**Plongez dans l’algorinthe : Guide de développement**

[**Étapes de l’activité**](#_hutjvxurub0a) **4**

[Connection et paramétrisation](#_qho5ungyi491) 4

[Tutoriel](#_4xgandb437ly) 4

[Une ronde](#_fz68f7d6y7kf) 4

[Intermission](#_d24qb7rj3ioj) 4

[**Communication entre les différents modules**](#_5sesgne70pyr) **5**

[Communication avec la base de données](#_oebm6x4wdjoy) 5

[Échanges avec le serveur](#_s3afotrkwiwm) 5

[Connection au server sur réseau local](#_qjnhv8gxi9gh) 5

[Communication entre l’ordinateur et les tablettes](#_fdztk5nr2e9w) 5

[Informations envoyées par l’ordinateur vers la tablette](#_9mvppqm1e4v0) 5

[Phase du jeu](#_tit6zeyrgtg5) 5

[Information sur le labyrinthe](#_xo4tcs86ght) 6

[Information sur l’algorithme](#_iwv6k4z7nji4) 6

[Information de pairage](#_hdja8h47z3u8) 6

[Informations envoyées par les tablettes vers l’ordinateur](#_xp348fpov34t) 6

[Statut](#_o8hf6sql1kqe) 6

[Information de l'appareil](#_udt3s4o83a90) 6

[Communication entre l’ordinateur et les casques](#_mvhae3jp0rrs) 6

[Information envoyée par l’ordinateur vers les casques](#_1xphypopb1jl) 6

[Phase du jeu](#_fuo9h0xnfe2z) 6

[Information sur le labyrinthe](#_6r6b9rxxcfv) 6

[Information de pairage](#_9y2zj7y4d5au) 6

[Information envoyé par les casques vers l’ordinateur](#_l3ueme80kwlr) 7

[Statut](#_qdr7fw3zzn15) 7

[Information de l'appareil](#_ccof1nawmd0u) 7

[Information d’événement](#_bnqt32bormq3) 7

[Communication entre les tablettes et les casques](#_8do5gp62mvsv) 7

[Information envoyée par la tablette vers le casque associé](#_j79xusre1f5r) 7

[Directives](#_pyaxkb2g61d2) 7

[Requête d’information du jeu](#_llmqvb9tlmyi) 7

[Requête de retour à la position de divergence](#_bgn4g128plun) 7

[Information envoyée par le casque vers la tablette associée](#_bfqalt9w66va) 8

[Informations du jeu](#_pcio0lc8lt2q) 8

[Information de l’équipe](#_kdwgvw6hb1sf) 8

[**Pairage des appareils**](#_6pfwpju3x7lz) **8**

[Enregistrer une paire d’appareils](#_xjnzyn134lqb) 8

[Connexion des paires](#_kof0b7ru5mjo) 8

[**Déroulement d’une ronde de jeu**](#_6jwdtl69xpj1) **9**

[Génération du labyrinthe](#_kltc91gsmvb7) 9

[Les interactions pour le casque](#_9gjvjvtsrheb) 9

[Les déplacements du casque](#_2d95t6khov38) 9

[Le marquage des tuiles](#_z8yufaud1ilw) 9

[La boussole](#_wnnfjwaava3v) 10

[L’indicateur de distance](#_1pxslcylvlg6) 10

[Les interactions pour la tablette](#_cpo5qagm3n88) 10

[Les déplacements de la tablette](#_gbisiiii2c9a) 10

[Les directives de la tablette](#_5v3fxf1n2fvu) 10

[La boussole](#_1pjm2189w8jj) 10

[Le respect de l’algorithme](#_7p1n48264j0p) 11

[Le marquage des tuiles](#_9p6onje2iq0z) 11

[Retour au point de divergence](#_ma2pohuu2l5) 11

[Les algorithmes](#_z63jdtj3qufl) 11

[**Reconnexion d’un appareil**](#_3oglra3l5v7j) **12**

[Reconnexion au niveau du serveur](#_u2qdbff36omf) 12

[Reconnexion de la tablette](#_njp3bz173i81) 12

[Reconnexion du casque](#_qwbqg8akgngl) 13

[**Particularités**](#_fzo7alz1skad) **13**

[Compilation du projet](#_ofmkij2qedqh) 13

[Problèmes connus](#_m1dza4usk9v3) 13

[Réinitialisation de la base de donnée](#_e0jx6y42ijr5) 14

[Identifiant du compte l’utilisateur promo-science](#_yn61ixggf4vk) 14

[Liens utiles](#_vngqx7b3eatk) 14

# Étapes de l’activité

L’activité se déroule en enchaînant différentes étapes débutant par la connection et la paramétrisation. Par la suite, les étapes de tutoriel, de rondes et d’intermission sont répétées jusqu’à la fin de l’activité.

## Connection et paramétrisation

À cette première étape, on établit les liens de communications entre les différents modules. L’ordinateur possède une interface pour faire un suivi des appareils connectés appelée *le Lobby*. Lorsque voulut, le server (ordinateur) peut démarrer l’accès au tutoriel ou la prochaine ronde.

## Tutoriel

Le tutoriel permet au joueur de se familiariser avec les contrôles de la tablette et du casque. Ils sont donc libre de circuler dans un labyrinthe tutoriel avec toutes les mécaniques de jeux présentes. Cette scène a pour but d’aider les joueurs à expérimenter ce qui leur a été préalablement expliqué. Du tutoriel, les joueurs passent à une première ronde ou en intermission, lorsque dicté par le serveur.

## Une ronde

Une ronde est démarrée par serveur (ordinateur) manuellement. Une manche est constituée d’un parcours dans le labyrinthe pour lequel le joueur avec le casque exécute des actions et les joueurs avec la tablette envoient des directives. Lorsque l’équipe atteint la fin du labyrinthe de façon adéquate, elle est transférée à la scène d’intermission. Le serveur peut également forcer l’arrêt d’une ronde de jeu et renvoyer les joueurs à l’intermission.

## Intermission

Scène avec un visuel minimum, pour le casque et la tablette, utilisée pour attendre entre les étapes et pendant les reprises vidéos. Lorsque dicté par le serveur, les casques et tablettes transitionnent vers la scène du tutoriel ou le début d’une ronde.

# 

# Communication entre les différents modules

## Communication avec la base de données

La base de données, de type *SQLite*, est située sur l’ordinateur et elle est locale. C’est donc l’ordinateur qui gère la circulation de l’information provenant de la base de données. Ce sera plus précisément sa classe *SQLiteUtilities* qui s’occupera de la communication avec la base de données.

## Échanges avec le serveur

Le serveur est géré par l’ordinateur. Il sera démarré lorsque le lobby du jeu sera activé. La connection au serveur s’effectue par connexion locale directe en connaissance de l’adresse du serveur. Les tablettes et les casques se connectent au serveur par la classe de *NetworkManager*. Après la connexion d’un appareil, le serveur générera un objet *Player* pour cet appareil qui sera présent sur le serveur et le client. Cet objet permet au client de communiquer avec le serveur car il a l'autorité d’envoyer des commandes au serveur. La classe serveur communique avec ses clients à l’aide appel de procédure à distance ciblée

(*TargetRpc)* qui vise un client spécifique.

## Connection au server sur réseau local

Le processus de découverte sur réseau local s’exécute à l’aide de la classe *NetworkDiscovery* de Unity adapté à cette situation. Elle permet d’obtenir l’adress IP du serveur sur le réseau local afin de pouvoir faire une requête de connection à celui-ci.

## Communication entre l’ordinateur et les tablettes

La communication entre l’ordinateur et la tablette s’effectue par le biais du serveur sur l’ordinateur qui envoie des procédures ciblées (*TargetRpc)* et par l'instance *Player* de la tablette qui envoie des commandes au serveur.

### Informations envoyées par l’ordinateur vers la tablette

#### Phase du jeu

Permet de communiquer au client lorsque l’étape de jeu est changé par le serveur entre le lobby, la période de test, une ronde, l’intermission et la fin du jeu.

#### Information sur le labyrinthe

Information puisée de la base de données sur la carte du labyrinthe.

#### Information sur l’algorithme

Information puisée dans la base de données sur l’algorithme associé à une équipe.

#### Information de pairage

Information (adresse réseau) permettant à la tablette de démarrer la communication avec le casque associé.

### Informations envoyées par les tablettes vers l’ordinateur

#### Statut

Permet de communiquer au serveur l’état du client, entre autres s'il est prêt à procéder à la prochaine phase du jeu.

#### Information de l'appareil

Permet de communiquer au serveur les informations de l’appareil dont son type, nom et numéro d’identifiant unique.

## Communication entre l’ordinateur et les casques

La communication entre l’ordinateur et la tablette s’effectue par le biais du serveur sur l’ordinateur qui envoie des procédures ciblées (*TargetRpc)* et par l'instance *Player* du casque qui envoie des commandes au serveur.

### Information envoyée par l’ordinateur vers les casques

#### Phase du jeu

Permet de communiquer au client lorsque l’étape de jeu est changé par le serveur entre le lobby, la période de test, une ronde, l’intermission et la fin du jeu.

#### Information sur le labyrinthe

Information puisée dans la base de données sur la carte du labyrinthe.

#### Information de pairage

Information (adresse réseau) permettant au casque de démarrer la communication avec la tablette associée.

### Information envoyé par les casques vers l’ordinateur

#### Statut

Permet de communiquer au serveur l’état du client, entre autres s'il est prêt à procéder à la prochaine phase du jeu.

#### Information de l'appareil

Permet de communiquer au serveur les informations de l’appareil dont son type, nom et numéro d’identifiant unique.

#### Information d’événement

Information d’un événement, envoyée au serveur pour qu’elle soit enregistrée dans la base de données. Un événement est constitué du moment où il est survenu ainsi que d’un numéro représentant un action ou une directive.

## Communication entre les tablettes et les casques

La communication entre les tablettes et les casques s'effectue seulement entre les paires associées lors de l’étape de pairage. Elle est gérée par la classe de communication respective de chacun des appareils. Le casque joue le rôle de serveur avec la classe *MessageServer.* La tablette est le client et communique avec sa classe *MessageClient*. Celle-ci se charge de de débuter la communication par réseau local direct en se connectant au serveur du casque à l’aide de l’adresse fournie par le serveur (ordinateur).

### Information envoyée par la tablette vers le casque associé

#### Directives

Les directives envoyées par la tablette sont les indices pour le joueur portant le casque. Elles peuvent prendre la forme d’une direction.

#### Requête d’information du jeu

Cette requête permet de signifier au casque que la tablette est prête à recevoir les informations de la ronde courante pour synchroniser les éléments du jeu avec ceux du casque.

#### Requête de retour à la position de divergence

Cette requête permet d’envoyer une demande de retour au point de divergence à l’utilisateur du casque.

### Information envoyée par le casque vers la tablette associée

#### Informations du jeu

Les information du jeu sont envoyées lorsqu’il y a un changement, que ce soit un déplacement, le changement de couleur d’une tuile, le respect de l’algorithme ou la complétion du labyrinthe engendré par les actions de l’utilisateur du casque.

#### Information de l’équipe

L’information de l’équipe est envoyée à la tablette par le casque pour s’assurer que la tablette soit toujours associée à la même équipe que son casque associé.

# Pairage des appareils

Le pairage d’un couple formé d’un casque et d’une tablette s’effectue en deux étapes:

1) On doit d’abord enregistrer la paire d’appareils dans la base de données sur le serveur (ordinateur).

2) Ensuite lorsque l’on veut participer à l’activité avec ces appareils, le serveur utilisera les paires enregistrées pour aider à établir la connexion entre les deux appareils.

## Enregistrer une paire d’appareils

L’enregistrement d’une paire d’appareils s’effectue en ouvrant l’application de pairage sur le casque et la tablette que l’on veut jumeler ensemble, ainsi que l’application serveur (ordinateur). Pour ce faire, les appareils se connectent au serveur à l’aide de la découverte par réseau local et envoie leur identifiant unique par message. Le serveur accepte seulement une tablette et un casque à la fois pour le pairage. Lorsque les deux appareils ont communiqué, il supprime les anciennes données de jumelage liées à ces appareils et en crée de nouveaux dans la base de données pour la nouvelle liaison.

## Connexion des paires

La connexion entre les paires s’effectue après la connection au serveur. Elle se déclenche lorsque le statut d’un appareil change pour la recherche d’une paire. Le serveur va alors chercher l’information pour l’appareil et se met à chercher en boucle pour son partenaire associé. Si l’appareil n’a pas de partenaire enregistré dans la base de données, son statut changera pour indiquer qu’il n’est associé avec aucun partenaire. Si son partenaire est connecté au serveur, la tablette obtiendra, du serveur, l’adresse IP de son casque associé et entamera le processus pour se connecter directement avec son partenaire.

# Déroulement d’une ronde de jeu

## Génération du labyrinthe

Sur le casque comme sur la tablette, la génération du labyrinthe est gérée par la classe *GameLabyrinth* qui s’occupe de faire prendre forme au labyrinthe présenté sous un format de tableau d’entiers fourni par l’objet *ScriptableLabyrinth*. Pour ce faire, chaque type d’élément du labyrinthe possède un identifiant unique et existe en *Prefab*. L’élément voulu est alors instancié à sa position indiquée dans la carte du labyrinthe. Le cumul de toutes ces tuiles forme le labyrinthe. La classe *GameLabyrinth* est par la suite la référence pour les autres modules qui ont besoin d’informations sur le labyrinthe. Cette forme de surcouche permet de recevoir du serveur le prochain labyrinthe sans pour autant être obligé de le générer immédiatement. La génération visuelle du labyrinthe est enclenchée par le changement d’état du jeu.

## Les interactions pour le casque

### Les déplacements du casque

Les déplacement du joueur portant le casque se réalisent à l’aide du contrôleur. Ils sont gérés par la class *HeadsetControls*. On peut changer la direction de marche en glissant le doigt vers la droite ou la gauche sur le pad tactile. On peut appuyer sur le pad tactile pour avancer dans cette direction. Les mouvements sont envoyés par message à la tablette sous forme de rotation et position. Pour aider à la synchronisation, ces messages sont envoyés à un débit fixe lorsque leur valeur change. L’activation des contrôles est gérée par le biais du *ScriptableObjectControls*.

### Le marquage des tuiles

Le joueur possédant le casque doit marquer son chemin puisque cette démarche est incluse dans les étapes des algorithmes. Le marquage de couleur rouge modifie la couleur de la tuile que le joueur quitte tandis que le jaune et le retrait des marques modifie la couleur de celle que le joueur est présentement dedans et les futures tuiles sur lesquelles il se déplace.

Pour ce faire, la classe *PaintingColorDisplay* affiche, au joueur du casque, un indicateur représentant la couleur qu’il a présentement de selectionnée pour le marquage. Le reste des interactions est géré par la classe *HeadsetControls* qui s’occupe d'interagir avec la classe *FloorPainter* que possède chaque tuile peinturable . Le joueur peut changer sa couleur en appuyant sur la gâchette pour circuler entre la couleur de marquage jaune et rouge. Il peut également maintenir la gâchette pendant une seconde pour passer en mode retrait des marques. Lorsque que le joueur se déplace, les tuiles sont automatiquements peinturées de cette couleur. Ce processus évite que le joueur doive peindre chaque case explicitement mais conserve tout de même l’élément de logique derrière le mécanisme. Les tuiles peinturées par le casques sont envoyée à la tablette en système de coordonnées et de couleur.

### La boussole

Géré par la classe *Compass*, la boussole indique les directions des points cardinaux. Pour ce faire, le nord de la boussole pointe toujours dans la bonne direction même si le joueur tourne et regarde une autre direction. Cet indicateur visuel aide le joueur à respecter les algorithmes.

### L’indicateur de distance

Géré par la classe *DistanceScanner*, l’indicateur de distance permet d’informer le joueur de la distance entre les positions adjacentes et la fin, ou de la longueur en ligne droite dans une direction en nombre de tuiles. Pour ce faire, le scanner envoie un rayon dans la direction où le joueur avec le casque regarde. Si il regarde un objet, possédant l’étiquette *Wall*, qui est aligné avec le joueur, il retourne la distance en ligne droite et l’affiche de couleur cyan. Si le joueur regarde une tuile au sol qui est adjacente ou en dessous de lui avec l’étiquette *Floor*, la distance entre cette tuile et la fin est retournée en nombre de tuiles. Cette distance est affichée de couleur lime. L’endroit où le joueur regarde est indiqué sur la surface visée par un cercle rouge pour aider le joueur visuellement.

## Les interactions pour la tablette

### Les déplacements de la tablette

Les déplacements de la tablette sont gérées par la classe *TableControls* qui réagit au positions et aux rotations reçues par messages. Pour rendre les mouvements plus fluides, la classes exécute une interpolation simple du mouvement en prenant en considération la fréquence d’envoi de l’information. La fréquence d’envoie est fixe pour faciliter le processus. Cette interpolation introduit un délai entre le casque et la tablette qui s’additionne au délai de réception des messages.

### Les directives de la tablette

Les directives de la tablette sont envoyées au casque en appuyant sur les boutons respectifs. Ce processus est géré par la classe *DirectiveControls* qui se charge de changer la valeur de l'objet scriptable associé à la directive. Cette valeur est communiquée par message au casque qui réagit au changement et affiche la directive à l’aide de la classe *DirectiveDisplay*. La même classe exécute également l’affichage pour la tablette.

### La boussole

Géré par la classe Compass, la boussole indique les directions des points cardinaux. Pour ce faire, le nord de la boussole pointe toujours dans la bonne direction même si le joueur tourne et regarde une autre direction. Cet indicateur visuel aide au joueur à respecter les algorithmes.

## Le respect de l’algorithme

La tablette affiche une barre de progression indiquant le taux respect de l’algorithme. Le taux respect de l’algorithme est envoyé par le casque lorsqu’il change pendant une ronde de jeu. Le taux doit être de 100% pour que les joueurs puissent compléter le labyrinthe. Tous les éléments du respect de l’algorithme sont gérés par la classe *AlgorithmRespect*. Lorsque le joueur du casque se déplace dans le labyrinthe, la classe se charge de vérifier si la nouvelle position est bel et bien la prochaine étape dans l’algorithme courant et que les tuiles ont été marquées de la bonne couleur. Si ce n’est pas le cas, le taux de respect diminue et les étapes cessent d’être suivies pour le respect de l’algorithme. En d’autres mots, le joueur diverge. Plus le joueur s’éloigne du point où il a divergé, plus la valeur de respect diminue et s’il s’en rapproche, l’effet contraire se produit. Lorsque le joueur retourne au bon endroit et ont corrigé leurs erreurs, il peut continuer l’exécution de l'algorithme. Chaque fois que le joueur passe de l’état de divergence, son compteur d’erreurs augmente.

### Le marquage des tuiles

Le marquage affecte directement le respect de l’algorithme et peut causer une divergence lorsque le joueur n’a pas marqué la tuile qu’il quitte de la bonne couleur. De plus, lorsque le joueur diverge, les tuiles marquées sont enregistrées comme des erreurs qui doivent être corrigées avant de pouvoir reprendre au point de divergence. Chaque tuile marquée comme erreur affecte le taux de respect de l’algorithme.

### Retour au point de divergence

Lorsque le joueur diverge de l’algorithme, un bouton apparaît pour les utilisateurs de la tablette afin d’envoyer une requête à l’utilisateur du casque pour retourner à l’endroit où il a divergé. Cette option à pour but d’aider les joueurs lorsqu’ils sont totalement perdus dans l’algorithme et elle engendre une pénalité d’erreur. Lorsque la requête est acceptée par le casque, celui-ci envoie les tuiles à remettre de la bonne couleur et la position à retourner, soit la position où ils avaient commencé à diverger, à la tablette et le joueur peut reprendre l’algorithme en ayant récupéré le taux maximal de respect de l’algorithme.

### Les algorithmes

Les classes d’algorithmes servent à générer leurs étapes respectives pour les labyrinthes.

Il y a tout d’abord l’algorithme de la main droite dont sa classe est *RightHandAlgorithm*. L’algorithme priorise toujours les déplacements vers la droite du joueur en marquant son chemin comme visité. Il rebrousse chemin en cas de cul de sac en marquant son passage comme visité et non concluant.

Il y a ensuite l’algorithme du plus court chemin à vol d’oiseau, géré par la classe *ShortestFlightDistanceAlgorithm*. Celui-ci priorise les déplacements dans la direction pour laquelle la distance entre l’endroit et la fin est la plus courte en marquant son chemin comme visité. En cas d’égalité, il priorise le nord, suivit de l’est, du sud et enfin de l’ouest. Lorsqu’un cul de sac est atteint, il rebrousse chemin jusqu’à la dernière intersection en marquant son passage comme visité et non concluant.

Il y a également l’algorithme de chemin le plus long en ligne droite avec sa classe *LonguestStraightAlgorithm*. Celui-ci priorise les déplacements pour lesquels on peut traverser la plus grande distance en ligne droite en marquant son chemin comme visité. En cas d’égalité, il priorise le nord, suivit de l’est, du sud et enfin de l’ouest. Lorsqu’un cul de sac est atteint, il rebrousse chemin jusqu’à la dernière intersection en marquant son passage comme visité et non concluant.

Enfin il y a l’algorithme directionnel représenté par la classe *StandardAlgorithm* qui priorise le nord, suivit de l’est, du sud et enfin de l’ouest en marquant son chemin comme visité. Lorsqu’un cul de sac est atteint, il rebrousse chemin jusqu’à la dernière intersection en marquant son passage comme visité et non concluant.

# Reconnexion d’un appareil

Il se peut que pendant l’activité, un ou plusieurs appareils perdent momentanément la connexion au serveur pour une raison ou une autre (exemple: l'application peut être accidentellement fermée). Pour prendre en compte ce genre de situations, l’application possède un mécanisme de reconnection.

## Reconnexion au niveau du serveur

La reconnexion est principalement gérée au niveau du serveur (ordinateur). Pour permettre au serveur de conserver les informations nécessaires pour reprendre le bon déroulement de l’activité, les informations du serveur ainsi qu’une liste des appareils qui se sont connectés à celui-ci sont enregistrés dans la base de donnée. Ces informations sont chargées à l’ouverture du serveur pour que le serveur puisse reprendre à la bonne étape de l’activité en cas de fermeture accidentelle du serveur. Les informations de connexion sont supprimées de la base de données lorsque le serveur est fermé adéquatement et permet ainsi à la prochaine ouverture de débuter l’activité normalement sur le serveur.

Lorsque un appareil qui avait perdu la connexion au serveur s’y reconnecte, il envoie ses informations au serveur comme pour une connection régulière. À la réception de ces informations, le serveur vérifie si l’identifiant unique de l’appareil correspond à celui de l’un des appareils dans la liste de ceux qui avaient déjà été connectés lors de la même session de l’activité. Si il est bien présent, le serveur entame le processus de reconnection pour cet appareil.

## Reconnexion de la tablette

Le processus de reconnexion au serveur pour la tablette est de réinitialiser comme si elle venait de se connecter pour la première fois. La tablette est donc dirigé, selon l’étape du serveur, vers l’une des différentes phases de jeu.

Lorsque la tablette perd connection au casque, elle entame le processus de reconnection avec celui-ci. Si la reconnection servi pendant une ronde de jeu, la tablette réinitialiser la phase du jeu pour s’assurer d’avoir les bonnes informations de la ronde.

## Reconnexion du casque

Le processus de reconnexion au serveur pour le casque nécessite, contrairement à celle de la tablette, des démarches particulière lorsqu’elle survient lors d’une ronde de jeu. Pour s’assurer que le casque et les étapes enregistrés dans la base de donnée sont toujours les mêmes, on réinitialise l’état de la ronde du casque par rapport aux informations de la base de données avant de permettre au joueur de continuer la ronde.

Lorsque le casque perd la connexion avec la tablet, il attendra la reconnexion de celle-ci.

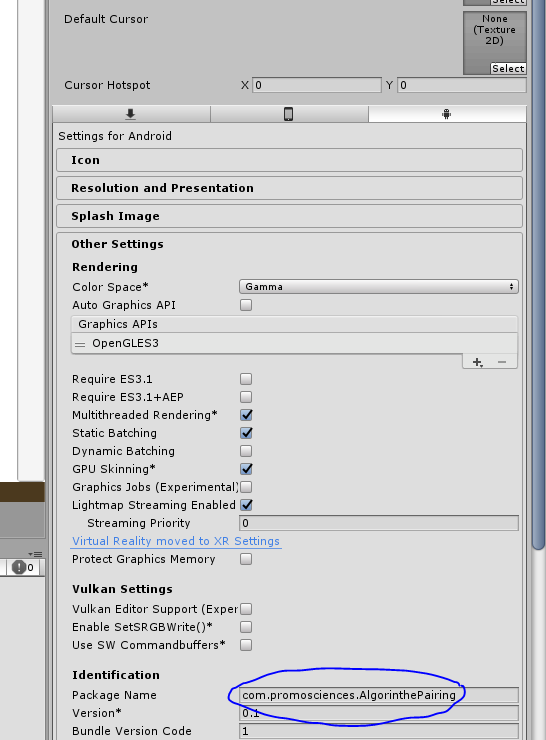
# Particularités

## Version de Unity

* 2018.2.5f1
  + Version manquant certaine fonctionnalités clés de Unity (new Input system, nested prefab workflow)
  + Impossible de mettre a jour dû aux dépendance sur `UnityEngine.Networking` (UNet), mechanisme de networking deprecated qui ne semble pas fonctionner dans >= 2018.3.
  + Course of action (remplacer UNet par api courrant)

## Compilation du projet

Pour compiler le projet sur un appareil on doit sélectionner la bonne scène parmi *Server*, *Tablet*, *Headset*, *TabletPairing* et *HeadsetPairing* ainsi que de mettre le bon *Pakage Name*  dans les options *Unity* selon la scène, soit *Algorinthe* ou *AlgorinthePairing.*  De plus, lorsque l’on compile sur la tablette, on doit mettre le paramètre de qualité android par défaut a *Very Low* et pour le casque on doit le mettre à *High.*



*Package Name*

Pour compiler sur android avec *Unity*, on doit tout d’abord préparer notre poste de travail en conséquence. Pour ce faire on doit installer le bon *SDK* et *JDK* en plus de l’option android de *Unity*.

On peut retrouver les applications compilées sur l’*Occulus Go* à l’aide de *Unity* dans le menu *Library* sous l’onglet *Unknown Sources*.

## Problèmes connus

Le casque *Oculus Go* à régulièrement des problèmes de connexions. Parfois, il ne semble pas vouloir se connecter au serveur (ordinateur). Désactiver le réseau du casque et le réactiver semble régler le problème temporairement.

Lors d’une séance de jeu si l’application de la tablette, sans redémarrer, se connecte pour une 16ème fois au casque, la connexion réseau de la tablette ne fonctionnera plus adéquatement et lancera l’erreur *Maximum host cannot exceed 16.* Redémarrer l’application de la tablette règle ce problème.

L’erreur *host id out of bound* qui survient à la connection et *connection time out* qui survient lorsque l’on perd la connection d’un appareil sont également présentes mais elles ne nuisent pas au bon fonctionnement de l’application.

## Sujet à test par des utilisateurs

1. Vitesse de déplacement (Mal de transport?)
2. Vitesse de rotation (Mal de transport?)
3. Le respect de l’algorithme est-il trop stricte?

## Idée de fonctionnalités à ajouter

1. Les applications devraient se fermer automatiquement sur tous les appareils après avoir complété la troisième et dernière ronde de jeu.

## Réinitialisation de la base de donnée

Pour réinitialiser la base de donnée complètement en cas de changement majeur, elle se situe généralement *C:\Users\[UserName]]\AppData\LocalLow\promo-sciences\...* et on doit supprimer le fichier nommé *AlgorintheDatabase.db* .

## Identifiant du compte l’utilisateur promo-science

nom : [UdesPromoScience@gmail.com](mailto:UdesPromoScience@gmail.com)

mot de pass: infostage11

## Liens utiles

<https://www.androidcentral.com/how-reset-your-oculus-go>