Alves Claude-André, Catel Torres Arzur Gabriel, Thomas Benjamin

SYM - Laboratoire n°4

Question 1.2

- Une fois la manipulation effectuée, vous constaterez que les animations de la flèche ne sont pas fluides, il va y avoir un tremblement plus ou moins important même si le téléphone ne bouge pas. Veuillez expliquer quelle est la cause la plus probable de ce tremblement et donner une manière (sans forcément l'implémenter) d'y remédier. Le capteur est très sensible, c'est une partie de matériel qui se doit de l'être. Nous avons l'impression de ne pas bouger mais il est certains qu'un léger tremblement de la main soit à l'origine de ce tremblement. On peut aussi ajouter le fait que le capteur n'a pas besoin d'être extrêmement précis (ce n'est pas un outil utilisé dans un cadre scientifique mais plutôt applicatif et informatif), ainsi, il est possible que le matériel lui-même change constamment la valeur même si aucun mouvement n'est fait. Le résultat est que l'on voit clairement sur l'écran que la flèche tremble.

Nous pourrions résoudre ce problème avec plusieurs solutions différentes que nous avons discutés entre nous dans un premier temps :

1. Il serait possible de diminuer le nombre de mesures prises par secondes pour avoir un capteur qui certes ne serait pas à 60 fps mais qui serait moins désagréable à la vue et tout aussi utilisable dans le cadre d'un smartphone.
2. Il est aussi possible de ne mettre à jour la position de la flèche que si la valeur est suffisamment différente. Le bon côté est que les micro tremblements disparaîtront sans problèmes mais ce n'est pas forcément la solution la plus optimale.
3. Il est aussi possible de lisser les mouvements de la boussole. C'est donc du côté de l'affichage de la boussole que l'on va agir cette fois. Il serait possible que le mouvement de la flèche démarre doucement et s'arrête doucement. Cela ne permettra pas de voir toutes les valeurs de transition mais l'affichage serait plus « doux ».
4. Enfin rendre les possibilités de position de la flèche plus restreintes. Si on découpe le cercle d'affichage différemment, lorsque les valeurs sont très proches, la valeur sera tronquée / arrondie et elle restera sur la même position. C'est probablement la solution la plus facile à mettre en place et la plus cohérente pour notre application.

Nous avons concrétisé nos recherches sur internet en trouvant sur différents sites :
https://developer.android.com/guide/topics/sensors/sensors_overview
https://github.com/Bhide/Low-Pass-Filter-To-Android-Sensors
Ce site montre comment gérer ce problème avec un système de « low pass filter » (filtre passe-bas), que l'on utilise aussi beaucoup dans la compression audio et vidéo pour garder un maximum d'information « visible » pour l'utilisateur en enlevant un maximum de données inutiles car imperceptible.
Par exemple pour le son, on sait que l'oreille humaine n'entend les fréquences qu'entre 20Hz et 20KHz environ (pour les jeunes). On va donc commencer par enlever toutes les autres fréquences que le capteur va, lui, prendre en compte. Le système est un peu différent ici mais le principe reste le même, le but est de lisser un maximum les données pour garder le nécessaire. On peut dire que cette solution correspond plus ou moins au point 4 de notre réflexion.

Question 2.2

- La caractéristique permettant de lire la température retourne la valeur en degrés Celsius, multipliée par 10, sous la forme d'un entier non-signé de 16 bits. Quel est l'intérêt de procéder de la sorte ? Pourquoi ne pas échanger un nombre à virgule flottante de type float par exemple ?

Le bluetooth est un système lent, dans notre cas ce n'est pas très important d'être rapide donc on aurait très bien pu passer un float sur 32 bits. Mais d'un point de vue performances, si on voulait un système efficace pour récupérer la température, envoyer 16 bits au lieu de 32 est toujours bénéfique, que ce soit pour la mémoire ou la vitesse.
Cependant une autre raison peut être que certains systèmes ne traitent pas les virgules flottantes. Pour être le plus générique possible, on veut passer cette valeur le plus génériquement possible pour que tout système puisse utiliser le système de température.

Enfin, les calculs en virgule flottante sont plus couteux et dans le cas de la température ce n'est vraiment pas nécessaire de s'entêter à passer un float alors que les valeurs souhaitées seront pertinentes au plus avec une précision de deux chiffres après la virgule (ce qui est déjà une précision que très peu de personnes utiliseront).

- Le niveau de charge de la pile est à présent indiqué uniquement sur l'écran du périphérique, mais nous souhaiterions que celui-ci puisse informer le smartphone sur son niveau de charge restante. Veuillez spécifier la(les) caractéristique(s) qui composerai(en)t un tel service, mis à disposition par le périphérique et permettant de communiquer le niveau de batterie restant via Bluetooth Low Energy. Pour chaque caractéristique, vous indiquerez les opérations supportées (lecture, écriture, notification, indication, etc.) ainsi que les données échangées et leur format.

Le niveau de charge de la pile peut être communiqué si le périphérique possède le service battery_service au code 0x180F. Il faut alors que le service possède la caractérisique battery_level au code 0x2A19 et, si c'est le cas, le niveau de charge (données échangées) pourrait être lu via une opération de lecture sous un format hexadécimal.

Une opération de notification serait également intéressante pour prévenir l'utilisateur lorsque la batterie atteint un certain niveau (batterie pleine, batterie à moins de 15%, etc…).

Les opérations d'écritures ne sont pas supportées pour une telle caractéristique.