figure 13 : exemple de chaîne json de réponse à une question

Le diagramme de séquence suivant résume le fonctionnement de réponse à une question :

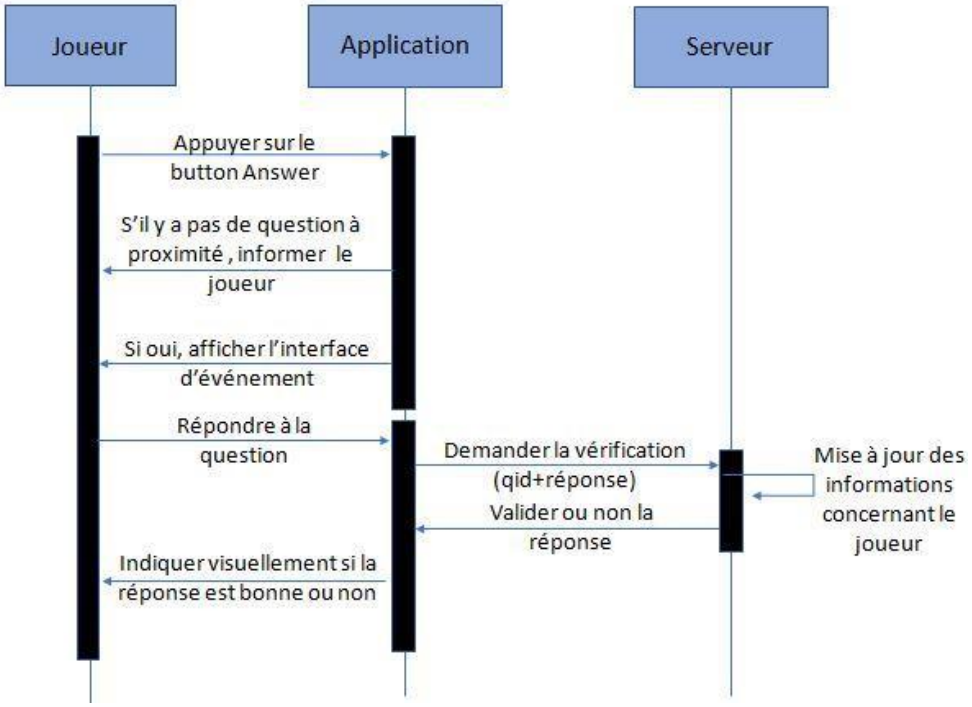

Figure 14 : diagramme de séquence de réponse à une

3.3.3 Critère de validation

Le cahier des charges spécifie trois fonctions techniques à réaliser :

- Représenter les questions : Les questions mises à jour sont rafraîchies et représentées toutes les 3,5 secondes. Le critère est validé.
- Répondre à une question : le joueur peut répondre à une question lorsqu'il est à moins de 10 « units » sur Unity3d, le critère est donc validé.
- Envoyer une réponse : les codes de confirmation de bonne réponse (code 200) et d'erreur (code 402) sont correctement reçus lors de l'envoi d'une réponse.

3.4 ACCES AU SCORE, AU CLASSEMENT, ET A L'HISTORIQUE

Rédacteur : Yuhao LIU Relecteurs : GAUTIER Lucas et GAUTIER Baptiste

3.4.1 Point de vue utilisateur

Figure 15 a : écran de score et de classement

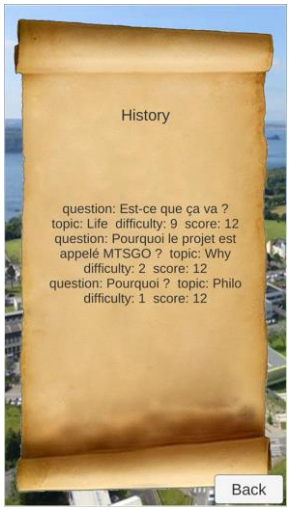

Figure 15 b : écran d'historique

À tout moment, Jean peut accéder à son score et son classement en appuyant sur le bouton associé, l'écran associé est affiché (fig. 15a). De même, il peut accéder à son historique (fig. 15b) en appuyant sur le bouton associé dans l'écran principale.

3.4.2 Point de vue développeur

Après avoir répondu à une question, l'application va retourner à la scène du campus. Dans cette scène, il peut appuyer sur le bouton en bas pour demander le score total, le rang et l'historique du serveur.

Le score, le rang et l'historique sont tous enregistrés par la base de données du serveur, il suffit de construire une requête avec la méthode "GET" puis la décoder, la mettre en page et l'afficher.

3.4.3 Critère de validation

Deux fonctions techniques sont à réaliser :

- Demander le score et le classement : les données sont bien reçues avec le code de confirmation 200
- Demander l'historique : les données sont bien reçues avec le code de confirmation 200

4. CONCLUSION

Rédacteur : GAUTIER Baptiste Relecteur : GAUTIER Lucas

Dans le cadre de notre projet nous avions pour but de réaliser la partie applicative du jeu MTSGo, ce qui comprend l'interface graphique pour l'utilisateur, les échanges de données et leur interprétation auprès du serveur, ainsi que le positionnement de l'utilisateur sur la carte du campus. Notre réalisation se situe au cœur du projet MTSGo. Nous avons dû apprendre en autonomie à développer une application Android et surmonter les difficultés que la réalisation du jeu nous a posés comme l'intégration de Unity3D pour recréer le campus et faire en sorte que l'avatar du joueur puisse s'y déplacer. À ce jour, le jeu est fonctionnel du point de vue applicatif.

L'utilisateur peut s'inscrire, se connecter, se déplacer sur le campus en répondant aux questions présentes, ainsi qu'accéder à son score et à son classement parmi les autres utilisateurs. Nous avons donc réussi à obtenir une application dont la jouabilité est semblable à celle de Pokémon Go et à respecter notre cahier des charges.

Bien que l'application MTSGo soit fonctionnelle, de nombreuses pistes d'amélioration sont présentes. Parmi elle, on note la conception de la fonction de géolocalisation améliorée que doit concevoir le groupe géolocalisation, et la réduction des délais liés au serveur. En outre, on peut encore améliorer le fonctionnement de la caméra, peaufiner toutes les interfaces de MTSGo et mieux intégrer la bande sonore dans le jeu. Enfin la liste des questions disponibles se limite pour l'instant à quelques questions tests, le groupe supervision a pour but de remédier à cet inconvénient.

À terme, si elle est améliorée par des élèves de l'IMT Atlantique les années suivantes, l'application pourrait devenir un faire-valoir du campus brestois de l'IMT Atlantique. De plus, elle pourrait permettre aux élèves de progresser dans leurs cursus grâce au jeu non seulement en MTS mais également dans les autres matières présentes à IMT Atlantique.

5. BIBLIOGRAPHIE

- [1] WIKIPEDIA. *Android Studio.* [en ligne]. Disponible sur : <https://fr.wikipedia.org/wiki/Android_Studio > (consulté le 08.05.2017)
- [2] JSON OFFICIAL.*Présentation de JSON*.[en ligne].Disponible sur : <http://www.json.org/json-fr.html > (consulté le 08.05.2017)
- [3] OKHTTP. Overview. [en ligne]. Disponible sur : <http://square.github.io/okhttp/> (consulté le 08.05.2017)
- [4] WIKIPEDIA. *Unity.* [en ligne]. Disponible sur : <https://fr.wikipedia.org/wiki/Unity_(moteur_de_jeu) > (consulté le 10.05.2017)
- [5] ANDROID DEVELOPER. The Activity Lifecycle. [en ligne]. Disponible sur : <https://developer.android.com/guide/components/activities/activity-lifecycle.html > (consulté le 08.05.2017)
- [6] RETO Meier. *Android 4 : développement d'applications avancées.* Pearson Education France, 4 Sept. 2012, 836 pages.

6. GLOSSAIRE

- AR (Augmented Reality) : superposition de la réalité et d'éléments (sons, images 2D, 3D, vidéos, etc.) calculés par un système informatique en temps réel.
- GPS (Global Positioning System) : système de géolocalisation fonctionnant au niveau mondial et reposant sur l'exploitation de signaux radio émis par des satellites dédiés.
- HTTP (HyperText Transfer Protocol) : protocole de communication client-serveur développé pour le World Wide Web.
- JSON (JavaScript Object Notation) est un format de données textuelles dérivé de la notation des objets du langage JavaScript. Il permet de représenter de l'information structurée
- URL (Uniform Resource Locator) : chaîne de caractères utilisée pour adresser les ressources d'internet.

Annexe

Fonctions	Critères	Niveau d' exigence	Importance /5
FT1:inscrire l' utilisateur auprès du serveur	code de confirmation du serveur	code 200/401	5
FT2:authentifier l' utilisateur auprès du serveur	code de confirmation du serveur	code 200/403	5
FT3:récupérer la position du joueur	période de mise à jour	<= 10s	3
FT4:mettre à jour la position du joueur	code de confirmation du serveur	code 200/401	5
	période d'envoi	<= 10s	3
FT5:faire bouger l'avatar du joueur	erreur de la position GPS et celle de l'avatar	50m	5
FT6:récupérer les questions	code de confirmation du serveur	code 200	5
	période de réception	<= 10s	3
FT7:représenter les question	période de rafraîchissement	<= 10s	5
FT8:répondre à une question	distance entre l'avatar et la question	<= 10 units en Unity	5
FT9:envoyer une réponse	code de confirmation du serveur	code 200/402	5
FT10:demander le score et le rang	code de confirmation du serveur	code 200	4
FT11:demander l'historique	code de confirmation du serveur	code 200	2
CT1:développement de l'application	logiciels utilisés	Android Studio/ Unity	
CS1:compatibilité avec le portable	version Android	>= 4.1	4
CS2:autorisation de l'utilisateur	position	enable	5
CS3: stockage	mégabit	<= 100M	2

Annexe 1 : Tableau des fonctions et contraintes de MTSGo

Suivi de projet

Notre groupe a su respecter les deadlines que nous nous étions imposé (cf suivi de projet MTSGo en annexe 2 ci-dessous). Cependant, d'une part, la prise en main des outils de développement fût plus longue que prévue. Mais d'autre part, dès que notre groupe eut finit cette tâche, nous avons su progresser sur le projet de façon plus rapide que prévue, ce qui nous a permis de rendre le premier prototype de l'application en avance.

Etant donné que nous avons dû refaire une seconde fois les interfaces utilisateurs, l'avance que nous avions gagnée précédemment, nous a été utile pour tenir les délais quant au rendu du second prototype de l'application.

Annexe 2 : Diagramme de Gantt de la partie applicative de MTSGo

MTSGo-partie applicative

Diagramme de Gantt

Nom	Date de début	Date de fin
Attribution des projets, formation des groupes	27/01/17	27/01/17
Première réunion avec les tuteurs	30/01/17	01/02/17
Rendu des rapports, poster et livrable	24/02/17	29/06/17
Dépôt du cahier des charges	24/02/17	24/02/17
Dépôt rapport V2	07/06/17	07/06/17
Dépôt poster	13/06/17	13/06/17
Soutenance poster	28/06/17	29/06/17
rendu du livrable	21/06/17	21/06/17
Prise en main des outils de développement et de gestion de projet	02/02/17	31/03/17
Prise en main d'Unity	02/02/17	20/03/17
Prise en main d'Unity (En réalité)	02/02/17	31/03/17
Suivie de tutoriaux en C# et en Java	02/02/17	20/03/17
Suivie de tutoriaux en C# et en Java (En réalité)	02/02/17	31/03/17
Formation sur Gantt	22/02/17	22/02/17
Formation Android	01/03/17	01/03/17
Gestion de la jouabilité de l'application (Yuhao LIU)	21/03/17	15/05/17
Gérer l'inscription de l'utilisateur	21/03/17	31/03/17
Gérer l'inscription de l'utilisateur (En réalité)	21/03/17	06/04/17
Gérer l'authentification de l'utilisateur auprès du serveur	07/04/17	14/04/17
Gérer l'authentification de l'utilisateur auprès du serveur (En réa...	07/04/17	07/04/17
Faire bouger le joueur en fonction de sa géolocalisation	21/03/17	21/04/17
Faire bouger le joueur en fonction de sa géolocalisation (En réa...	03/04/17	21/04/17
permettre au joueur de répondre aux questions	11/04/17	12/05/17
permettre au joueur de répondre aux questions (En réalité)	11/04/17	27/04/17
Rendu du premier prototype de l'application	15/05/17	15/05/17
Rendu du premier prototype de l'application (En réalité)	28/04/17	28/04/17
Réalisation de l'interface graphique (Lucas GAUTIER)	17/04/17	05/06/17
création des interfaces utilisateur	17/04/17	26/05/17
création des interfaces utilisateur (En réalité)	17/04/17	26/04/17
création de l'environnement de réalité augmentée	24/04/17	02/06/17
création de l'environnement de réalité augmentée (En réalité)	24/04/17	02/06/17
rendu du second prototype de l'application	05/06/17	05/06/17
rendu du second prototype de l'application (En réalité)	05/06/17	05/06/17
Gestion des échanges de données (Baptiste GAUTIER)	02/02/17	20/06/17
Serveur-application (questions, classement...)	15/05/17	20/06/17
Serveur-application (questions, classement...) (En réalité)	30/03/17	19/05/17
Géolocalisation	24/04/17	20/06/17
Géolocalisation (En réalité)	24/04/17	07/06/17
Synchronisation avec les autres groupes	02/02/17	20/06/17
Synchronisation avec les autres groupes (En réalité)	02/02/17	07/06/17

Technopôle Brest-Iroise
CS 83818
29238 Brest Cedex 3
France
+33 (0)2 29 00 11 11
www.telecom-bretagne.eu