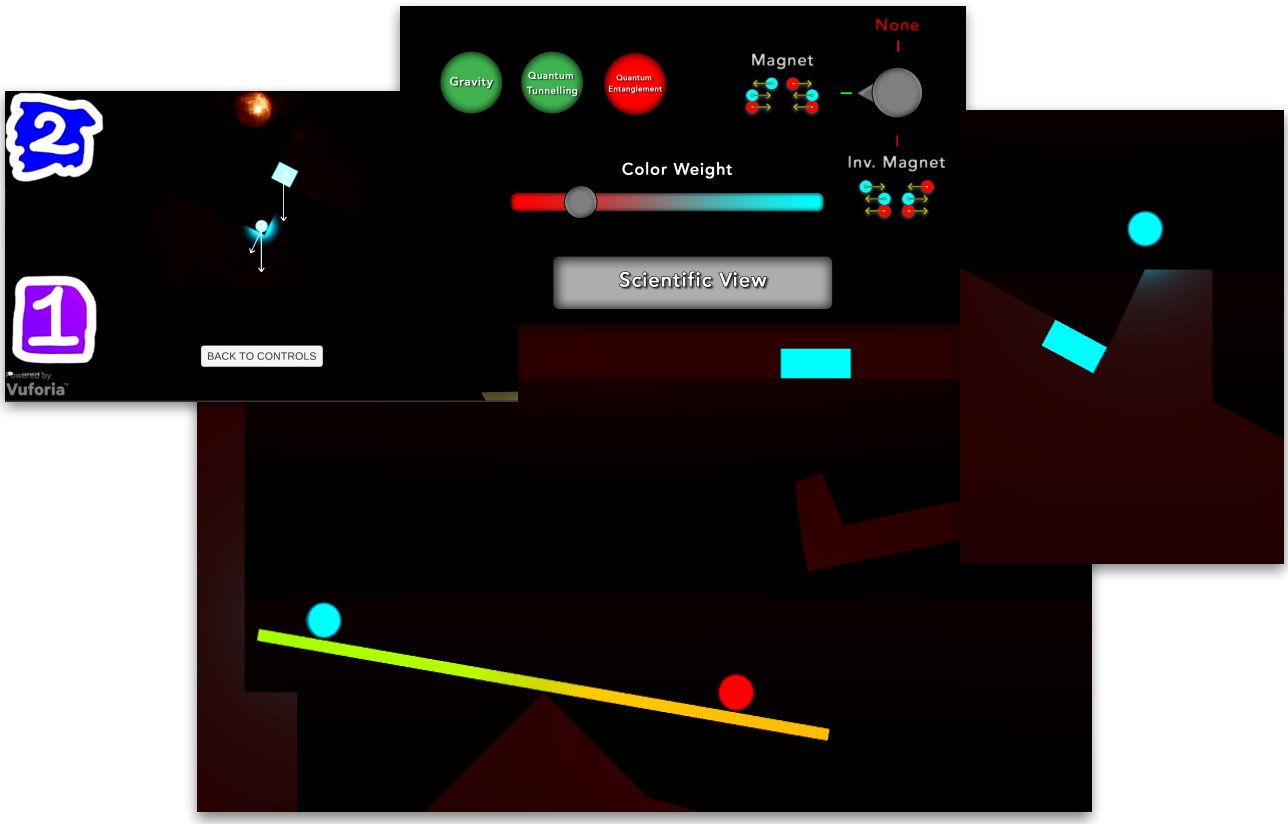
Bastien Giafferi

ENJMIN - 2015

Momentum



Images issues du prototype (www.youtube.com/watch?v=i8KHXROcr8o)

Table des matières

Introduction

Ce dossier, réalisé dans le cadre du concours d'entrée de l'ENJMIN, présente mon idée de jeu issue de mon interprétation du sujet de l'année 2015 : le « *Boson de Higgs* ».

Thème

Le Boson de Higgs, prédit en 1964 par quatre scientifiques indépendants (dont Peter Higgs, qui donnera son nom au boson) et confirmé récemment (en 2012) par le CERN, permettrait d'expliquer la masse des objets. La théorie de la physique quantique (physique des particules) sépare les particules élémentaires en deux familles : les bosons et les fermions. Les fermions sont les particules qui composent la matière alors que les bosons transportent les forces : ce sont des particules d'interaction.

Cheminement et analyse du sujet

Lors de l'annonce du sujet, je me suis tout de suite renseigné sur la physique des particules afin de mieux cerner le sujet et de pouvoir dénicher une idée originale. C'est alors qu'un ami qui étudie la physique à l'université de La Rochelle m'a parlé de la physique quantique, et plus particulièrement des principes intéressants qui en découlent (et qui sont la plupart du temps complètement opposés à la physique « traditionnelle ») :

- **l'intrication quantique** : le fait que deux systèmes soient « connectés » et qu'une action sur l'un des systèmes est immédiatement répercuté sur l'autre.
- **l'effet tunnel** : une particule peut passer à travers un obstacle s'il est suffisamment fin
- **le principe de superposition** : une particule peut avoir deux états différents en même temps, ce qui est complètement à contre courant de la physique « traditionnelle »
- **l'indéterminisme de la mesure** : comme une particule peut avoir plusieurs états en même temps, quel état allons-nous mesurer ?

Il a ensuite terminé en me citant un reportage qu'il avait vu dans lequel le commentateur expliquait que dans quarante-cinq ans, la moitié de tous ces principes seront réfutés (c'est la demie-vie du savoir).

C'est d'ailleurs comme cela que la science avance : on essaye de définir des modèles qui représentent la réalité et on se base sur les observations pour mettre à jour ces modèles. Par exemple, le LHC (le collisionneur de particules qui a mis en évidence le boson de Higgs) sera redémarré dans quelques mois pour faire des mesures plus précises du boson et éventuellement modifier le modèle standard.

J’ai donc décidé de partir sur les concepts suivants :

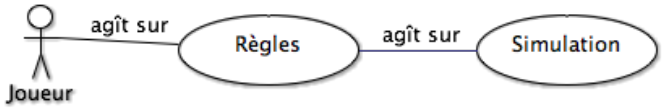
- L’incertitude du modèle (on essaye de faire correspondre le modèle à la réalité)
- L’ensemble des principes inhérents à la physique quantique
- Le côté expérimental de la science (on fait des expériences et on regarde ce que ça donne)

Le jeu vidéo permet de faire des choses impensables dans la réalité. C’est aussi pour faire un parallèle avec la contradiction physique quantique / physique classique que j’ai décidé de prendre le problème à l’envers.

Partie commune

L'idée

Dans ce jeu de simulation, ou plutôt expérience, on ne peut pas interagir directement avec les éléments du jeu. La seule possibilité de gameplay réside dans la modification du modèle physique du jeu. On simplifiera l'aspect physique en regroupant les notions de physique quantique telles que les principes vus ci-dessus et la physique plus classique avec des notions comme la gravité. Contrairement à la réalité où les scientifiques cherchent un modèle qui correspond à la réalité, on va ici choisir un modèle que la physique va appliquer. De plus, le jeu possèdera un aspect « expérimental », qui fera partie inhérente du gameplay du jeu.



L'impact du joueur sur le jeu

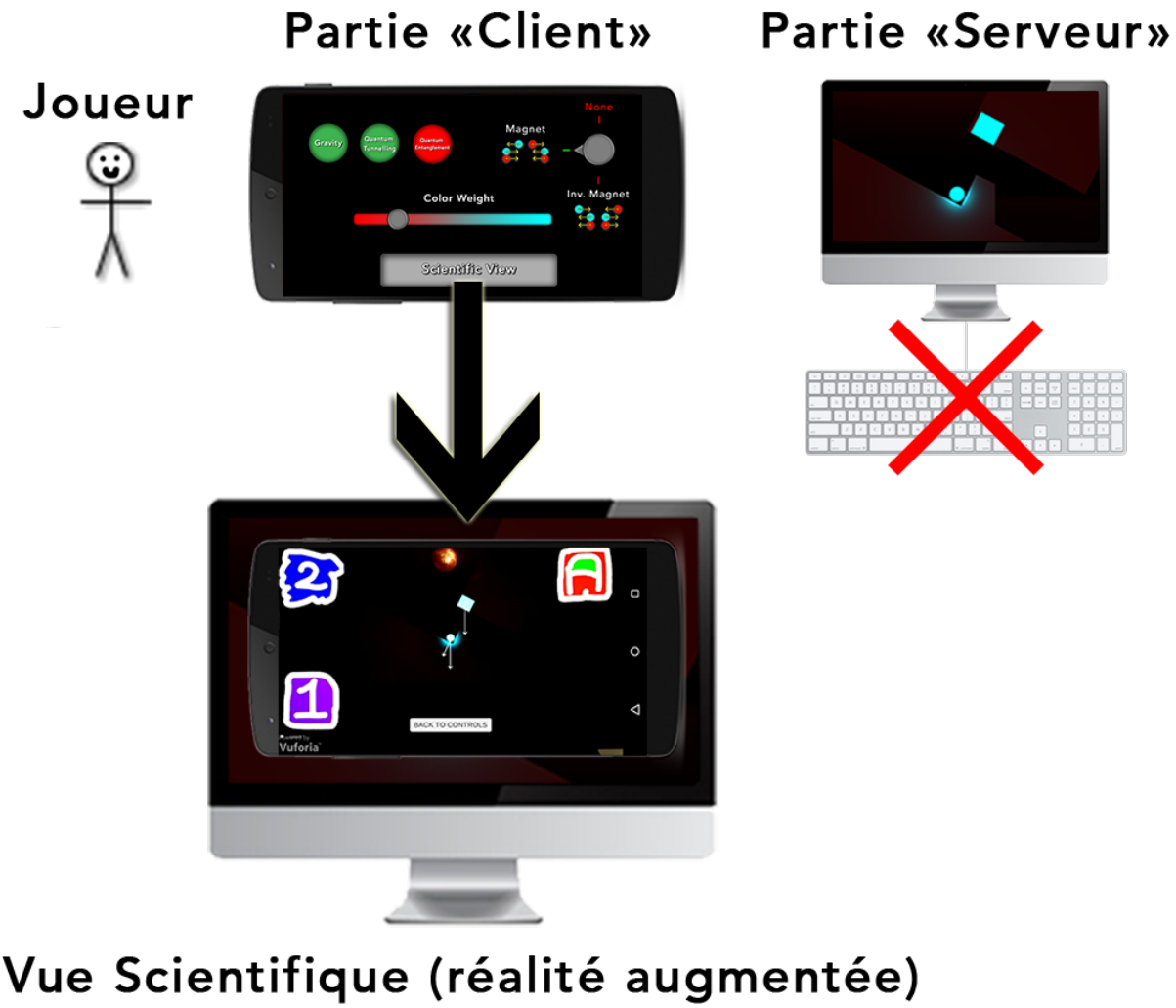
Pitch

Un couple de particules (des particules cyans et rouges) vient d'être découvert par un modèle informatique. Le jeu place le joueur dans la peau d'un scientifique (au sens propre, puisque l'on va voir que le joueur ne va pas contrôler d'avatar mais fera partie intégrante du jeu). Il va devoir tester ces particules dans un simulateur afin de vérifier la viabilité de ces dernières. Pour cela, un test a été mis en place par les scientifiques (un peu à la manière du test de Turing, qui permet de savoir si l'on a affaire à une IA ou un être humain). Si le joueur arrive à faire parvenir les particules à la fin du test, elles seront validées.

Cette expérience est principalement composée d'une suite d'énigmes et le scénario a peu d'importance : l'intérêt repose principalement sur le gameplay.

Gameplay

C'est ici que réside l'originalité de l'expérience. Tout le gameplay du jeu est indirect. Le joueur doit résoudre les énigmes afin de pouvoir faire avancer au moins une particule. Pour cela, le joueur va pouvoir changer les règles physiques qui s'appliquent grâce au deuxième écran (le smartphone) et ainsi pouvoir influencer la simulation.



Les différentes parties impliquées dans ce jeu

Partie « Serveur »

C'est l'ordinateur du joueur. Cela représente le simulateur et c'est ici que le joueur va avoir une vision « réaliste » du jeu. Il va voir les obstacles, les particules, le sol, etc… Cependant, il ne peut pas interagir avec le jeu via l'ordinateur, il se contente d'observer son modèle physique évoluer.

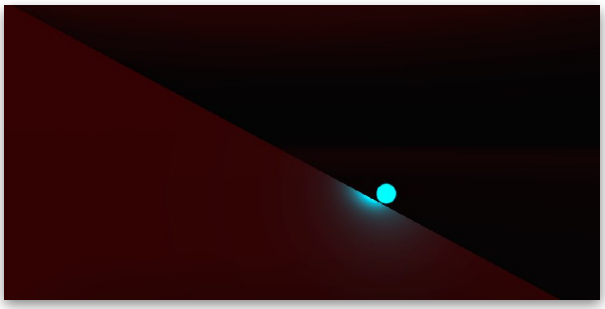
Partie « Client »

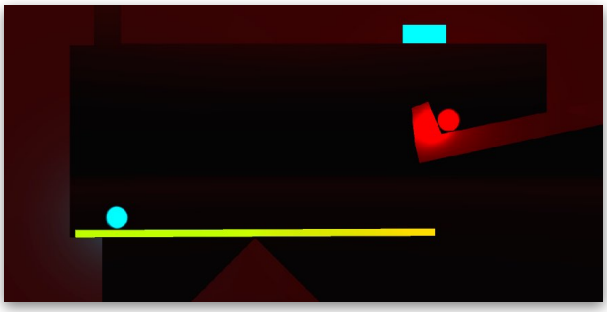
C'est le smartphone du joueur. Cela représente le panneau de contrôle de la physique du jeu. Le joueur va pouvoir changer les règles de cette dernière pour faire progresser de manière indirecte les particules du jeu et résoudre les énigmes. De plus, il pourra aussi activer une vue « scientifique » qui permettra alors de voir, en réalité augmentée, toutes les forces qui sont appliquées à chacun des objets du jeu, et ainsi pouvoir comprendre pourquoi la particule reste bloquée ici, pourquoi elle ne réagit pas comme prévu, etc. Cette vue permet aussi de voir certains obstacles qui sont invisibles autrement ainsi que les objets connectés (avec l'intrication quantique).

Navigation

Le jeu est constitué d'un seul et unique niveau dans lequel on va progresser et débloquer de nouvelles options de personnalisation du modèle physique au fur et à mesure, qui, en les combinant avec les éléments débloqués précédemment, permettront de résoudre l'énigme actuelle et donc d'avancer.

La caméra suit un objet constamment. Au début, elle va suivre la particule. Parfois, elle va prendre du recul pour donner une vue d'ensemble d'un passage (d'une énigme) puis suivre une particule différente une fois le passage traversé (à certains moments du niveau, il est possible de « changer » de particule).





Figures 1 et 2 : Illustration du suivi de la caméra. A gauche, la caméra va suivre la particule. A droite, elle est centrée sur l'énigme

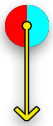
Lorsque le joueur sera bloqué sur une énigme ou qu'il se passera quelque chose d'étrange, il devra utiliser la vue scientifique pour voir ce qui cloche. Par exemple, il pourra se rendre compte qu'en réalité, l'objet reste bloqué car telle ou telle force (la gravité par exemple) annihile complètement la force qu'on essaye de faire subir à l'objet (via les forces d'attraction par exemple).

Le jeu n'a pas de Game Over. En revanche, il y a un système de checkpoint après chaque énigme. En effet, si une particule du joueur tombe dans le vide ou s'il bloque une particule de sorte à rendre impossible la résolution de l'énigme, il pourra le remettre à zéro.

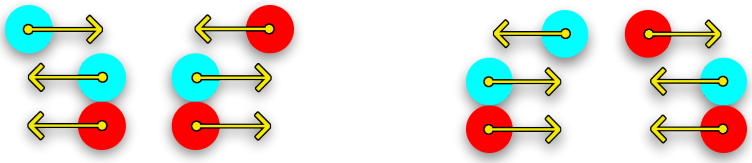
Il est intéressant de surprendre le joueur sur l'usage des mécaniques de jeu. En effet, lorsque l'on va introduire une mécanique, il est intéressant de montrer au joueur un usage « normal » de cette dernière pour ensuite l'inciter à en trouver des usages détournés. Par exemple, un obstacle invisible va bloquer la progression de la particule et on va montrer au joueur que l'effet tunnel lui permet de traverser cet obstacle. La faculté de passer à travers un obstacle invisible est acquise. Dans l'énigme d'après, le joueur va devoir utiliser ce mur invisible à son avantage en bloquant sa particule par exemple.

Le modèle physique : éléments de gameplay

Comme expliqué ci-dessus, le smartphone du joueur (la partie cliente de l'application) permet de modifier le modèle physique du jeu. Voici les options imaginées jusqu'à présent (qui se débloqueront au fur et à mesure afin de ne pas submerger le joueur et de s'assurer qu'il maîtrise chaque notion) :

1. La **gravité**. Permet d'activer ou non la gravité. 

2. L'**attraction** entre les objets colorés. Il y a deux couleurs : cyan et rouge. Trois modes sont disponibles : aucune attraction, mode magnétique (deux couleurs identiques se repoussent, deux couleurs différentes s'attirent) et le mode magnétique inversé (deux couleurs identiques s'attirent, deux couleurs différentes se repoussent).



Magnétisme / Magnétisme inversé

3. L'**intrication quantique**. Certains objets sont connectés entre eux (le joueur n'a pas la possibilité de choisir quelle particule est connectée avec quelle autre). Il peut mettre en évidence ce lien en utilisant la vue scientifique (un lien est alors tracé entre deux particules connectées). Quand l'intrication quantique est activée, chaque force exercée sur l'une des particules (sauf la gravité et les forces qui correspondent aux réactions des supports comme le sol et les murs) est aussi appliquée sur la particule connectée.

4. L'**effet tunnel**. Certains obstacles invisibles (que l'on peut tout de même voir via la vision scientifique) peuvent être traversés par les particules quand l'effet tunnel est activé. Dans certains cas, ces obstacles vont bloquer le passage et devront donc être traversés. Dans d'autres cas, ils vont servir à emprisonner une particule pour la protéger ou l'isoler afin de pouvoir résoudre l'énigme.

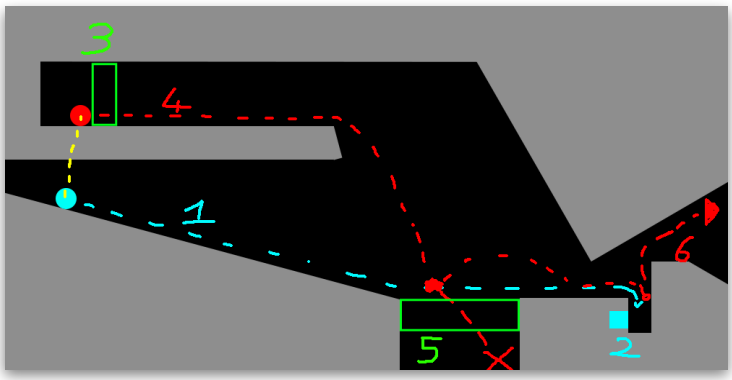
5. La **masse des couleurs**. C'est un slider qui permet de répartir les masses associées à chaque couleur. Par exemple, dans le schéma ci-dessous, les objets cyans auront une masse importante et les objets rouges une masse faible. Cela signifie que les objets cyans seront moins sensibles aux forces, ils auront aussi moins de mal à activer les bascules. De la même manière, les objets rouges seront plus sensibles aux forces et beaucoup plus légers.

Slider pour sélectionner la masse des couleurs

6. L'**indéterminisme de la mesure**. Certains objets (obstacles, murs, particules) n'auront pas de couleur. Le joueur ne saura donc pas de quelle couleur est l'objet. Il devra donc utiliser la vue scientifique et en déduire, en fonction des forces qui s'appliquent sur l'objet, sa couleur pour pouvoir l'utiliser efficacement afin d'avancer.

Autre exemples d'énigmes

Des exemples d'énigmes utilisant ces mécaniques sont visibles dans la vidéo du prototype. Voici un autre d'exemple avec les mécaniques qui ne sont pas présentes dans le prototype.



Autre exemple d'énigme mélangeant intrication quantique, effet tunnel et forces d'attraction

1. On laisse l'effet tunnel désactivé pour aller dans le trou (les obstacles verts sont des obstacles invisibles que l'on ne voit qu'avec la vue scientifique)
2. Comme les deux particules sont connectées on met la force d'attraction en mode magnétique. La force appliquée avec la particule cyan (vers la droite) est répercutée sur la particule rouge
3. La particule étant bloquée, on active l'effet tunnel pour pouvoir avancer
4. Maintenant débloquée et grâce à l'intrication quantique, la particule va avancer vers la droite
5. On désactive l'effet tunnel pour ne pas tomber
6. Une fois les deux particules dans le trou, on passe en mode magnétisme inversé pour faire remonter la particule rouge et pouvoir avancer

Partie spécialité : Programmation

Dans cette partie, je vais vous présenter le comportement et les interactions client/serveur ainsi que le fonctionnement de la vue scientifique.

Introduction et outils de développement

Tout d'abord, convenons des outils utilisés. Dans toute cette section, j'utiliserais le moteur Unity. J'emploierais donc toutes les méthodes présentes au sein du moteur (notamment pour la partie réseau). Le choix de ce moteur est lié à différents paramètres :

- la partie réseau est grandement simplifiée grâce aux classes dédiées
- le fait que l'on puisse exporter vers plusieurs plateformes ce qui permet notamment de n'avoir qu'un seul projet pour gérer la version client et la version serveur en même temps (par conséquent, il y aura deux scènes distinctes)
- c'est un moteur que je connais bien

De plus, pour l'ensemble des diagrammes ci-dessous, j'utiliserais le langage de modélisation UML.

La connexion client/serveur

Tout d'abord, voici le diagramme de machine à état du serveur et du client.

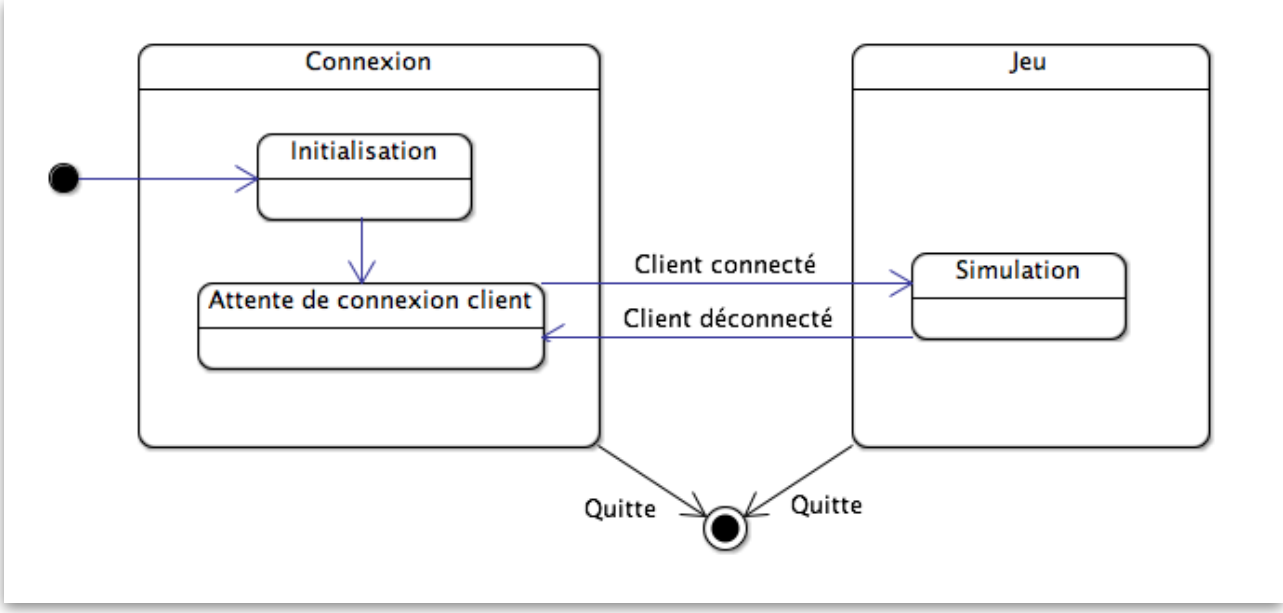


Figure 3 : Diagramme de machine à état du serveur

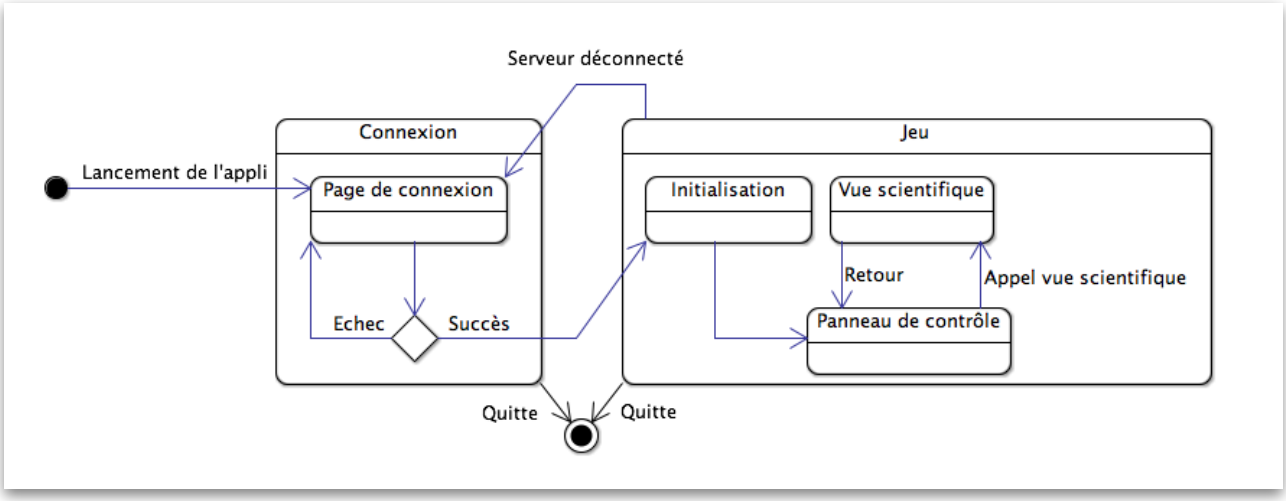


Figure 4 : Diagramme de machine à état du client

On pourra également utiliser une technique de connexion automatique appelée **UDPBroadcast**. Cette technique permet au client de trouver automatiquement l'adresse du serveur (à condition que les deux parties soient connectées sur le même réseau local). Pour cela, le serveur va envoyer des messages (contenant son adresse) sur le réseau en « broadcast » afin que tout le monde les voit. Le client va se mettre à l'écoute du réseau à la recherche de ce message. Lorsqu'il va le trouver, il aura juste à lire l'adresse qui est contenue dans le message et il pourra donc engager la conversation avec le serveur.

La partie ne va commencer qu'à partir du moment où le serveur et le client seront connectés. Une fois le client connecté, le serveur va envoyer au client sa résolution d'écran (nous verrons l'utilisation de cette information dans la partie consacrée à la vision scientifique).

Si le client se déconnecte, le jeu (côté serveur) est mis en pause le temps que le client se reconnecte. De la même manière, du côté client, comme indiqué dans le diagramme de machine à état ci-dessus, lorsque la connexion est perdue, on va retourner sur la page de connexion en attendant de retrouver le serveur.

Les interactions client/serveur

Voici le diagramme de séquence correspondant aux interactions client/serveur.

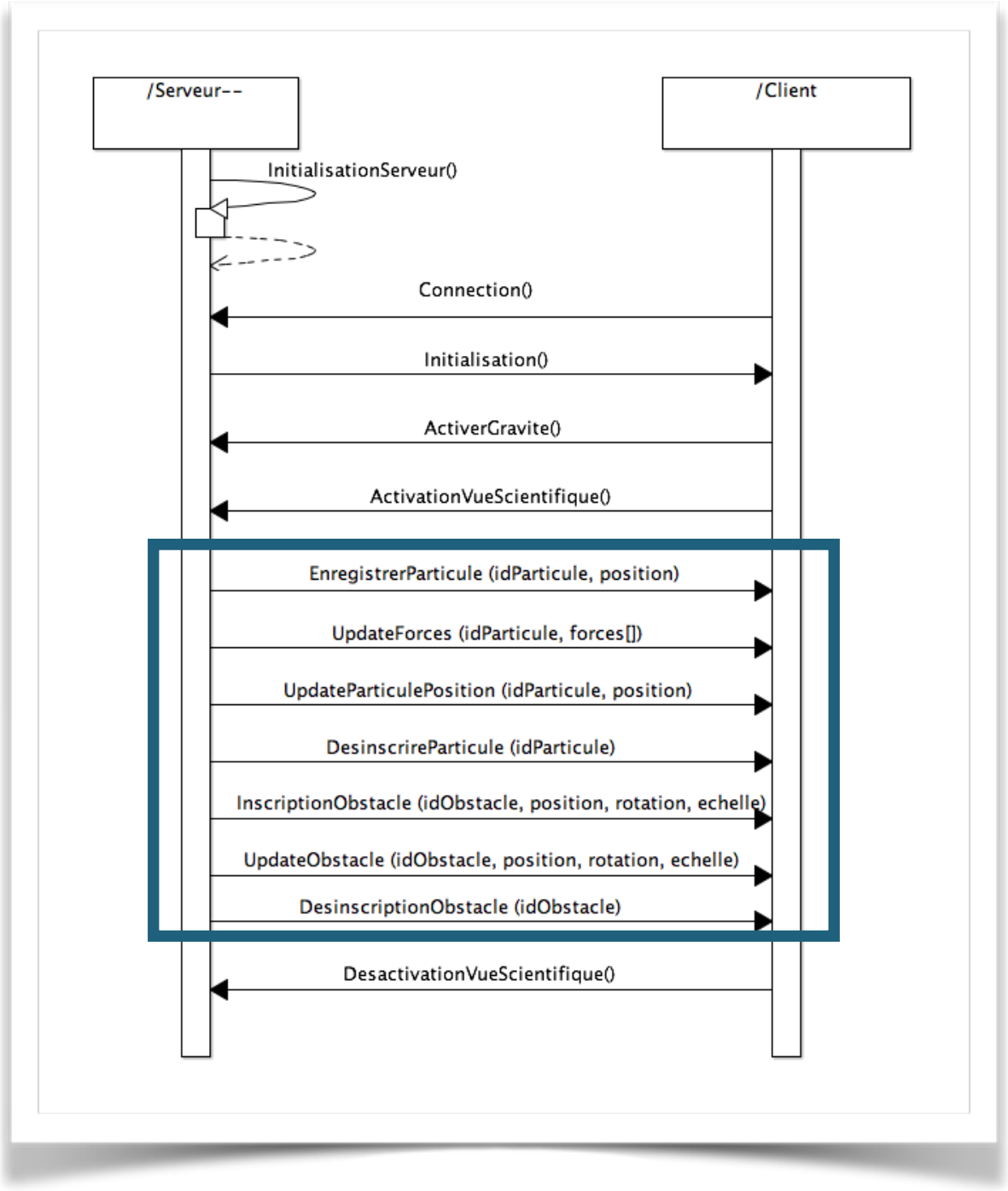


Figure 5 : Diagramme de séquence montrant les interactions client/serveur

Pour faire communiquer le serveur et le client, on utilise les mécaniques réseau implémentés dans Unity. Ce dernier possède plusieurs manières de communication réseau et notamment le système de RPC (*Remote Procedure Call*). Il s'agit simplement d'appeler une fonction sur un script distant en passant d'éventuels paramètres. Pour utiliser les RPC, il va falloir

utiliser le composant NetworkView de Unity. De plus, la fonction distante doit être marquée comme RPC. Voici un exemple pour la gravité.

Côté client :

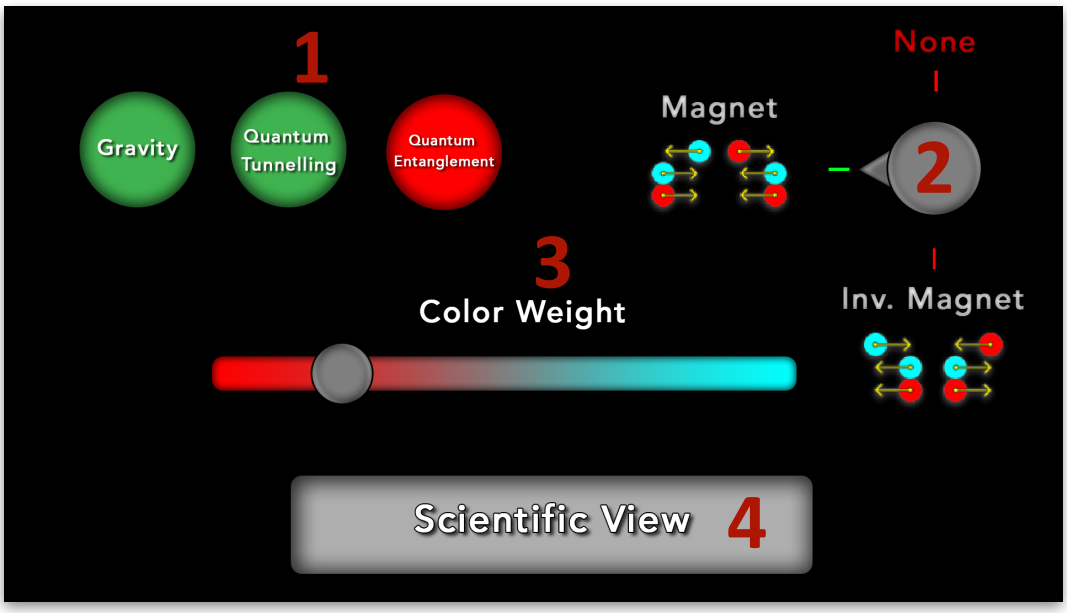
```
NetworkView nv = GetComponent<NetworkView>();

[…]

nv.RPC("ChangeGravity", RPCMode.Server, toggleGravity.isOn);
```

Côté serveur :

```
[RPC]
void ChangeGravity(bool enabled) {
      if(enabled)
            PhysicsHandler.Instance.EnableGravity();
      else
            PhysicsHandler.Instance.DisableGravity();
}
```

PhysicsHandler (qui implémente le design pattern Singleton) est donc notifié et va réellement activer ou non la gravité. Il est aussi important de noter que l'ordre dans lequel on appelle les RPC sera exactement le même que celui dans lequel les fonctions seront appelées. Par ailleurs, les RPC ne prennent pas en charge tous les types de paramètres, et il faudra donc ruser (dans le pire des cas, on utilise la sérialisation permise par le C#).

Voici l'IHM côté client et les différents prototypes des fonctions appelées.



IHM côté client

- **1** : Ces boutons n'envoient qu'un booléen (activé ou non) lors de l'appui.

```
[RPC]
void ChangeXXX(bool enabled) {
     [...]
}
```

- **2** : Ici, plusieurs états sont possibles. On définira donc une énumération (comme la partie client et serveur appartiennent au même projet, les deux verront l'énumération). On passera donc en paramètre l'index du mode choisi (0, 1 ou 2 dans ce cas).

```
[RPC]
void ChangeAttraction(int mode) {
     PhysicsHandler.Instance.SetAttractionMode(
          (PhysicsHandler.AttractionType)mode);
}
```

- **3** : Ici, on a juste besoin d'un flottant compris entre 0 et 1 (0 = rouge >> cyan, 1 = cyan >> rouge).

```
[RPC]
void ChangeColorWeight(float ratio) {
     [...]
}
```

- **4** : Pour ce dernier cas, on a même pas besoin de paramètre. En effet, ce bouton va activer la vue scientifique et donc cacher la vue. Côté serveur, il va afficher les marqueurs. Dans la vue scientifique, un bouton servira à désactiver cette dernière et appellera donc la fonction contraire.

```
[RPC]
void EnableARMode() {
     ARServerSide.Instance.EnableARMode();
}
```

La vue scientifique

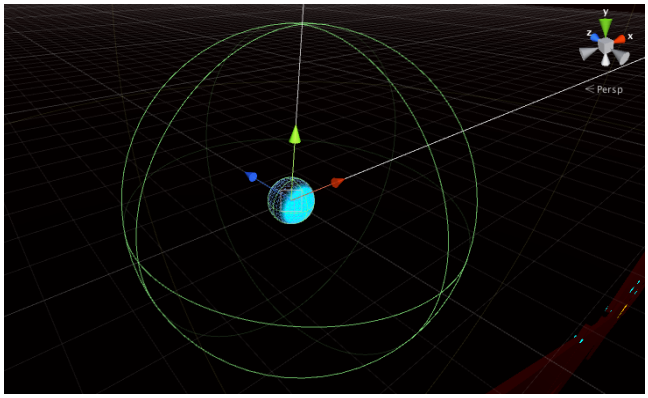
Je vais maintenant vous présenter le fonctionnement de la vue scientifique du jeu. La vue scientifique telle qu'imaginée jusqu'à présent offre trois fonctionnalités, qui s'affichent en réalité augmentée lorsque l'on regarde l'écran via le smartphone :

- Affichage des forces
- Affichage des particules connectées (intrication quantique)
- Affichage des obstacles invisibles

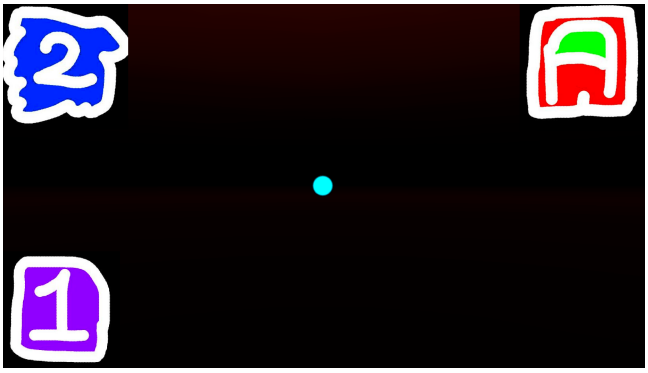
Pour la réalité augmentée, plusieurs SDK sont disponibles sur Unity et c'est Vuforia (développé par Qualcomm) qui à retenu mon attention. Lorsque le joueur va activer la vue scientifique, le serveur va en être informé afin de pouvoir placer les marqueurs qui seront ensuite reconnus, permettront de repérer l'écran et de calculer différentes choses comme l'inclinaison de l'écran. De plus, toutes les x secondes (une fréquence trop élevée surchargera le réseau mais une fréquence trop faible donnera un résultat trop peu précis) le serveur enverra au client les informations concernant la position des particules, les forces qui sont appliquées, les obstacles invisibles, etc. Ces fonctions sont représentées dans le diagramme de séquence ci-dessus (figure 5) par les fonctions encadrées en bleu. Ces fonctions ne sont appelées que pour des objets visibles à l'écran.

Lorsque cette vue sera désactivée, les marqueurs seront cachés et les informations concernant les forces et les positions des objets ne seront plus transmises.

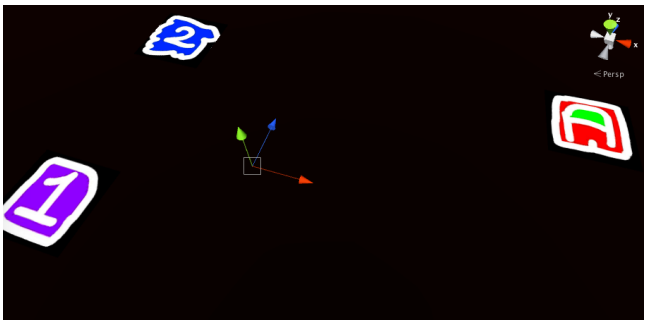
Je vais maintenant détailler comment gérer l'affichage des forces sur les particules.



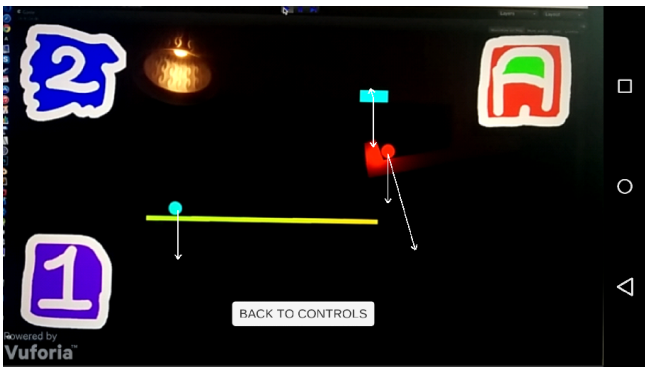
Chaque particule côté serveur à une position dans l'espace et un dictionnaire qui contient l'ensemble des différentes forces qui lui sont appliquées (gravité, attraction, etc…).



On récupère la position de la particule sur l'écran. Comme la taille de l'écran du serveur est connue par le client (elle est donnée lors de la connexion au serveur) on saura trouver la position de la particule sur l'écran vue par la caméra du client.



Vuforia replace dans l'espace (côté client) les marqueurs qu'il va trouver dans l'image de la caméra du smartphone. Il nous suffit donc de calculer la position de la particule côté client en fonction de la position des marqueurs et de la position de la particule sur l'écran.



Enfin, on convertit la position dans l'espace (côté client) en position de l'écran afin de pouvoir afficher en surimpression les forces sur les bons objets. Le serveur a donc besoin d'envoyer la valeur des forces (c'est un tableau de vecteur).

Voici le code correspondant au calcul de la position d'une particule en fonction de la valeur reçue par le serveur, de la position des marqueurs et de la taille de l'écran :

```
float xRatio = particleScreenPos.x / serverScreenSize.x;
float yRatio = particleScreenPos.y / serverScreenSize.y;

Vector3 xDiff = HD.position - HG.position;
Vector3 yDiff = HG.position - BG.position;

Vector3 pos = BG.position + xDiff * xRatio;
pos += yDiff * yRatio;
```

HD, HG et BG correspondent respectivement au composant `transform` d'Unity (qui contient la position, la rotation et l'échelle de l'objet) du marqueur en haut à droite, en haut à gauche et en bas à gauche. Enfin, `pos` est la position de la particule dans l'espace client. Il suffit donc ensuite de convertir cette position en une position sur l'écran pour pouvoir afficher les forces (la taille du corps de la flèche dépend de la norme du vecteur).

Bonus : Prototype

En parallèle à la rédaction de ce dossier, j'ai aussi voulu développer un prototype de l'idée que je viens de vous exposer. Ce prototype, développé en une dizaine d'heures, me permettait à la fois d'étendre mes connaissances (je n'avais jamais fait de réseau avec Unity), de pouvoir présenter mon idée de manière concrète et enfin, de vérifier l'intérêt du jeu. Un niveau complet a été réalisé, avec quelques éléments de gameplay. Le scénario n'est pas implémenté et le niveau présenté dans ce prototype n'est pas un niveau final, mais plus une démo de quelques énigmes avec certains éléments de gameplay du jeu (la gravité et les différentes forces d'attraction).

Une vidéo de présentation du prototype est disponible à cette adresse : www.youtube.com/watch?v=i8KHXROcr8o .