Coques de smartphones interactives imprimés en 3D

DESJOBERT Nathanaël - SAÜL Nicolas

Février 2015

Résumé : Ce projet à pour objectif d'enrichir l'interaction avec les portables avec plusieurs boutons NFC pour pouvoir jouer à un jeu comme le Tetris sans batterie et sans câblage entre le portable et la manette de jeu.

Mots clé : NFC ; manette ; gamepad ; coque ; Impression 3D ; Prototypage Rapide ; Android

1 Introduction

En 2014, pour la première fois en France, il s'est vendu plus de smartphones que de téléphone dit classiques. En 2013, 23,6 millions de mobiles ont été vendus auprès des français, dont 15,8 millions de smartphones. On explique ce chiffre par le changement d'utilisation du mobile en particulier. On observe ainsi que les utilisateurs passent deux fois plus de temps sur internet, que de temps passé à téléphoner. Suite à ce changement de comportement, et accompagné par l'explosion des applications mobiles, nos smartphones se transforment peu à peu en couteaux suisse, doté de plus en plus de technologies. En plus d'Internet, les mobiles peuvent télécharger des applications et des jeux afin d'agrandir l'expérience utilisateur. Ainsi on compte 500 millions de joueur sur mobile en 2013.

Dans ce contexte, Notre objectif est d'employer la technique de prototypage rapide afin de créer de objets rapidement, comme des objets connectés par exemple. Comme objet, nous avons dû réaliser une coque imprimée en 3D manette de jeu, qui sans batterie et sans fil, dialogue avec le téléphone. Notre coque sera utilisée par des utilisateurs grand public, est réalisé avec une imprimante 3D et la partie communication avec des tags NFC et un téléphone compatible NFC. Pour la démonstration de notre manette nous avons réalisé un Tetris pour Android. Notre projet est encadré par Mr Aurélien Tabard enseignant-chercheur au LIRIS (Laboratoire d'InfoRmatique en Image et Systèmes d'information), qui nous a guidé durant toute la durée de notre projet.

2 Etat de l'art

2.1 NFC : Near Field Communication

La communication en champ proche est un type de communication sans fil de courte distance (allant jusqu'à 10 centimètres). Il existe plusieurs procédures de communication :

- Mode pair à pair : Ce mode permet l'échange d'information entre deux terminaux actifs. Ainsi on peut, sans protocoles complexes comme le Bluetooth, échanger facilement et rapidement des fichiers entre deux mobiles.
- Mode lecteur : Ce mode permet un échange d'information unidirectionnel. En effet celui-ci est permis grâce à des composant asymétriques. On a d'une part un terminal mobile équipé d'un lecteur NFC et d'autre part un ≪ Tag ≫NFC, composé d'une bobine et d'une puce. Lors d'une lecture par le mobile, le mobile charge la bobine par induction et la bobine alimente la puce. La puce renvoie alors des informations, comme un numéro de série unique ou des informations préalablement chargées par un mobile. Les informations pouvant être chargés sont multiples, allant du simple texte à un lien vers une application.

Le mode lecteur permet la réalisation de circuits simples, sans batterie. En effet il est aisé de réaliser un tag NFC simple, il suffit d'une puce NFC et d'un peu de fil. En complexifiant un petit peu ce simple circuit, on peut rajouter un bouton (un inverseur), un levier qui permet de basculer entre deux puces alimentés pour chaque états du bouton. Ou alors on peux utiliser un bouton poussoir, qui, quand il serait fermé, permet de lire la puce par un lecteur. Et si le bouton serait ouvert alors le circuit est ouvert et le tag serait illisible car non alimenté. Les possibilités sont multiples tout en restant simple [3].

Différents matériaux sont utilisés pour fabriquer la bobine. Car non seulement il est possible de la faire en fil de cuivre classique, mais il est aussi possible de la réaliser en ruban de cuivre. Il est même possible de l'imprimer grâce à une imprimante utilisant une encre conductrice [2]. La forme spiraloïde de la bobine n'est même pas obligatoire. On peut réaliser une forme originale, pourvu qu'elle ai une inductance élevée et qu'elle puisse permettre l'induction [4]. Un bon choix d'épaisseur et d'un bon nombre de tour de bobine devra être opéré.

2.2 imprimante 3D

Les imprimantes 3D sont de plus en plus populaires. Elles permettent de reproduire des objets en un temps qui dépend de la complexité de l'objet ou de sa taille. Les imprimantes peuvent être de différentes qualités et de taille, par exemple une maison a été imprimée récemment en Chine [1]. Il existe donc plusieurs types d'imprimante 3D [2] :

- Le frittage laser (SLS) qui fonctionne avec un laser et de la poudre plastique, de céramique, de verre ou de métal. On retrouve donc le laser et un bac qui contient le matériau voulu (la poudre). Un rouleau vient déposer une fine couche sur la plateforme d'impression puis le laser solidifie. On répète cette opération à chaque couche jusqu'à que l'objet soit fini puis on retire le surplus de poudre non utilisé.
- La stéréo-lithographie (SLA) qui fonctionne avec un laser ultra violet et dans un bac de photopolymère liquide. Ce système imprime couche par couche, le laser frappe le liquide qui se solidifie sous l'effet des ultra-violets.
- Le procédé PolyJet ce procédé s'appuie sur la photopolymérisation comme pour le SLA. Des jets de matériau sont projetés sur le support d'impression, ce jet est définie par le logiciel d'impression de l'imprimante.
- Le procédé DLP, ce procédé sert efficacement pour les travaux de grandes précisions comme la fabrication de prothèses, bijoux... . Le procédé est à base de solidification d'un polymère liquide, grâce à une lumière UV qui vient frapper le polymère qui se trouve dans un bac pour le solidifier.
- Celle par dépôt de filament fondu (Figure 1), on met une bobine de filament (ABS , PLA , Nylon...). Son principe est simple le filament passe à travers une buse d'extrusion chauffée entre 170 et 260°C. il fond et se dépose sur un support par couche. On peut faire varier les réglages en fonctions du filament.

1. http://www.lemondeinformatique.fr/actualites/lire-le-chinois-winsun-imprime-des-maisons-en-3d-59961.html
2. http://www.lesnumeriques.com/imprimante-3d/impression-3d-differents-procedes-a1876.html

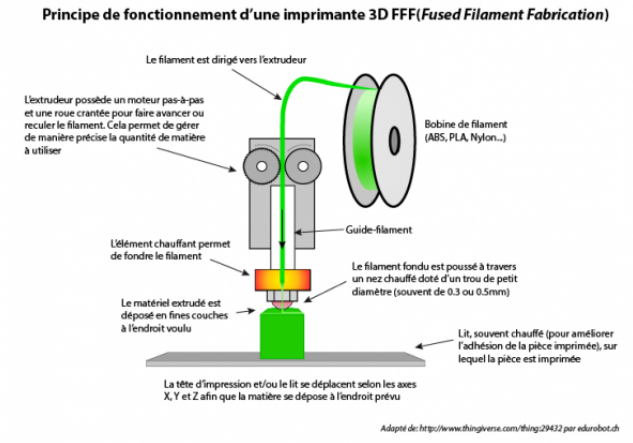

FIGURE 1 – Fonctionnement l'imprimante à filament fondu

Nous travaillons sur cette imprimante : une Makerbot Replicator 2. Elle est de gamme moyenne et peux imprimer des objets de 15 cm de diamètre environ. Nous avons accès à deux types de filament un solide et un flexible.

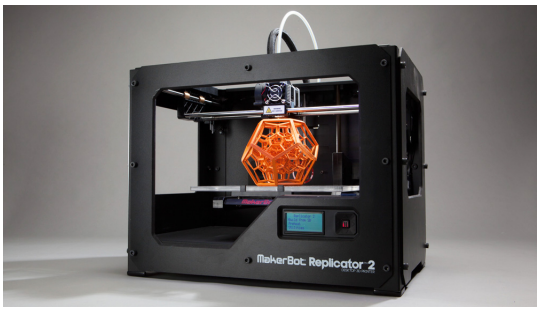

FIGURE 2 – L'imprimante utilisée dans notre projet

3 Design de la coque augmentée

3.1 Etude préliminaire

Pour commencer, nous avons remarqué que nos deux téléphones respectifs possédaient la technologie NFC. Nous sommes donc partit d'une coque déjà existante en choisissant une parmi une collection de modèle [3]. Puis nous l'avons modifié sur sketchup [4] un logiciel de modélisation gratuit et facilement accessible, de façon à ce que la coque ai une extension sur le dessus et le dessous pour l'emplacement des boutons et de la croix directionnelle. Après l'avoir modélisé nous avons rencontré différents problèmes d'ergonomie comme par exemple la prise en main. Nous hésitions aussi sur le choix de mettre deux ou quatre boutons à droite, finalement nous nous sommes décidé sur deux boutons.

3. `http://www.thingiverse.com/thing:388516`
4. `http://www.sketchup.com/fr`

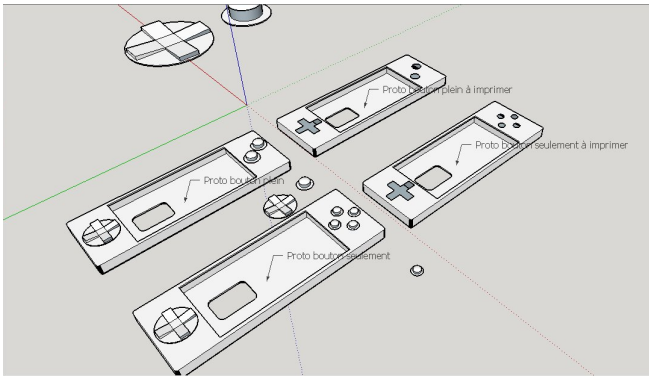

FIGURE 3 – Design de nos premières modélisation

La Figure 3 montre deux variations du modèle de base, avec deux ou quatre boutons avec à chaque fois deux versions : un rendu final et une version à imprimer.

3.2 Sondage

Nous hésitions sur la prise la main de la manette de jeu, nous avons donc construit des maquettes de manette de jeu en papier puis en carton :

FIGURE 4 – Bords rectangulaire

FIGURE 5 – Bords arrondie

Nous avons à la suite de ça effectué un sondage pour nous aider à choisir entre les deux coques réalisées, sur huit personnes venant de milieux diffèrents (étudiant, travailleur, enfants). Nous leur avons donné les coques individuellement et avec deux questions simple quel sont vos impressions sur la prise en main, puis quel est la meilleure prise en main ?

Pour la première forme avec les bouts carrés les intervenants ont trouvé que les bords inférieurs les gênaient ce qui n'était pas le cas pour la 2ème forme plus arrondie. Nous avons proposé une solution pour la prise main à l'arrière avec des poignées mais la plupart n'ont pas trouvé de différence. Nous n'avons donc pas retenue cette solution car les utilisateurs ne voyaient pas de différences et cela entraînerait un surplus de matériel et une complexité accrue lors de l'impression.

4 Prototypage rapide

Le concept de prototypage rapide consiste à créer une pièce puis de noter les avantages et les désavantages puis recommencer jusqu'à l'obtention d'une pièce satisfaisante [1]. Ce concept intègre trois notions telles que :

– Le temps, son objectif est de réaliser rapidement des modèles pour pouvoir gagner du temps dans le développement des produits.
– Le coût, car grâce à notre imprimante il est possible de créer une pièce puis de recommencer, nous pouvons donc faire rapidement des prototypes sans avoir recours à des outillages coûteux.
– La complexité des formes, les imprimantes 3D sont capables de réaliser des formes extrêmement complexes, irréalisables par des procédés industriel classique.

Grâce à cela on peut explorer les différentes variantes de notre objet afin d'obtenir la solution satisfaisante.

4.1 Partie modélisation

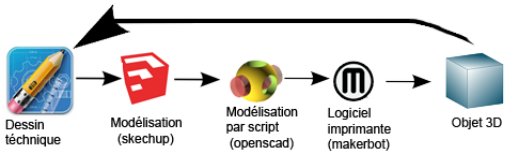

FIGURE 6 – Le cycle de modélisation

Dans cette partie nous avons procédé sous forme de cycle. Pour commencer, nous voulions imprimer la coque en entier puis la partie tête des boutons que l'on aurait rajouter ensuite. Nous avons réalisé sur papier les dessins techniques de la coque et des différents boutons.Puis nous l'avons modélisé à partir d'un modèle de coque d'un de nos téléphones sur SketchUp, pour ensuite transférer sur le logiciel de prévisualisation de l'imprimante pour avoir un rendu virtuel. Pour finir, nous l'avons donc imprimé. L'impression à pris environ 6 heures et le résultat était catastrophique. Les problèmes étaient simples : nous avons surestimé la qualité de l'imprimante. De plus, le filament s'était décroché pendant l'impression, ce qui a eu pour effet de ne pas terminer complètement notre coque. Nous avons compris les limites de l'impression 3D avec une imprimante comme celle-ci. Mais aussi, les erreurs que nous avions commise lors de la conception. Nous avons appris de nos celles-ci et recommencé en fonction des contraintes que l'imprimante nous imposait, c'est à dire faire des modèles plus grossier car la précision n'était point au rendez-vous.

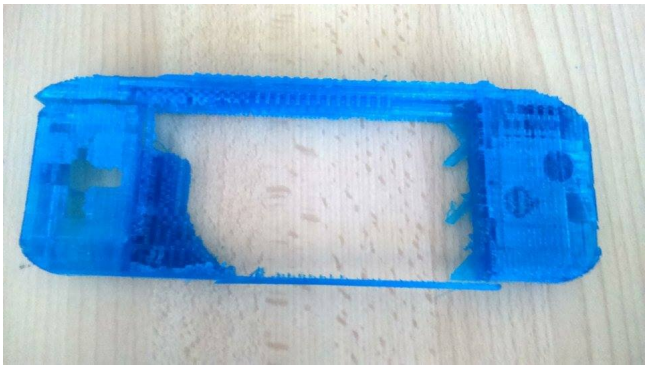

FIGURE 7 – Première impression

Suite à cette déconvenue, nous nous sommes donc moins précipités et nous avons imprimé éléments par éléments, c'est à dire les boutons individuellement par exemple. Nous avons également essayé différent matériaux pour la coque et pour les boutons (plus flexible par exemple).

Après plusieurs itérations de dessins, modélisations et d'impressions réalisées nous avons fini par avoir une solution qui fonctionne avec nos objectifs et qui n'entre pas en conflit avec l'imprimante. La dimension de l'imprimante nous a poussé à diviser la coque en deux. Comme nous aurions dût

répéter cet opération, nous avons choisi de réaliser cette opération de manière automatique avec un outil de script pour modèle 3D : OpenScad. Nous avons aussi fait le choix de séparer la coque de la partie contrôles avec les boutons (en module) nous avons réalisé la séparation et l'attache avec openscad.

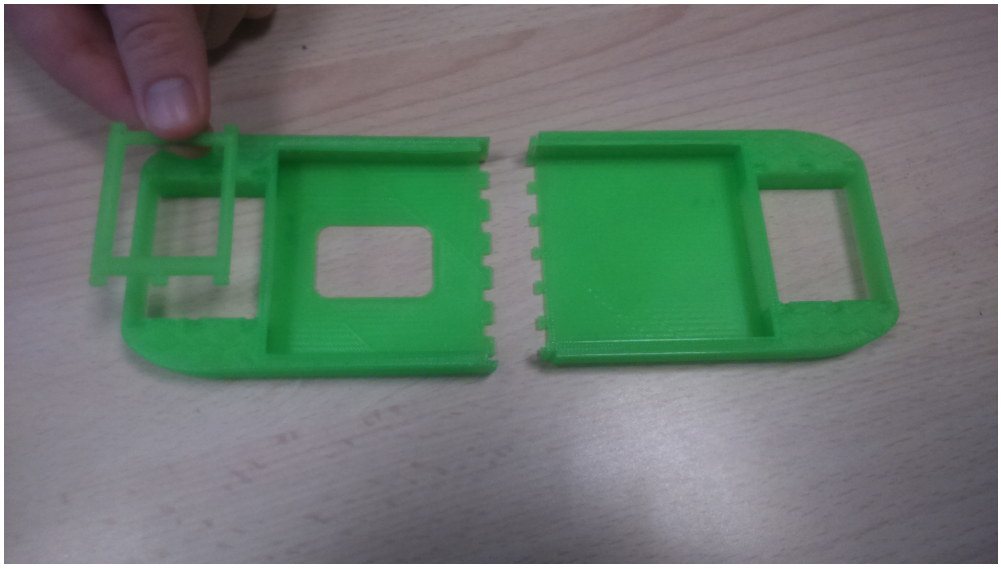

FIGURE 8 – Dernière impression

4.2 Partie électronique

Dans cette partie nous avons suivie un cycle plus rapide, dessin technique, création, test. Le but étant de créer un circuit pour les boutons. Il fallait que le bouton, lorsque l'on appuie dessus, ferme le circuit. Les tags NFC seraient alors alimenté et peuvent renvoyer un signal au téléphone. Une des premières questions que nous nous sommes posé est l'appuie simultané de plusieurs boutons. La réponse est simple il n'y a qu'un seul tag NFC qui est reconnu. Nous avons réfléchi à un moyen de palier ce problème, ce qui était compliqué sans circuit imprimé, trop de porte ET / OU ... (voir annexe). Pour des difficultés de reprogrammation du circuit nous n'avons pas pu le réaliser. Nous avons donc choisi de ne pouvoir appuyer que sur un seul bouton ce qui n'était pas gênant pour notre jeu de Tetris. Nous avons par la suite, créé des circuits avec une puce NFC puis nous les avons testés, nous avons eu des résultats satisfaisans.

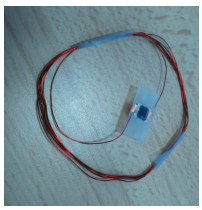

FIGURE 9 – Circuit NFC simple : bobine et Tag

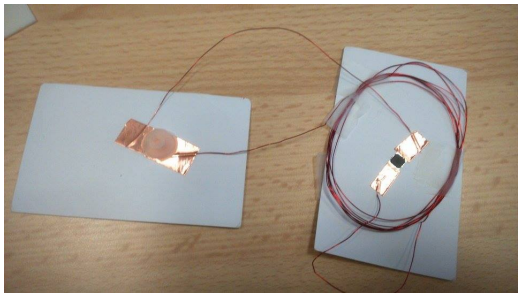

FIGURE 10 – Circuit NFC avec un bouton décentralisé

Nous avons eu l'opportunité de travailler avec l'université de Sarrebruck pour la conception de circuits imprimés à l'aide d'encre conductrice [2]. Ce qui nous permit d'avoir des circuits plats et donc de pouvoir les introduire dans notre coque. Après les avoir testé nous avons pu comprendre que plus la surface conductrice est importante plus l'inductance est élevé et donc la puce est bien alimentée. Et au contraire plus la surface conductrice est petite plus l'inductance est faible ce qui à pour effet de ne pas assez alimenter notre tag NFC et donc le circuit ne fonctionne pas. En conclusion, les circuits sont fonctionnel pour certaines configurations mais ils ne conviennent pas totalement à nos besoins, c'est à dire qu'il nous faudrait un circuit pouvant gérer jusqu'à quatre puces en même temps. La première impression nous a permis de tester la technique. Il faut donc maintenant adapter le circuit par rapport aux résultats trouvés. Par manque de temps nous allons juste essayé d'adapter notre circuit en ajoutant des fils.

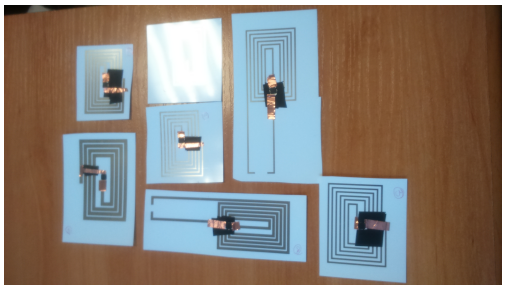

FIGURE 11 – Les différents circuits plats

5 Application

Nous avons réalisé une application de démonstration pour notre manette de jeu et Android qui permet de jouer à Tetris. Pour tester, nous avons utilisé 6 tags NFC : autant que nous avons d'action dans le jeu (Droite, Gauche, Faire descendre, Faire descendre d'un coup, échanger la pièce, et faire tourner la pièce).

Chaque tag étant associé à une action. Comme il le sera dans la manette de jeu, où chaque bouton aura une puce différente. Il a donc fallu faire une interface permettant d'associer un tag à chaque action.

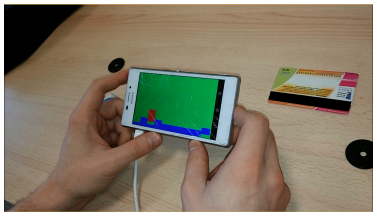

FIGURE 12 – Test de l'application avec 5 Tags (Voir la vidéo ici)

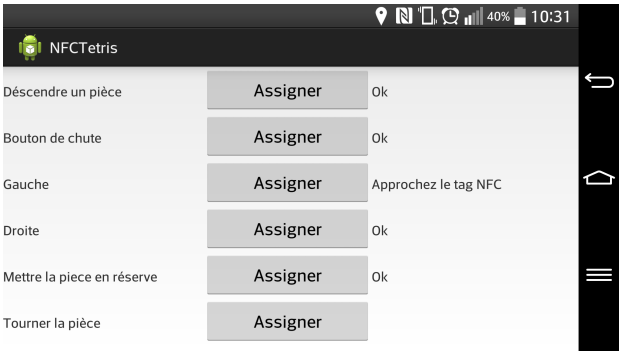

FIGURE 13 – Activité d'assignation

Le Tetris a utilise le pattern MVC (modèle , vue, contrôleur), nous avions déjà programmé ce jeu en java pour swing nous avons donc repris le modèle et refait la vue et le contrôleur.

5.1 Avant Le SDK 19

Une application Android est composé de pages ou « Activités ». On peut gérer un grand nombres d'évènements sur ces activités, comme la fermeture, la mise en pause, etc... Les Intents sont des messages asynchrones qui permettent aux composants d'une application de demander des fonctionnalités à d'autres composants Android. Les Intents permettent d'interagir avec nos propres composants ou des composants d'autres applications.

Le mécanisme est assez complexe, on enregistre statiquement une activité pour la réception d'intent de type `NfcAdapter.ACTION_NDEF_DISCOVERED` ou `NfcAdapter.ACTION_TECH_DISCOVERED`.

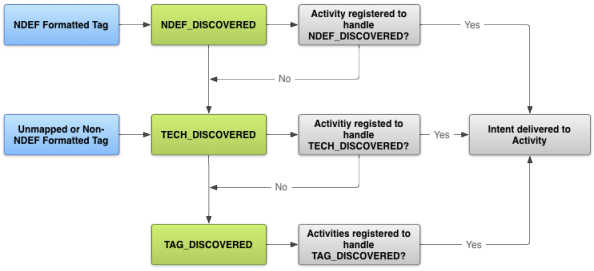

FIGURE 14 – Mécanisme de notification de découverte de tag en fonction du type

Lorsqu'un Intent arrive dans notre activité, il faut encore extraire l'information reçue. Un Intent peut contenir des informations supplémentaires sur le message qu'il transporte. Appelé « extra », ces informations sont organisées en ensemble de clés/valeurs. Ainsi le composant émettant le signal mettra en extra le Tag scanné. C'est ainsi qu'on arrive à retrouver l'objet « tag », et on pourra alors avoir directement son numéro de série ou les informations supplémentaires enregistré sur le tag.

5.2 Après Le SDK 19

Comme nous avons des téléphones récent, nous avons opté pour le nouveau système de découverte de tag. Ce système est apparu dans le SDK 19 (Android KitKat). Basé sur le pattern Observer/Observable, il est très simple d'utilisation. On enregistre l'observer grâce à cette méthode :

```
public void enableReaderMode (Activity activity, NfcAdapter.ReaderCallback callback,
int flags, Bundle extras)
```

On radie l'observer grâce à cette méthode :

```
public void disableReaderMode (Activity activity)
```

Ainsi à l'ouverture de l'activité (sur le onResume()) on démarre l'observation de la découverte de tags NFC. Et lorsque l'activité disparait du premier plan (évènement onPause(), car l'utilisateur a quitté en appuyant sur retour, ou qu'une autre activité a été lancé) on arrête l'observation.

Lors de la lecture du tag NFC, la méthode suivante est appelée :

```
public abstract void onTagDiscovered (Tag tag)
```

On a alors accès au tag, et on peut alors regarder toutes les données qu'il contient.

Un tag NFC est identifié par un numéro unique. Dans notre application, on reconnait un tag et on déduit l'action à effectuer grâce à cet identifiant. L'ensemble des tags est stocké dans un dictionnaire (une HashMap), et ce dictionnaire est stocké dans un fichier pour pérenniser les informations d'un lancement à l'autre.

6 Conclusion

Nous avons appliqué la méthode de prototypage rapide en utilisant des technique de conception avant-gardiste. Les contraintes au niveau de l'imprimante 3D étaient vraiment élevées. Des réglages très précis ont dût être effectué. Mais même avec des réglages très précis, il y a un caractère aléatoire sur le résultat. Par exemple : le filament qui sort de la tête d'impression, un structure qui s'effondre, etc... Nous avons mit beaucoup de temps à prendre l'imprimante en mains. Avec l'outil de prévisualisation, et à force d'essais, nous avons de mieux en mieux put prédire la réaction de l'imprimante face à notre modèle et ainsi nous avons pu contrer par des réglages spécifiques ou des dispositions non naturelles sur le plateau.

L'application Android a été réalisé plus rapidement que prévu sur le diagramme de Gantt (environ 1 jour). Un de nous deux avait déjà réalisé plusieurs applications dans le cadre de son stage. Nous avons donc juste découvert comment était géré la découverte de tag NFC dans Android. L'application est donc complètement fonctionnelle. Des graphismes ont pu être intégré (voir Figure 15).

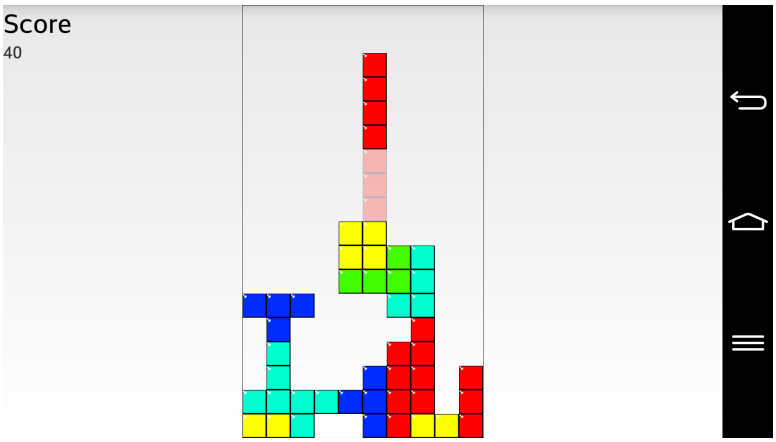

FIGURE 15 – Apparence finale

En ce qui concerne la gestion de projet, nous avons mit en place un repertoire sur la forge [5] où nous avons tenu un journal de bord quotidien. Ce qui nous a permit de résumer notre journée et de voir notre avancement au cours du temps. Nous avons essayé de suivre le diagramme de Gantt exposé dans le cahier des charges. Mais pour plusieurs raisons, dont l'impossibilité d'avoir accès à l'imprimante autant que nous le voulions, nous ne pouvions pas suivre le programme dicté par le diagramme de Gantt.

Pour finir, malgré de nombreuses difficultés, nous avons pu imprimer la coque finale avec succès. Un prototype simpliste fonctionnel à deux boutons a été réalisé.

5. `http://forge.univ-lyon1.fr/projects/p1106669-ter3dprint/`

Références

[1] Daniel Avrahami and Scott E Hudson. Forming interactivity : a tool for rapid prototyping of physical interactive products. In *Proceedings of the 4th conference on Designing interactive systems : processes, practices, methods, and techniques*, pages 141–146. ACM, 2002.

[2] Yoshihiro Kawahara, Steve Hodges, Nan-Wei Gong, Simon Olberding, and Jurgen Steimle. Building functional prototypes using conductive inkjet printing. *Pervasive Computing, IEEE*, 13(3) :30–38, 2014.

[3] Nicolai Marquardt, Alex S Taylor, Nicolas Villar, and Saul Greenberg. Rethinking rfid : awareness and control for interaction with rfid systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2307–2316. ACM, 2010.

[4] Shu-An Yeh, Hua-Ming Chen, Yi-Fang Lin, Yu-Chang Kao, and Jen-Yea Jan. Single-layer circularly polarized slot antenna for rfid reader application. In *Antennas and Propagation Society International Symposium (APSURSI), 2010 IEEE*, pages 1–4. IEEE, 2010.

7 Annexe

7.1 Différentes productions

FIGURE 16 – Coque avec filament flexible

Nous avons réalisé cette coque pour pouvoir tester un bouton sur le coté qui puisse changer par exemple le son la luminosité avec une connection nfc.

FIGURE 17 – Puce NFC avec un bouton poussoir

Nous avons essayé différents boutons pour pouvoir avoir un aperçu de ce que cela aller donner. Nous avons donc combiné le bouton créé avec l'un des circuits que nous avions réalisé et le résultat fonctionné.

7.2 Électronique

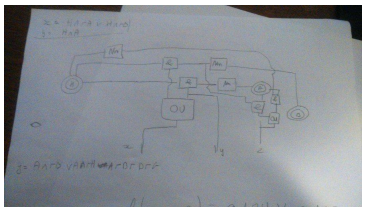

FIGURE 18 – Associer de porte ET / OU pour appuyer sur A et haut en même temps

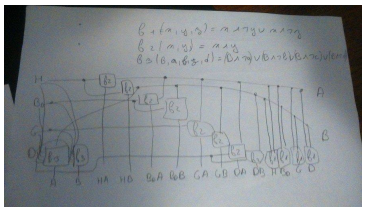

FIGURE 19 – Rendu final du nombre total de porte qu'il faudrai

7.3 Chaine de production, exemple : production finale

Tout d'abord nous avons conçu grossièrement un modèle respectant notre étude ergonomique et les dimension du smart-phone. À cette étape nous avons une simple coque de smartphone allongée sur les cotés. Les dimension sont celles du modèle original téléchargé sur Thingiverse auxquelles nous avons modifié l'ergonomie grâce à l'outil de modélisation "Sketchup".

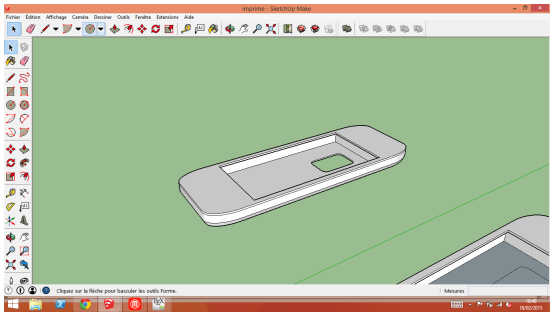

FIGURE 20 – Modèle grossier SketchUp

Une fois le modèle terminé, il faut exporter le modèle dans un fichier `.stl`. Malheureusement SketchUp n'exporte des fichiers « erroné », refusé par certains logiciels, c'est un problème récurrent. Il faut donc réparer le fichier grâce à un outil en ligne [6].

Nous avons maintenant un modèle répondant à la norme. Mais le modèle n'est pas terminé. Il faut encore lui appliquer les modifications pour insérer les modules de boutons et pour séparer le modèle en deux. L'imprimante étant trop petite pour imprimer le modèle en une fois.

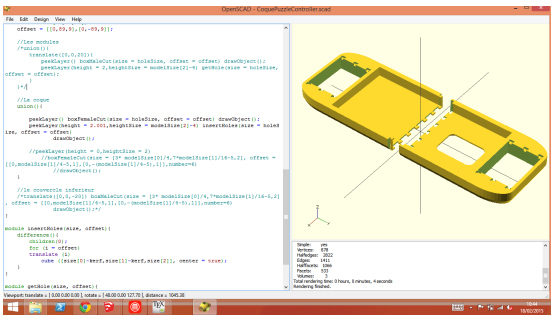

FIGURE 21 – Exécution du script sur le modèle

En cherchant un outil permettant l'opération voulue, nous avons trouvé une bibliothèque OpenSCAD permettant de clipser des pièces avec des pièces de puzzle [7]. openSCAD est un outil de script pour modèle 3D. Le langage SCAD que propose ce logiciel est relativement simple mais très puissant. Le langage SCAD propose en plus des structures de contrôles habituelles, des opérations Booléennes sur les solides (union, différence, intersection) et des opérateurs de transformations (Translation, Mise à l'échelle, Rotation). Au final le langage ressemble à OpenGL.

La bibliothèque de code téléchargé et modifié pour nos besoins :

```
//OpenSCAD PuzzleCut Library Demo - by Rich Olson
//http://www.nothinglabs.com
//Tested on build 2012.08.22
//License: http://creativecommons.org/licenses/by/3.0/
```

6. https://netfabb.azurewebsites.net/
7. http://www.3ders.org/articles/20130111-puzzlecut-lets-you-cut-object-into-smaller-pieces-for-3d-printing.html

```
module xMaleCut(offset = 0, cut = xCut1)
{
        difference()
        {
                children(0);
                translate([0,offset,0]) makePuzzleStamp(cutLocations = cut);
        }
}


module xFemaleCut(offset = 0, cut = xCut1)
{
        intersection()
        {
                children(0);
                translate([0,offset,0]) makePuzzleStamp(cutLocations = cut,
                        kerf = kerf);  //only set kerf on female side
        }
}


module yMaleCut(offset = 0, cut = yCut1)
{
        difference()
        {
                children(0);
                rotate ([0,0,90]) translate([0,offset,0]) makePuzzleStamp(cutLocations = cut);
        }
}


module yFemaleCut(offset = 0, cut = yCut1)
{
        intersection()
        {
                children(0);
                rotate ([0,0,90]) translate([0,offset,0]) makePuzzleStamp(cutLocations = cut,
                        kerf = kerf);  //only set kerf on female side
        }
}


module makePuzzleStamp(kerf = 0)
{
        difference()
        {
                //make the cube
                translate ([0,stampSize[0] / 2 - kerf,0])
                        cube (stampSize, center = true);

                //make the cuts
                for ( i = cutLocations )
                {
                        translate([i,0,0])
                                makePuzzle(kerf=kerf,height=stampSize[2]);
                }
        }
}

module boxMaleCut(size, offset = offset, number = 3){
        intersection()
        {
                children(0);
                for ( i = offset )
                {
                        makePuzzleStamp2(size = size,offset = i, number=number);
                }

        }
}

module boxFemaleCut(size = holeSize, offset = offset, number = 3){
        difference()
        {
                children(0);
                for ( i = offset )
                {
                        makePuzzleStamp2(size = size,offset = i,
                          kerf = kerf, number=number);    //only set kerf on female side
                }
        }
}

module makePuzzleStamp2(kerf = 0,size,offset,number)
{
        union()
        {
                //make the cube
                translate (offset)
                        cube ([size[0]-kerf,size[1]-kerf,size[2]], center = true);

                for (i = [offset[1]-size[1]/2+cutSize/2:
                                        (size[1]-cutSize)/(number-1):
                                        offset[1]+size[1]/2-cutSize/2]) {

                translate([offset[0]+size[0]/2,i,offset[2]])
                        rotate([0,0,-90])
                                makePuzzle(kerf=kerf,height=size[2]);

                translate([-(offset[0]+size[0]/2),i,offset[2]])
                        rotate([0,0,90])
                                makePuzzle(kerf=kerf,height=size[2]);

                }
        }

}

module makePuzzle(kerf=0,height=stampSize[2])
```

```
{
                        cube ([(cutSize / 2) - kerf * 2, cutSize - kerf * 2,height], center = true);
                        translate([0,cutSize / 2,0])
                                cube ([cutSize - kerf *  2,(cutSize / 2) - kerf * 2,height], center = true);
}
```

Le script permettant de dessiner tout ce dont on a besoin :

```
include <puzzlecutlib.scad>

modelSize = [77.11,218.36,15.29];

heigthCuts = 2;

stampSize = [500,500,100];              //size of cutting stamp (should cover 1/2 of object)

cutSize = 4;    //size of the puzzle cuts

xCut1 = [-34.5,-24,-12,0,12,24,34.5]; //locations of puzzle cuts relative to X axis center
yCut1 = [];     //for Y axis

kerf = -0.3; //supports +/-  numbers (greater value = tighter fit)
//using a small negative number may be useful to assure easy fit for 3d printing
//using positive values useful for lasercutting
//negative values can also help visualize cuts without seperating pieces

//makePuzzle(height = 2);
cutInTwo();     //cuts in two along y axis
//cutInFour();  //cuts in four along x / y axis
//drawComplexObject();

//comment out lines as needed to render individual pieces

module cutInTwo()
{
        translate([0,-6,0])
                xMaleCut() drawComplexObject();

        translate([0,6,0])
                xFemaleCut() drawComplexObject();
}

module cutInFour()
{
        translate([6,-6,0])
                xMaleCut() yMaleCut() drawComplexObject();

        translate([-6,-6,0])
                xMaleCut() yFemaleCut() drawComplexObject();

        translate([6,6,0])
                xFemaleCut() yMaleCut() drawComplexObject();

        translate([-6,6,0])
                xFemaleCut() yFemaleCut() drawComplexObject();
}

module peekLayer(heightSize = 2, height = modelSize[2]-2, stamp = 400){
        intersection(){
                children(0);
                translate([-stamp/2,-stamp/2,height]) cube([stamp,stamp,heightSize]);
        }
}

module drawComplexObject()
{
        holeSize = [45,34,14];
        offset = [[0,89,9],[0,-89,9]];

        //Les modules
        /*union(){
                translate([0,0,20]){
                        peekLayer() boxMaleCut(size = holeSize, offset = offset) drawObject();
                        peekLayer(height = 2,heightSize = modelSize[2]-4) getHole(size = holeSize, offset = offset);
                }
        }*/

        //La coque
        union(){

                peekLayer() boxFemaleCut(size = holeSize, offset = offset) drawObject();
                peekLayer(height = 2.001,heightSize = modelSize[2]-4) insertHoles(size = holeSize, offset = offset)
                        drawObject();

                //peekLayer(height = 0,heightSize = 2)
                        //boxFemaleCut(size = [3* modelSize[0]/4,7*modelSize[1]/16-5,2], offset = [[0,modelSize[1]/4-5,1],[0,-
                                //drawObject();
        }

        //le couvercle inferieur
        /*translate([0,0,-20]) boxMaleCut(size = [3* modelSize[0]/4,7*modelSize[1]/16-5,2], offset = [[0,modelSize[1]/4-5,1],[
                        drawObject();*/
}

module insertHoles(size, offset){
        difference(){
                children(0);
                for (i = offset)
                translate (i)
                        cube ([size[0]-kerf,size[1]-kerf,size[2]], center = true);
        }
}

module getHole(size, offset){
        for (i = offset)
                translate (i)
                        cube ([size[0],size[1],size[2]], center = true);
}
```

```
module drawObject(){
        import("coqueSanstrou_fixed.stl");
}
```

Une fois que le résultat nous convient, il faut l'exporter en fichier `.stl`. Ensuite on l'importe sur Makerware afin de régler l'impression, divers réglages sont nécessaire en fonction du matériau du filament utilisé. On peux aussi voir la quantité de matériel utilisé et une approximation du temps d'impression. L'intérêt est surtout de voir l'aperçu du résultat final, qui devra potentiellement être rééditer sur openSCAD ou Sketchup

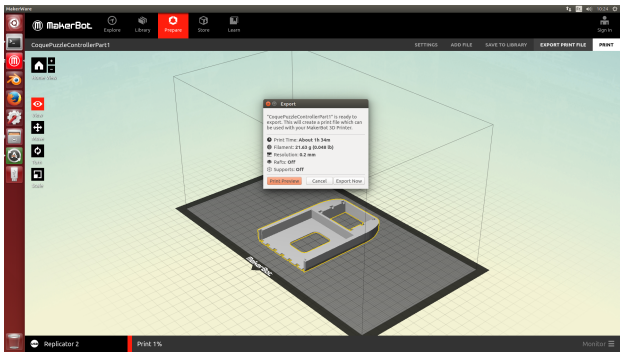

FIGURE 22 – Les données sur le modèle que l'on imprime

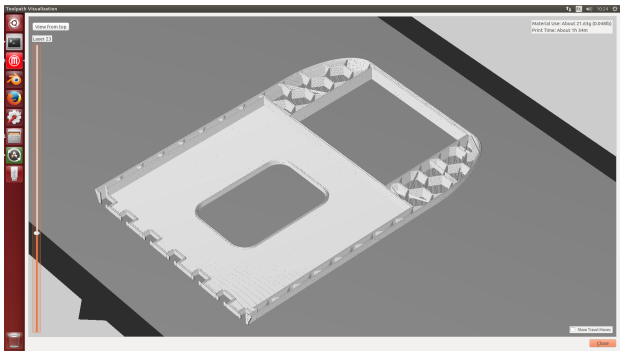

FIGURE 23 – Prévisualisation des couches - Structure en alvéoles pour les volumes pleins

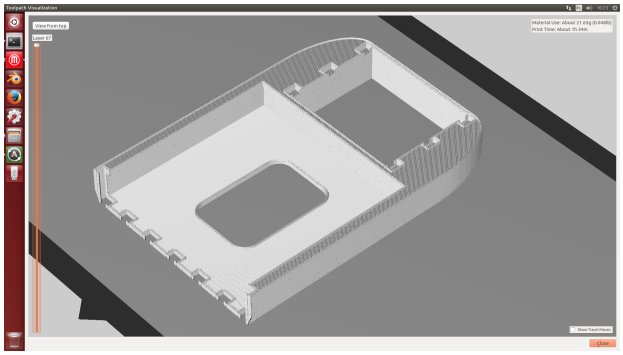

FIGURE 24 – Previsualisation du modèle terminé