Projet Arduino 2019/2020

CUBINO

GABRIEL REYNES
THIBAULT SOULABAILLE

SOMMAIRE

INTRODUCTION

Au cours de la deuxième année du cycle préparatoire, un projet liant électronique et informatique est mis en place. L'idée est d'utiliser Arduino et les modules qu'il propose pour se lancer dans un projet en binôme, qui nécessite plusieurs semaines de préparation, et 8 séances encadrées de 3h en classe. Le but est de donner aux élèves une idée de ce à quoi peut ressembler un projet en entreprise et de les entrainer à travailler en équipe, à chercher des solutions concrètes à des problèmes qui n'ont pas forcément été déjà résolu, afin de donner un aperçu de ce à quoi peut ressembler le métier d'ingénieur.

Lorsque les objectifs de ce projet nous ont été présenté, nous avons longuement hésité à quel sujet choisir. Notre première idée était de réaliser un drone qui se stabiliserait seul, à l'aide d'un accéléromètre et gyroscope. Nous nous sommes cependant rendu compte, après avoir réalisé plusieurs tests sur les capteurs, qu'un tel projet ne nous plaisait pas vraiment, et risquait de nous lasser rapidement. Nous avons ainsi eu l'idée de réutiliser le travail réalisé avec les capteurs pour l'adapter à un nouveau projet : le Cubino.

PARTIE I : DEFINITION DU PROJET

1. Idée globale

Le but du projet est de réaliser un cube de LED, à l'intérieure duquel nous simulerions, avec un programme, des billes ou particules, leurs interactions entre elles et leurs interactions avec la paroi du cube. Le cube est ainsi couplé à un accéléromètre, qui donne la position et l'accélération du cube à chaque instant, afin de pouvoir réaliser la simulation. Nous pensions pouvoir incorporer plusieurs modes de fonctionnement (simulation de billes, de bulles, mini-jeu interactif), mais nous n'avons malheureusement pas eu le temps de les mettre en place.

2. Cahier des charges

F1 : Simulation de l'accélération	Le cube doit être capable, en utilisant les valeurs renvoyées par un accéléromètre, de représenter des objets sur lesquelles s'applique la gravité et l'inertie. On doit ainsi pouvoir représenter les chocs et interaction entre les objets et le cube, et entre les objets eux-mêmes.
F2 : Création d'une application mobile	L'utilisateur doit pouvoir être capable de sélectionner les différents modes proposés à l'aide d'une application mobile. Cette application mobile est reliée au cube à l'aide d'une connexion Bluetooth. A l'issu du projet, nous n'avons pas eu le temps de réaliser cette application, et le cube finalement réalisé ne comporte donc pas de module Bluetooth.
F3 : adaptation à l'utilisation	Le cube doit être assez petit pour être manipulable par l'utilisateur, tout en contenant assez de LED pour que les effets soient perceptibles. Nous avons choisi les dimensions 7x7x7 LED. Au bout du compte, nous avons réussi à avoir un cube de 7x7x4 (seulement 4 plaques)

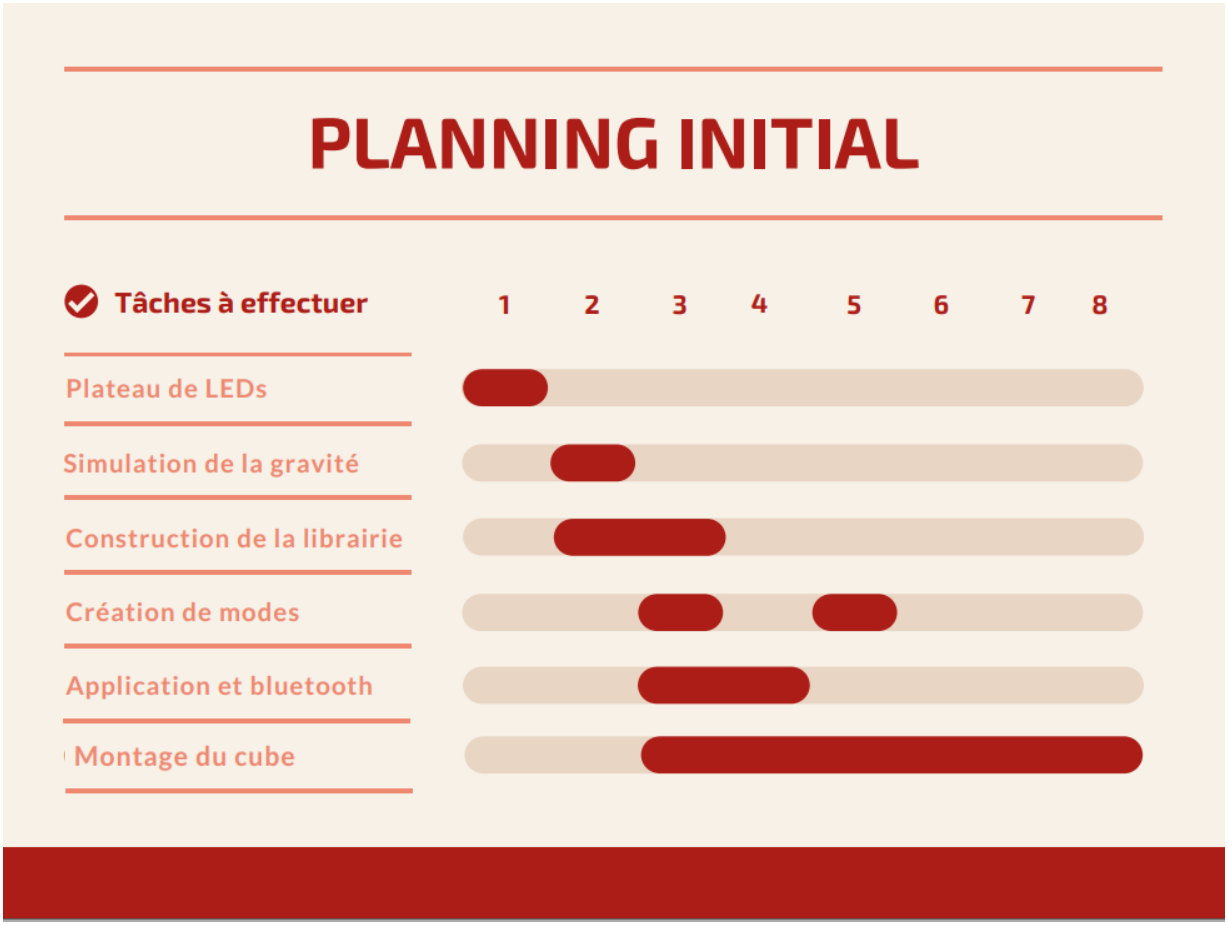
Fonctions principales du projet

Matériel nécessaire :

- 7x7x7 = 343 LED RGB
- Des tiges de métal rigides pour l'armature
- Un accéléromètre
- Des câbles et résistances
- Une alimentation 5V 20A

3. Déroulement du projet

Le planning que nous avions initialement prévu s'est avéré totalement différent du planning final. Nous avions ainsi prévu de terminer la construction d'une plaque du cube en une séance, et de commencer à faire des tests dessus dès la séance d'après. Nous pensions ensuite pouvoir finir le code rapidement et consacrer les dernières séances à la construction du cube. Cependant, nous avaons rapidement compris que la construction du cube prendrait énormément de temps.



Pour réaliser le cube, nous avons en grande partie réutilisé des composants d'un projet similaire réalisé l'année dernière. En effet, un de projets était un cube de 8x8x8 LED, et nous nous somme donc appuyé sur le travail fait pour ce cube là pour construire le notre. Nous avons réutilisé le « patron » : une plaque de bois percé aux endroits prévus pour les LED, qui nous a permis de créer nos différentes plaques, le métal de l'armature et la plupart des LED. Cependant, les dimensions n'étant pas les mêmes (8x8x8 contre 7x7x7 pour notre cube), nous avons du démonter entièrement leur cube, pour le réadapter à nos dimensions. De plus, lors de la première séance, nous nous sommes apperçu que certaine des LED avaient été grillées et ne fonctionnnaient donc plus. Il a donc fallu, au cours de la construction, tester chaque LED individuellement avent de la souder à la plaque.

Nous nous sommes aussi rendu compte que la construction du cube prendrait beaucoup plus de temps que ce que nous avions prévu, car il faut construire l'armature puis souder les 4 pattes de chaque LED, sans oublier de les tester les unes après les autres pour éviter les problèmes techniques. Nous avons donc construit le cube en parallèle à la réalisation du code Arduino et des librairies. M. Masson nous a prété un fer à souder, ce qui nous a permit d'avancer la construction en dehors des séances.

Lors de la séance 4, nous avons fait face à un nouveau défi : le weekend du 8 février s'est tenu la journée portes-ouvertes de Polytech, et M. Masson et M. Abderrahmane nous ont donc proposé d'y participer, ce que nous avons accepté. Nous nous sommes dès lors concentré sur une seule plaque, pour avoir quelque chose de fonctionnel. La journée portes-ouvertes a été une expérience très enrichissante. Nous avons pu présenter une partie de notre projet (une plaque sur laquelle était simulée des chuttes et chocs de billes) à de nombreuses personnes, et discuter de Polytech et de notre expérience personnelle.

Nous avons décidé de monter le cube une fois toutes les plaques terminées. Au final, nous avons réussi à réaliser un cube de 7x7x4 (4 plaques). Lors des derniers jours de tests, nous avons dû faire face à de nombreux problèmes, et l'un d'eux était que l'accéléromètre s'est mis à renvoyer des valeurs erronées. Nous n'avons donc pas pu avoir le résultat désiré lors de l'oral.

PARTIE II : PARTIE TECHNIQUE

1. Cube de LED

• LED RGB

Les LED utilisées pour le projet sont des LED RGB (Red Green Blue) à quatre broches : une broche pour le 5V, une pour le ground, une pour la réception d'informations et une autre pour l'envoie d'informations.

Toutes les LED du cube sont branchées en série. Pour allumer une LED en particulier, on envoie des informations à la première LED. Lorsqu'on a envoyé trop d'informations, sa mémoire devient saturée, et elle envoie alors des informations à la deuxième, ainsi de suite. Ce procédé permet ainsi, en donnant un indice à chaque LED, de choisir laquelle allumer, de quelle couleur, et avec quelle intensité.

• Plaque de LED

Comme expliqué précédemment, le cube est composé de plaques empilées les unes sur les autres. Ces plaques elles-mêmes sont faites d'une armature faite de tige de fer rigides, et de 49 LED branchées en série.

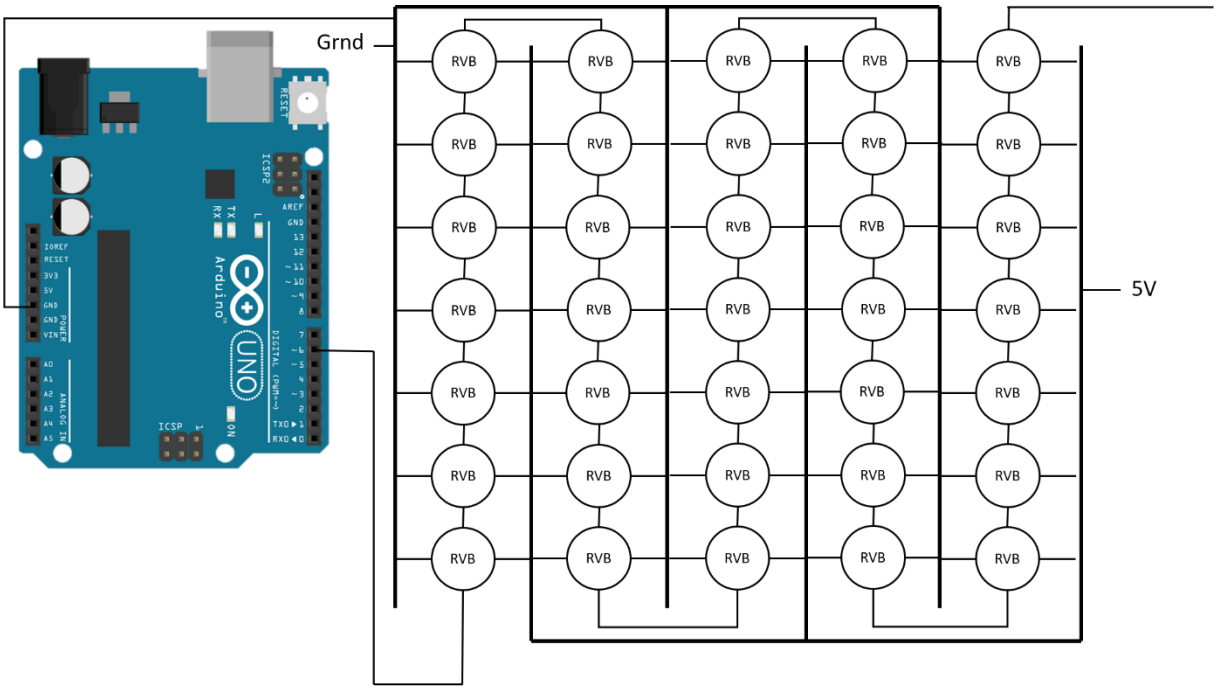
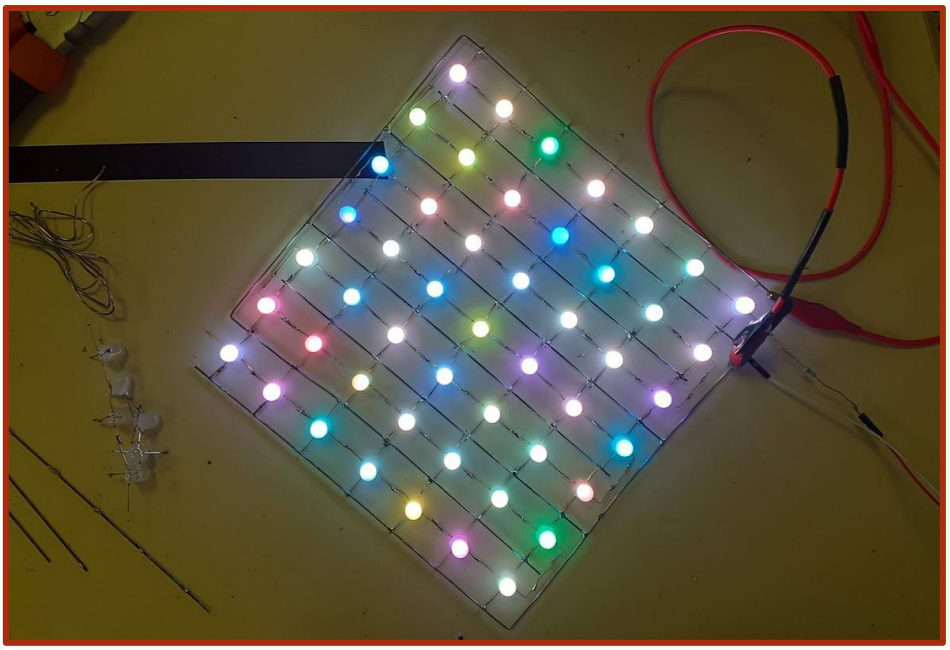
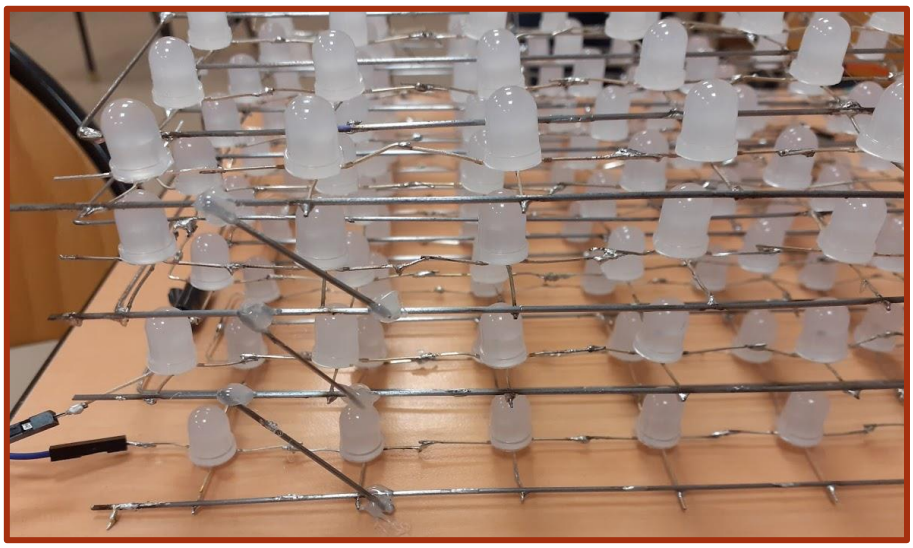


Schéma d'une plaque de LED

Comme on peut le voir sur le schéma, un des « râteaux » de l'armature est relié au 5V, l'autre est relié au ground. Cet agencement permet à chaque LED d'être alimentée. La dernière LED de chaque plaque est branchée directement à la première LED de la plaque suivante, afin que toutes les LED du cube soient reliées entre elles en série.



Une fois toutes ces plaques finies, il faut les empiler afin de monter le cube. Pour cela, on utilise les mêmes tiges de fer rigides que pour l'armature, collées avec un pistolet à colle pour éviter les courts-circuits. On relie ensuite les 5V et les ground de chaque plaque avec ceux de la plaque suivante, pour alimenter tout le cube.



2. Capteurs

Il nous fallait, pour pouvoir simuler les mouvements des billes dans le cube, une manière de savoir comment le cube était déplacé / tourné. Nous nous étions, dans le cadre de la première idée de projet (celle du drone stabilisé), déjà renseignés sur le principe en électronique de la fusion de capteurs. Le premier de nos essais en la matière consistait en la « fusion » d'un accéléromètre et d'un gyroscope. Le gyroscope renvoyait des données en radian par seconde, qui, une fois intégrées nous permettait de connaitre la rotation approximative du cube. Ces données devaient être ensuite « corrigées » puisque trop peu fiables lorsque le cube subissait des rotations plus rapides. Nous corrigions donc ces données en créant une nouvelle variable ainsi composée sur chaque axe :

RotationX = DonnéeXGyro * 0.9 + DonnéeXAcc * 0.1

Ceci permettait d'avoir des données fiables qui corrigeait les défauts de chacun des deux capteurs (le manque de précision du gyroscope et la trop grande sensibilité de l'accéléromètre).

Cependant, cette fusion de capteur ne nous permettait d'avoir que des informations sur la rotation du cube. Et nous souhaitions que celui-ci soit complètement manipulable. Il nous fallait donc avoir des informations sur son déplacement. Nous avons longtemps cherché un moyen de connaitre avec assez de fiabilité les déplacements du cube en plus de sa rotation, une simple intégration des données de l'accéléromètre n'étant pas suffisante puisque l'accélération du cube due à un mouvement se confondait avec celle due à la gravité. Une solution ayant été trouvée fût celle de l'algorithme de Madgwick, un algorithme puissant utilisé pour la fusion d'un gyroscope, un accéléromètre, et d'un magnétomètre. Cet algorithme permettait de connaitre avec une bonne précision la position et la rotation de notre cube à tout instant dans un espace donné.

Malheureusement, il n'a pas été possible pour nous de mettre en place cet algorithme dans notre programme par manque de temps mais aussi d'espace et de puissance sur la carte que nous utilisions. Cependant, nous nous sommes familiarisés avec l'algorithme et son fonctionnement (notamment par l'utilisation des quaternions). Et avec un peu plus de temps pour finir notre projet, l'implémentation de cet algorithme aurait été la prochaine et dernière étape du projet pour atteindre le résultat imaginé dans un premier temps.

Il a donc été décidé de ne se servir que de l'accéléromètre et du gyroscope, le résultat obtenu avec ceux-ci étant tout de même largement satisfaisant.

PARTIE II : PARTIE INFORMATIQUE

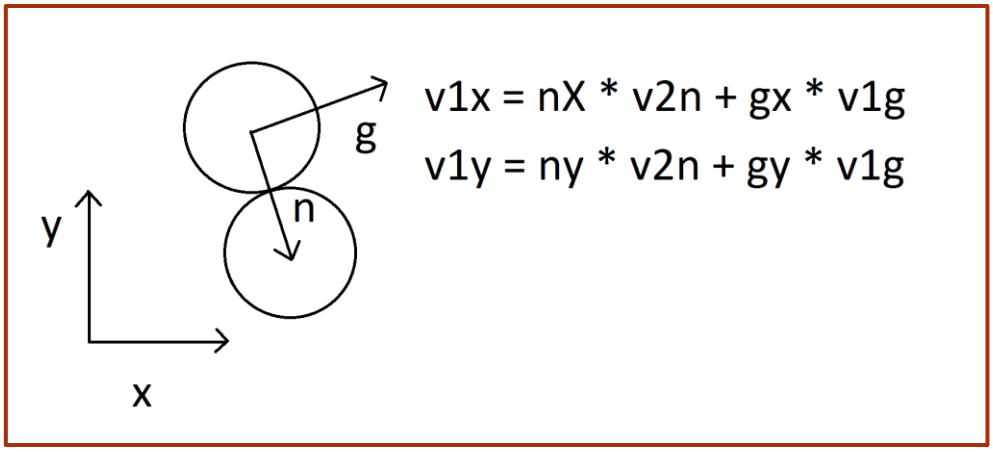
1. Simulation des interactions de billes

Un des premiers challenges du projet fût de mettre en place notre propre système de simulation d'interaction entre des billes. Il a fallu dans un premier temps nous renseigner sur les formules de physique applicables aux chocs. Et, nous avons, heureusement, assisté à des cours de mécanique l'année dernière dans lesquels cet exact sujet était traité. Cependant, deux problèmes se posait face à l'utilisation des formules vues en cours. En effet, en plus d'être très lourdes au niveau des calculs nécessaires à leurs résolution, ces formules ne s'appliquaient qu'à des collisions aux conditions très précises. Or, il nous fallait un programme capable d'exécuter rapidement le calcul de collisions de plusieurs billes (jusqu'à dix minimum) en temps réel, à chaque « loop » de notre programme, et ce, sur une carte ayant une capacité de calcul et un espace de stockage limité.

Il semblait assez évident que ce programme devait être orienté objet, puisqu'il s'agissait ici de simuler le déplacement d'objet indépendants aux caractéristiques similaires. Il a donc été choisi de commencer les premiers essais de simulation sur Processing en langage JAVA. Le logiciel Processing nous permettant de visualiser le résultat des données du programmes avec un outil graphique relativement simple.

Après de nombreux tests peu concluants et plusieurs réécritures du programme jusqu'à sa racine. Nous avons trouvé une solution qui semblait correspondre parfaitement aux attentes. Nous avons en effet utilisé des projections de vecteurs de vitesses de chaque bille pour simuler l'échange de force entre deux billes en collision et les calculs, bien que relativement rapides à effectuer pour la machine, renvoyait des résultats plutôt convaincants.

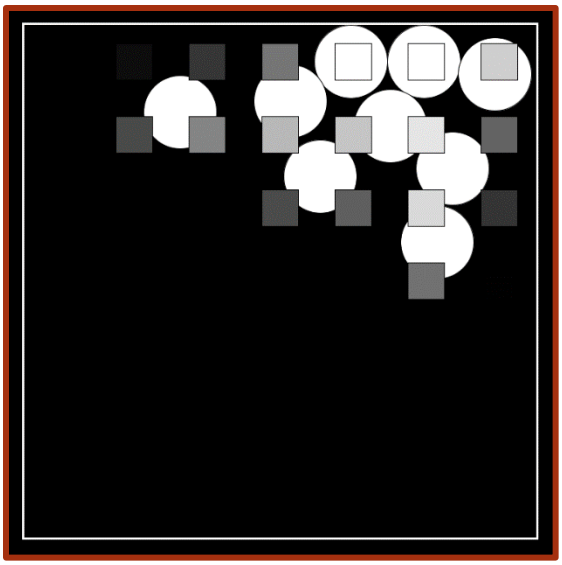
Détails des calculs des échanges de vitesse entre les billes :

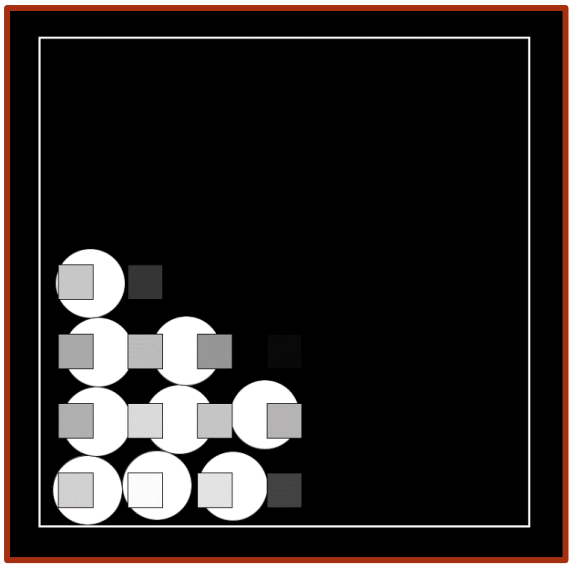


Evidemment, trouver cette solution pour la simulation des échanges de force n'a pas suffi. Puisque de nombreux problèmes subsistaient. En effet, il a fallu différencier la collision de deux billes dont les directions étaient opposées de celle la situation de billes en contact mais dont il était inutile de calculer des échanges de force (billes collées allant dans la même direction ou immobiles), mais aussi limiter le nombre de calcul de collisions par « loop » dans le cas où le nombre de billes était trop grand.

Une fois cette solution trouvée, il a fallu terminer le programme afin de le rendre malléable. C'est-à-dire, rendre la taille et le nombre de billes complètement modifiables au gré de l'utilisateur sans qu'aucun problème ne soit rencontrés.

Images de la simulation de billes sur Processing :





2. Utilisation au sein de l'IDE Arduino

Une fois la simulation satisfaisante sur le logiciel Processing, il a fallu rendre le programme utilisable sur l'IDE Arduino afin de l'importer dans la carte UNO. Afin de garder l'orientation objet du programme, nous nous sommes vus obligés d'avoir recours à la création d'une librairie, qui permet, sur l'IDE Arduino, d'utiliser des programmes écrit en langage C++ orienté objet.

Le programme ayant été initialement codé en JAVA, il a donc fallu réécrire tout le code en C++, un langage avec lequel aucun de nous n'était familier. Il a donc fallu plusieurs heures de réécriture et d'apprentissage avant d'avoir une librairie fonctionnelle et utilisable de notre programme au sein de l'IDE Arduino.

Le passage de JAVA à C++ a aussi permis le passage au fonctionnement du programme en trois dimensions. En effet, les simulations sur Processing n'avaient été codées que sur deux axes (une simulation en trois dimensions étant bien plus dure à visualiser sur un outil graphique aussi simpliste). Il a donc fallu transformer, une fois de plus, le code de la librairie, afin de rendre tout le programme compatible avec un cube en trois dimensions.

La version finale de notre bibliothèque se résumait en une classe appelée « LedArray » dont le constructeur prenait en paramètre le nombre et la taille des billes voulues et dont la fonction « update » récupérait les informations de l'accéléromètre sur les trois axes X, Y, et Z.

Cependant, les ajouts des calculs nécessaires à la transmission des forces entre deux billes sur trois axes ont eu raison de la carte UNO sur laquelle nous travaillions depuis le début du projet (l'utilisation de la mémoire dynamique atteignant les 256%). Il a donc été nécessaire de changer de carte.

Nous avons dans un premier temps, opté pour une carte MEGA, qui possédait assez de mémoire pour contenir le programme fonctionnant sur trois axes. Cependant le processeur et la mémoire RAM de la carte se sont montrés trop peu puissant pour rendre un résultat convaincant. C'est donc une carte Teensy 4.0 qui fût notre dernier choix. La carte possédant à la fois une mémoire largement nécessaire au stockage de notre programme et possédant des caractéristiques bien supérieures à celles des cartes Arduino, plus limitées.

Cependant, bien que la carte Teensy puisse paraitre parfaite pour notre utilisation, elle s'est avérée être un des problèmes majeurs de notre projet. En effet, la carte Teensy est conçue pour émettre des signaux d'un voltage de 3.3V, et les LEDs que nous utilisions ne fonctionnaient qu'en recevant des signaux de 5V.

Il a donc été nécessaire d'utiliser une deuxième carte, recevant les informations sortantes du programme exécuté sur la carte Teensy via les pins Serial, qui envoyait ensuite les informations aux LEDs.

CONCLUSION

Le projet Cubino était un projet ambitieux de par le fait qu'il nécessitait un travail non négligeable à la fois dans la partie électronique, l'assemblage du cube étant une tâche fastidieuse, mais aussi dans la fusion des capteurs, et la programmation.

Le fait de devoir pousser notre apprentissage dans ces différents domaines a été un réel challenge lors de la conception de ce cube. Et même si le résultat final n'a pas été à la hauteur de nos attentes, nous sommes satisfaits de ce que nous avons produit et ressortons de cette expérience avec bien plus de connaissance que ce que nous avions au départ.

Plusieurs points auraient cependant pu être améliorés. Dans un premier temps, la structure du cube en elle-même, qui n'a jamais été réellement terminée, puis, les capteurs qui, lors de la version finale, ne permettait que de connaitre la rotation du cube et non pas sa position et son accélération propre, en dehors de celle de la gravité.

Une fois ces éléments terminés, il aurait aussi été nécessaire d'ajouter différents « modes », comme il était convenu au départ, afin de prolonger les possibilités d'éclairage et de jeu.

Ce projet a à la fois été une réelle expérience de travail en groupe, la bonne répartition des tâches étant primordiales pour la bonne avancée du projet, mais aussi un réel apprentissage, le projet nécessitant des connaissances que nous n'avions pas encore à ses débuts, à la fois en électronique et en programmation.

Bibliographie

Nous avons utilisé beaucoup de ressources sur des supports différents afin de réaliser notre projet, voici les sources principales :

- http://users.polytech.unice.fr/~pmasson/index.html (cours d'électronique)
- https://x-io.co.uk/res/doc/madgwick_internal_report.pdf (présentation de l'algorithme de Madgwick)
- https://www.mathworks.com/help/fusion/examples/Estimating-Orientation-Using-Inertial-Sensor-Fusion-and-MPU-9250.html?s_eid=PSM_15028 (démonstration de l'utilisation de fusion de capteurs sur Arduino)