Master 2 Informatique, E-services
Université des Sciences et Technologies de Lille 1

AppInventor

Projet de Génie Logiciel et Interactions Homme-Machine (GLIHM)

Baptiste CARLIER & Alban DE UBEDA
2010-2011

Remerciements

- Nous souhaitons tout d'abord remercier M. Tarby, responsable et initiateur des projets Génie Logiciel et Interactions Homme-Machine de notre Master 2 E-services.

- De même, nous tenons à remercier l'ensemble de nos camarades auprès de qui nous avons pu échanger informations et points de vue.

- Aussi, nous voudrions remercier nos familles et amis pour leur généreuse participation dans les tests d'AppInventor compte tenu du fait qu'ils ne connaissaient pas l'outil de Google.

- Enfin, nous aimerions remercier les communautés Android, à la fois celle du forum officiel AppInventor mais aussi la communauté en générale. Nous apprécions la qualité et la rapidité de réponse à nos questions.

Résumé

AppInventor est une application fournie par Google depuis quelques mois permettant à quiconque de créer des applications logicielles pour le système d'exploitation Android.**Son installation est rapide et son utilisation, via une connexion Internet, une application web et une application fenêtrée est simple.**

Nous avons réalisé trois principales applications pour tester l'outil développé par Google. L'application *ShareAccelerometer* nous a servi à prendre en main AppInventor et à tester, entre autre, le capteur accélérométrique et l'envoi de SMS. L'application *MoveTheBall* tente de vérifier si la cible des utilisateurs d'AppInventor est aussi jeune qu'elle semble l'être. Enfin, l'application *MessageDirectory* se veut volontairement plus complexe. Elle est destinée aux utilisateurs experts car elle mélange AppInventor, développement classique pour Android (avec l'IDE Eclipse) et site web avec base de données.

De ces travaux, nous tirons la conclusion qu'**AppInventor reste un outil marginal**. En effet, pour les développeurs confirmés, même si il est pratique, il reste trop limité en termes de fonctionnalités. Aussi, pour les novices (enfants, non informaticiens, etc.), l'outil ne semble pas adapté car il requiert des compétences basiques de programmation (notions de fonctions, variables et conditions).

Sommaire

Introduction 1

I. Débuter avec AppInventor 2

A. AppInventor, qu'est-ce que c'est ? 2

B. Comment l'installer ? 3

1. Version beta 3

2. Version publique 3

C. Comment utiliser ? 3

1. Interface web 3

2. Fenêtre Scratch 7

3. Tester et installer 9

II. Applications réalisées 10

A. ShareAccelerometer 10

1. But du projet 10

2. Réalisation 10

B. MoveTheBall 11

1. But du projet 11

2. Réalisation 11

C. MessageDirectory 12

1. But du projet 12

2. Réalisation 12

III. Utile et utilisable ? 14

A. Pour un développeur, comparaison avec le développement standard 14

B. Pour un novice 18

C. Quels usages pour AppInventor ? 20

Conclusion 21

Annexes 22

Introduction

Le projet que nous avons réalisé avait pour but de tester l'outil « AppInventor », développé par Google. Cette application se base principalement sur un site Internet. En utilisant les composants graphiques et multimédias mis à disposition, les utilisateurs peuvent se mettre à la place des développeurs de programmes Android et faire leurs propres applications.

Quels sont les possibilités, les limites, les usages de cet outil ? Notre travail tente de répondre à ces questions.

Celui-ci comprend plusieurs étapes : création d'applications basiques pour prendre en main et utiliser les composants, création d'une application plus complexe pour trouver les limites de l'outil et tests avec des sujets non informaticiens.

I. Débuter avec AppInventor

A. AppInventor, qu'est-ce que c'est ?

Avant de préciser ce qu'est AppInventor, il est nécessaire de comprendre la logique commerciale du système d'exploitation mobile Android. Celui-ci se veut être un maximum libre. Cette notion se décline sous plusieurs angles :

- Le code du système est Open Source : libre redistribution d'accès au code source et de travaux dérivés
- Le SDK (Kit de développement) est disponible gratuitement ainsi, le développement d'applications est abordable.
- Ce système est embarqué sur des smartphones de différents constructeurs : *HTC*, *Samsung*, *Acer*, *LG*, etc.
- Il est généralement possible d'installer n'importe quelle application sur son téléphone fonctionnant sous Android sous condition de disposer du fichier d'installation (*.APK*).

Nous comprenons alors mieux le dessein d'AppInventor : **permettre aux utilisateurs lambda de créer leurs propres applications**.

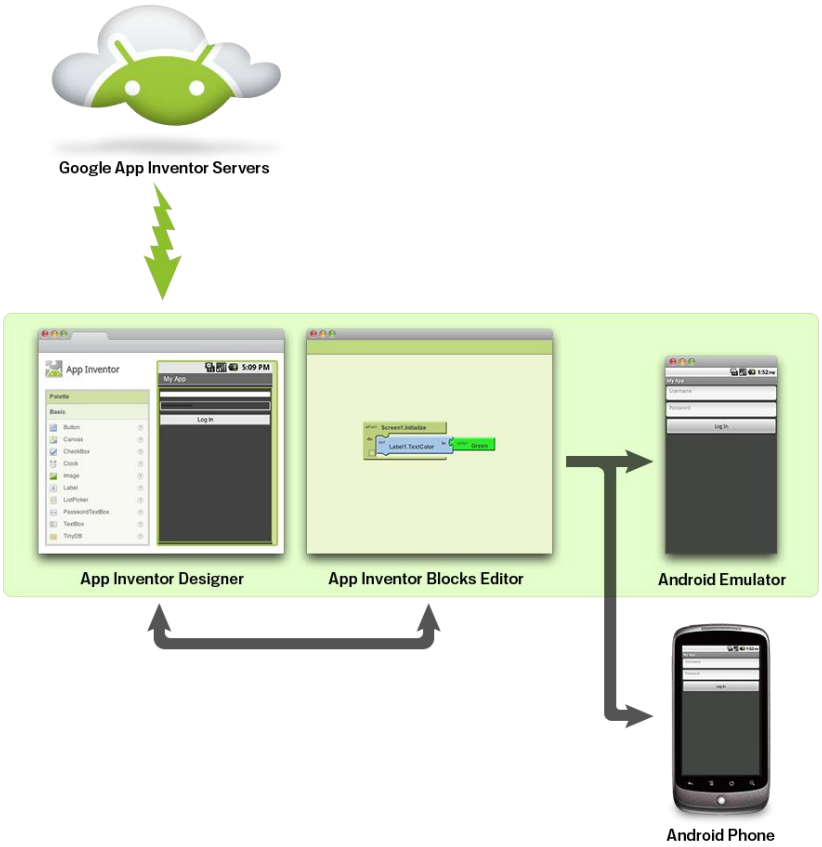

Figure 1 : Fonctionnement d'AppInventor

AppInventor est une **application à la fois web et fenêtrée,** conçue par Google, qui permet à chacun de créer son application personnalisée pour le système d'exploitation Android. Il utilise entre autre une interface

graphique − similaire à *Scratch*et *StarLogo TNG* −, permettant d'effectuer des Drag-and-Drop (Glissé-Déposé) d'éléments visuels. Il a été rendu disponible sur demande à partir du 12 Juillet 2010 et a récemment été rendu publique depuis le 15 Décembre 2010. L'utilisabilité est orientée **vers les personnes qui ne sont pas familières avec la programmation informatique**, tels que l'école primaire ou les élèves en générale. Le raisonnement veut que si les jeunes développent des applications pour répondre à leurs besoins propres et arrivent à les installer sur leurs propres téléphones, ils sont susceptibles d'utiliser ces téléphones plus souvent et, si ce n'est pas encore le cas, de passer à l'OS Android.

Nous allons maintenant voir comment installer AppInventor et comment « développer » avec.

B. Comment l'installer ?

1. Version beta

Au lancement beta d'AppInventor, **il était nécessaire d'effectuer une demande d'accès**. Nous nous étions déjà personnellement investis dans ce projet et avions eu une réponse positive dès le 30 Juillet 2010. Nous avions alors un accès complet avec nos comptes Google personnels.

Cet accès n'étant pas toujours garantis, M. Tarby a eu la mauvaise surprise de ne pas avoir de code d'accès à nous fournir pour débuter ce projet de GLIHM et devait l'annuler. Heureusement, nous avions fait la demande et avions déjà des accès.

Une fois l'accès obtenu, il n'y avait rien de requis pour utiliser *AppInventor Designer* (l'éditeur graphique). Cependant, il était tout de même nécessaire de télécharger et installer le SDK d'Android pour utiliser *AppInventor Blocks Editor* (l'éditeur de *blocks* – partie logique). La JRE (*Java RuntimeEnvironment*) était aussi nécessaire.

2. Version publique

Le 15 Décembre, cinq mois après l'ouverture d'AppInventor sur demande, **le site a été ouvert au publique**. De ce fait, tout le monde peut maintenant travailler avec AppInventor.

Aussi, pour simplifier encore plus l'utilisation de l'outil, il n'est maintenant plus requis de disposer du SDK Android entier. **Un installateur spécifique, *AppInventor Setup*, disponible sur le site[1], permet d'installer tout le nécessaire pour utiliser l'éditeur de *blocks*.**

Si vous ne disposez pas de smartphone sous Android, vous pourrez tester vos applications sur l'**émulateur d'Androphone** que l'installateur aura aussi placé sur votre machine.

La JRE reste nécessaire.

C. Comment utiliser ?

1. Interface web

Premièrement, il faut se rendre sur le site d'AppInventor[2] et s'y connecter à l'aide son compte *GMail*.

[1]http://appinventor.googlelabs.com/learn/setup/#setupComputer

Nous avons beaucoup d'informations sur la page principale, notamment une vidéo de présentation et des liens vers des aides, plutôt bien faites mais en anglais. Pour créer une application, cliquez sur le lien « MyProjects ».

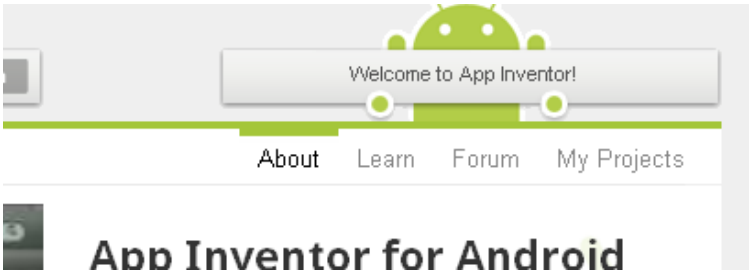

Figure 2 : Menu d'AppInventor

Nous arrivons alors sur une page nous présentant tous les projets AppInventor du compte, pour en créer un nouveau il suffit de cliquer sur « New ».

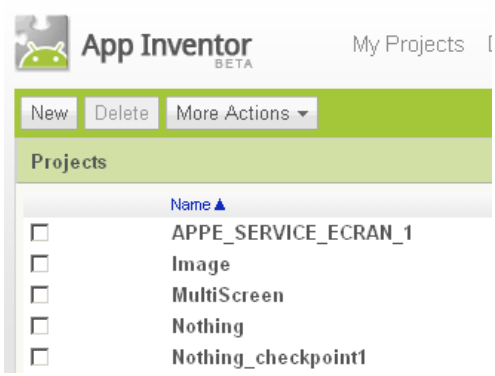

Figure 3 : Projets dans AppInventor & actions possibles

Nous entrons le nom du projet et c'est parti! Nous pouvons aussi charger et télécharger des projets. De cette manière, comme deux comptes ne peuvent pas travailler ensemble sur le même projet, nous pouvons tout de même transmettre notre projet à un collègue.

Attention : le projet n'est pas exportable en code Java réutilisable par un IDE standard.

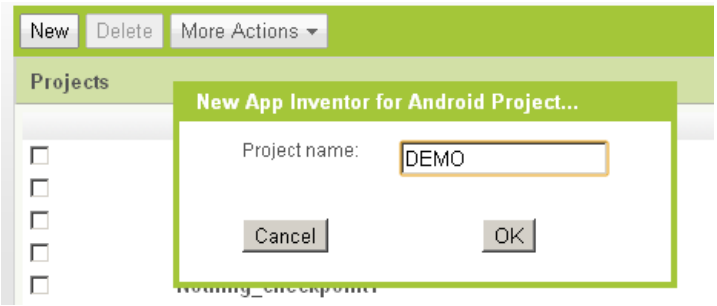

Figure 4 : Création d'un nouveau projet

Pour créer une application, la première phase est de la designer. Pour cela le site affiche un écran de téléphone dans lequel nous pouvons placer, par Drag-and-Drop, les éléments que nous voulons afficher.

[2] http://appinventor.googlelabs.com/about/

Il y a des éléments graphiques comme des boutons, des labels, … ainsi que beaucoup d'éléments non graphiques, comme des sensors (accéléromètre, orientation, géo localisation) ou des fonctions permettant d'effectuer des actions : lecteur audio, lanceurs d'applications basiques du téléphone (contact picker, camera, barecode scanner, etc). L'annexe X établit une liste complète des composants disponibles. Il est important de savoir que certains composants sont parfois ajoutés. Nous parlons ici de la section « LEGO® MINDSTORMS® » qui a été ajoutée dans AppInventor depuis que nous avons commencé ce projet.

Pour résumer, cet écran représente à la fois les services implantés dans l'application (capteurs) et les XML d'une application Android classique :

- Le XML du layout de l'interface.
- Le *manifest* avec la liste des permissions (envoie de SMS).

Nous allons vous détailler tout le processus de création d'une petite application avec AppInventor. Cette application toute simple affichera la valeur de l'angle d'inclinaison du téléphone quand l'utilisateur appuiera sur un bouton.

Nous continuons sur le projet DEMO commencé précédemment. Tout d'abord nous mettons un label qui permettra d'afficher la valeur. En cliquant sur le bouton d'aide du composant, une petite description du composant s'affiche.

Figure 5 : Aide rapide sur un composant

Encore une fois, elle est en anglais, comme l'ensemble d'AppInventor, ce qui le rend difficile à utiliser pour les personnes non-anglophones, notamment les plus jeunes. Nous notons cependant que le texte en lui même n'est pas très compliqué à comprendre même si il utilise des termes informatiques.

Pour placer le label sur le téléphone, il suffit d'effectuer un Drap-and-Drop vers l'écran.

Figure 6 : Drag-and-Drop d'un composant Label

Pareillement, nous pouvons afficher l'aide du bouton et l'ajouter aussi sur l'écran.

Comme écrit dans l'aide, nous pouvons modifier de nombreuses propriétés des composants. Voici un extrait des propriétés modifiables pour un bouton :

Figure 7 : Propriétés d'un Label

Nous avons la possibilité, par exemple, de changer simplement le texte ou la couleur mais beaucoup d'autres options sont modifiables.

Finalement nous ajoutons un composant permettant d'obtenir l'orientation du téléphone : l'*OrientationSensor*. Nous le glissons sur le téléphone, celui-ci n'étant pas visible, il s'affiche sous l'écran. Le fait de l'ajouter permet simplement de pouvoir accéder à ses méthodes dans la seconde étape du développement.

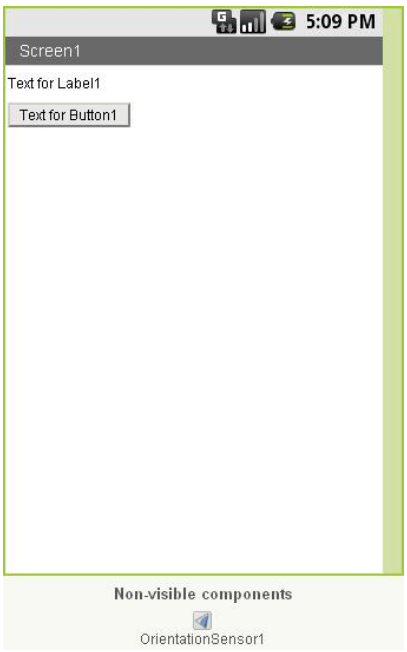

Figure 8 : Résultat du designer de l'application DEMO

Une fois les composants mis en place et désignés, nous passons à la deuxième phase de développement d'une application via AppInventor : l'interface Scratch. Pour cela, il faut cliquer sur "Open the Blocks Editor" en haut à droite de la page.

Figure 9 : Menu du Designer

Il faut alors télécharger un fichier .JNLP (Java Network Launching Protocol) permettant de lancer une application Java depuis le web. Une fois téléchargée, nous pouvons lancer cette application fenêtrée.

Figure 10 : Téléchargement de l'AppInventorForAndroidCodeblocks.jnlp

L'application se lance et demande le chemin du dossier où aura été installée l'*AppInventor Setup*. Comme exprimé précédemment, avant de pouvoir utiliser AppInventor, il est nécessaire d'installer un logiciel sur la machine. Le logiciel en question est appelé *AppInventor Setup*. Il est disponible pour :

- Macintosh (Intel processor): Mac OS X 10.5, 10.6
- Windows: Windows XP, Windows Vista, Windows 7
- GNU/Linux: Ubuntu 8+, Debian 5+

2. Fenêtre Scratch

L'interface Scratch permet d'imbriquer des éléments graphiques entre eux pour effectuer la partie programmation de l'application à développer.

Figure 11 : AppInventor, éditeur de blocks

Comme vous pouvez le voir sur la *Figure 11 : AppInventor, éditeur de blocks*, cette interface est très simple et épurée. En effet, en haut on retrouve des éléments classiques – « Save », « Undo » et « Redo » ainsi qu'un bouton de test pour lancer l'application sur le mobile ou sur l'émulateur.

Lorsque cette interface est lancée, il nous est demandé si nous souhaitons tester l'application sur un mobile (à brancher après l'ouverture de la fenêtre et dont les pilotes doivent préférablement être installés) ou sur l'émulateur fourni.

Aussi, sur la gauche, nous avons un système d'onglets. Nous y retrouvons les différents composants (blocks) que nous avions pris soin de placer dans le designer – « My Blocks » – et des blocks utilitaires – « Built-in ».

Figure 12 : Menus de l'éditeur de blocks

Dans l'onglet « My Blocks » de notre projet DEMO, on retrouvera nos éléments et leurs accesseurs & fonctions :

- MyDefinitions : variables & procédures globales.
- Button1 : variables & procédures spécifiques au bouton.
- Label1 : variables & procédures spécifiques au label.
- OrientationSensor1 : variables & procédures spécifiques au capteur.
- Screen1 : variables & procédures spécifiques à l'écran (assez restreint).

Dans l'onglet « Built-in », nous retrouverons toujours les mêmes éléments :

- Definition : morceaux permettant de définir des procédures (avec/sans résultats/attributs).
- Text : morceaux permettant de traiter du texte. Assimilables au type *char* et à la classe *String* en Java.
- Lists : morceaux permettant de traiter des listes. Assimilables aux sous-classes de *List* en Java.
- Math : morceaux permettant de traiter des nombres. Assimilables au type *int* et à la classe *Integer* en Java.
- Logic : morceaux permettant de traiter des booléens. Assimilables au type *boolean* et à la classe *Boolean* en Java.
- Control : outils permettant d'effectuer de la programmation conditionnelle.
- Colors : couleurs.

Aussi, la corbeille est utilisée pour jeter des morceaux de pseudo-code et la loupe sert à zoomer/dézommer sur l'éditeur.

Pour en revenir à notre projet DEMO, nous avions exprimé le souhait que notre application « affiche la valeur de l'angle d'inclinaison du téléphone quand l'utilisateur appuiera sur un bouton ». Ce qui se traduit en pseudo-code par : `When Button1.Click do setLabel1.Text(OrientationSensor1.Roll)`. C'est à quelques détails près ce que nous pouvons faire à l'aide de Drag-and-Drop dans l'éditeur de blocks :

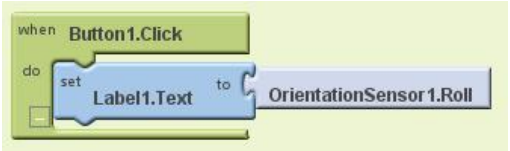

Figure 13 : Blocks du projet DEMO

Il s'agit là d'un exemple extrêmement simple.

3. Tester et installer

Nous distinguons bien les termes « tester » et « installer » car il n'est pas nécessaire de disposer du fichier d'installation de l'application (le *.APK*) pour l'installation. Lorsque nous avons branché notre téléphone à l'ordinateur et sommes passé en mode Debug (*Paramètres>Applications>Développement>Débogage USB*), une icône apparaît en haut de l'éditeur de *blocks*, *cfFigure 14 : Téléphone connecté*.

Figure 14 : Téléphone connecté

Figure 15 : Application en test

Lorsque nous cliquons dessus, l'application est compilée, envoyée au téléphone pour être installée et lancée dessus, *cfFigure 15 : Application en test*. Cependant, une fois celle-ci quittée, elle n'est plus disponible sur le mobile. L'application est donc temporairement installée.

Pour installer son application, il y a trois solutions. Toutes trois démarrent sur l'interface web, en haut à droite de la fenêtre, via le bouton *Package for Phone* :

Figure 16 : Compiler l'application

- En cliquant sur « Show Barcode », l'application sera empaquetée et, après quelques secondes, une fenêtre avec QRCode apparaîtra sur l'interface web. En le scannant avec une application du mobile, le QRCode vous redirigera sur une page web. Vous devrez alors vous identifier avec le même compte Google que celui qui développe le projet pour pouvoir télécharger le fichier *.APK*.
- En cliquant sur « Download to this Computer », l'application sera empaquetée et une fenêtre de téléchargement vous permettra de sauvegarder le fichier *.APK* sur votre ordinateur.
- En cliquant sur « Download to Connected Phone », l'application sera empaquetée et installée sur le téléphone Android connecté à l'ordinateur. Le téléphone doit là encore être passé en mode Debug.

Pour ces trois manipulations, il est nécessaire que l'éditeur de *blocks* soit encore ouvert.

II. Applications réalisées

Afin de tester les différentes possibilités d'AppInventor, nous avons réalisé quelques applications. En voici quelques unes.

A. ShareAccelerometer

1. But du projet

Le dessein de ce projet était de prendre en main l'interface et de voir les possibilités d'une multitude de composants. Parmi eux : la reconnaissance vocale, l'envoie et la réception de SMS, le notifieur, etc.

L'application devait permettre d'envoyer un SMS à un contact pour lui montrer les valeurs maximums que nous avons atteint sur l'accéléromètre. La personne doit être sélectionnée dans la liste de contacts du téléphone et une partie du contenu du SMS sera rédigé à la voix. Des sons et notifications viendraient illustrer l'application.

2. Réalisation

Au début, nous avions ajouté beaucoup de composants à cette application. Par la suite, nous avons nettoyé la liste pour arriver à cette dernière :

- Screen1
  - VerticalArrangement1
    - HorizontalArrangement1
      - Label
      - Button1
  - TextBox
  - PhoneNumberPicker1
  - Button
  - Button
  - SpeechRecognizer
  - Texting1
  - AccelerometerSensor1
  - Sound1
- Notifier1

Le développement de cette application nous a permis de **mettre en valeur de bons et de mauvais points pour AppInventor**.

Le côté positif est qu'il permet un **accès extrêmement simple aux composants** graphiques (boutons, etc.) et à quelques services très amusant (SMS, reconnaissance vocale). Grace aux *blocks*, il est très simple d'envoyer un SMS (trois lignes). Un autre point fort est que nous pouvons **facilement personnaliser les boutons et jouer des sons en ajoutant des fichiers image et son dans l'interface web**.

Malheureusement, nous avons aussi trouvé quelques faiblesses à ce type de développement. Il n'est **pas possible de créer de variables locales**. Nous sommes toujours obligés, même dans les procédures créées par nos

soins, d'utiliser des variables globales. Aussi, pour des fonctions assez simples, les **tailles de celles-ci prennent vite de la place**. On remarque cependant qu'il est possible de réduire les fonctions et de réorganiser les éléments.

B. MoveTheBall

1. But du projet

Nous nous sommes mis dans la peau d'un enfant souhaitant réaliser une petite application simple : faire **bouger une balle sur l'écran grâce à l'orientation du téléphone**.Le but de ce scénario est d'établir si un enfant peut utiliser AppInventor seul pour faire ne serait-ce qu'une application très simple.

Nous avons eu l'idée de cet application car les composants *Canvas* et *Ball* d'AppInventor sont spécifiquement faits pour ce type de développement : la balle se déplace sur ce qui pourrait être appelé « un plateau », représenté ici par le canvas.

2. Réalisation

Dans un premier temps, nous avons créé l'interface. Voici son architecture :

- `Screen1`
  - `Canvas1`
    - `Ball1`
  - `OrientationSensor1`

Le canvas est la zone où la balle peut se déplacer. Il remplit l'écran en largeur et possède une hauteur de 200 pixels.

Ensuite, avec l'éditeur de *blocks*, nous avons utilisé une fonction très pratique : `OrientationSensor1.OrientationChanged()`. Cette fonction est appelée à chaque changement d'orientation du téléphone. Elle est donc très sollicitée.

Après quelques tests, le résultat est concluant. Lorsque le téléphone s'incline sur la droite ou sur la gauche avec un angle supérieur à 10°, la balle se déplace sur l'axe X. Lorsque le téléphone s'incline vers l'avant ou vers l'arrière avec un angle supérieur à 10°, la balle se déplace sur l'axe Y.

La conception de cette petite application n'a posé **aucun problème particulier**. Nous avons juste perdu quelques minutes à déterminer à quoi correspondaient les angles *Pitch*, *Roll* et *Yaw* du capteur d'orientation et à connaître leurs valeurs lorsque le téléphone est à plat.

Voici l'algorithme que nous avons pu mettre en place après quelques minutes sur l'interface Scrach :

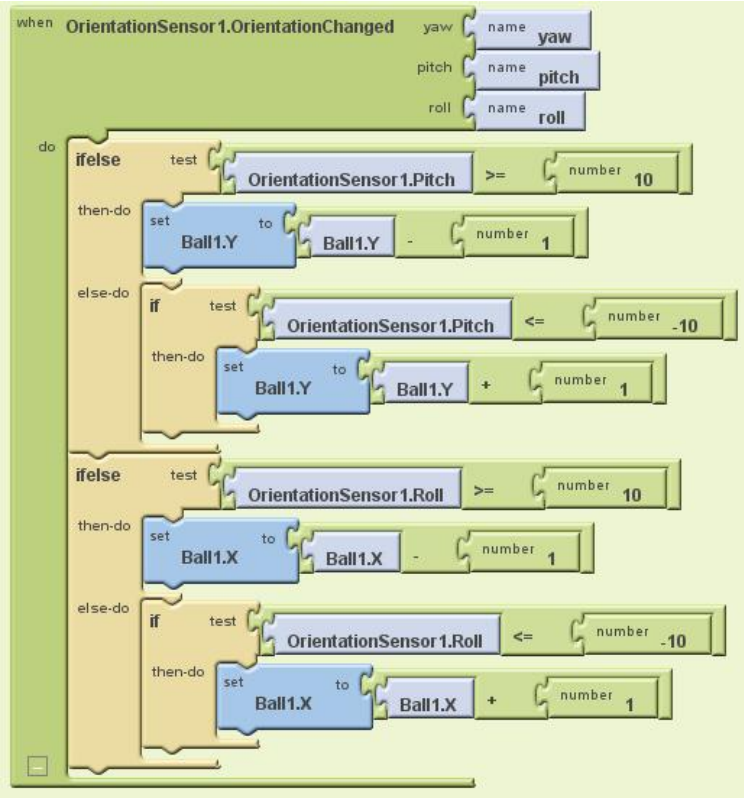

Figure 17 : Blocks de l'application MoveTheBall

D'un point de vue développeur, un petit problème se pose : **il n'y a pas la possibilité de fixer l'écran dans une orientation par défaut** (portrait ou paysage). Ainsi, lorsque le téléphone est trop incliné vers la droite ou la gauche, l'écran passe en mode paysage détériorant le concept car le rendu est faussé.

Aussi, nous nous posons réellement la question de savoir si un enfant d'une douzaine d'années pourrait coder cette application. Les notions de fonctions, variables et conditions sont certes des concepts de base en programmation mais, si ils sont inconnus, peuvent être difficiles à utiliser pour un enfant.

C. MessageDirectory

1. But du projet

Les applications précédentes se voulaient assez accessibles. Pour cette dernière, **nous avons voulu augmenter la difficulté**. Le principe de cette application est de pouvoir poster un message à un endroit, à un moment donné. Ce message ne doit être accessible en lecture que si on se trouve près de là où il a été posté.

Nous allons voir quels problèmes se sont posés et comment nous avons pu y faire face.

2. Réalisation

Dans un premier temps, nous souhaitions qu'un utilisateur A partage seulement avec un utilisateur B un message visible uniquement dans le rayon où il se trouve. Ce processus devait utiliser les mails ou les SMS pour partager le message (sans l'afficher) et la localisation.

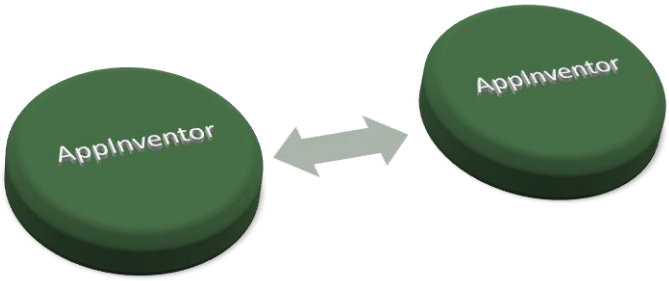

Figure 18 : MessageDirectory version 1

Un premier problème s'est posé : **comment transmettre le message sans l'afficher ?** Avec les outils d'AppInventor, c'était proprement impossible. Nous avons donc pensé à **mettre en place un serveur web** qu'il faudrait joindre pour envoyer/recevoir les messages.

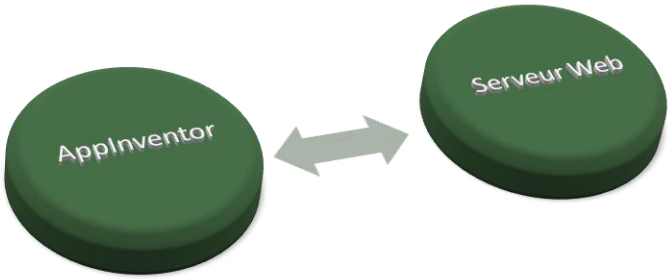

Figure 19 : MessageDirectory version 2

Malheureusement, un second problème s'est posé : **comment joindre le serveur dans l'application AppInventor ?** À nouveau, aucun outil ne correspond à cette fonctionnalité dans AppInventor. De ce fait, nous avons eu la possibilité, mais surtout la nécessité, d'utiliser le composant *ActivityStarter* d'AppInventor.

Ce composant permet de **lancer une autre *Activity*** (d'une autre application) disponible sur le téléphone. Ainsi, en développant nous même une application en Java, nous pourrions communiquer avec le serveur Web.

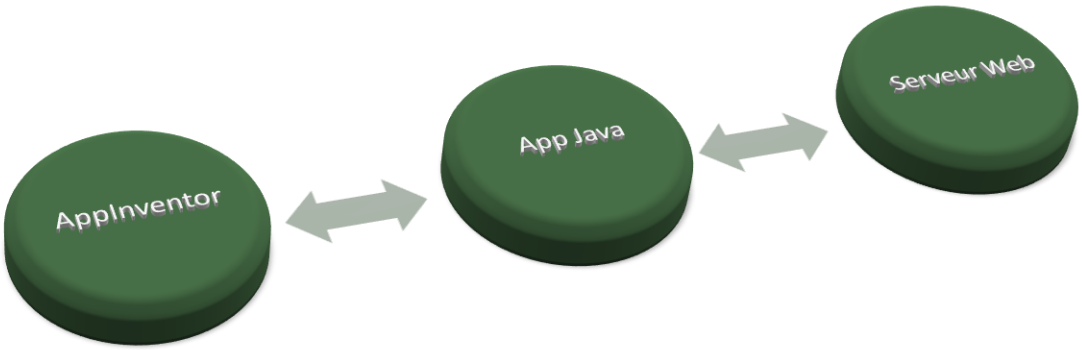

Figure 20 : MessageDirectory version finale

De ce fait, le message et la géolocalisation se font dans l'application AppInventor, le tout est transmis à l'application en Java qui l'envoie au serveur. Celui-ci répond favorablement ou non à la requête. L'application java transmet la réponse à l'application AppInventor et l'utilisateur a enfin une réponse.

Pour obtenir les messages se trouvant à proximité, l'utilisateur clique sur le bouton « Rafraichir » pour envoyer à l'application Java les coordonnées de géolocalisation. L'app Java va contacter le serveur qui lui

retournera les messages datant d'une semaine, disponibles dans un rayon d'un kilomètre. Enfin les messages seront retournés à l'application AppInventor qui traitera le retour.

Pour réaliser cette application, nous étions **partis du principe que tous les composants étaient disponibles dans AppInventor**. Nous nous sommes vite aperçus que ce n'était pas du tout le cas et avons dû changer l'architecture du projet, à plusieurs reprises. Ceci étant, nous avons aussi pu utiliser un composant intéressant (*ActivityStarter*) qui nous a pris beaucoup de temps à comprendre.

En effet, lancer une *Activity* externe à l'application AppInventor n'est pas réellement difficile mais passer des paramètres en entrée/sortie n'est pas très bien décrit dans l'aide disponible sur le site web[3]. Après quelques recherches et envois de messages sur des forums, nous avons trouvé une solution. Cependant, il s'avère que **le passage de paramètres en entrée/sortie est moins complet avec AppInventor** qu'en développement de façon classique en Java.

III. Utile et utilisable ?

A. Pour un développeur, comparaison avec le développement standard

En master E-services, nous avons pu travailler à de nombreuses reprises sur la plateforme Android avec une programmation classique (en Java, avec l'IDE Eclipse). Grace à cette expérience, nous allons pouvoir déterminer l'utilisabilité d'AppInventor par rapport à une programmation plus classique.

Au début, ce qui saute aux yeux est la simplicité et la rapidité avec laquelle nous « programmons » sous AppInventor. L'utilisation de composants et de services ce fait naturellement (pour le programmeur) en ajoutant le dit composant au projet puis en complétant les *blocks* en rapport, alors qu'en programmation classique, il faut forcement passer par la documentation et les tests pour faire marcher des composants inconnus convenablement.

Cependant, plus les fonctionnalités d'une application sont nombreuses, plus les *blocks* seront nombreux sous AppInventor. En effet, alors que sous Eclipse le code est divisé en fichiers, classes et méthodes, sur AppInventor toutes les méthodes utilisées sont regroupées au même endroit. Quand les *blocks* deviennent massifs et nombreux, il est difficile de s'y retrouver, même en les organisant entre eux. De plus, même si nous avons la possibilité de commenter les *blocks*, cela rajoute du contenu graphique et de l'illisibilité.

[3] http://appinventor.googlelabs.com/learn/reference/components/other.html#ActivityStarter

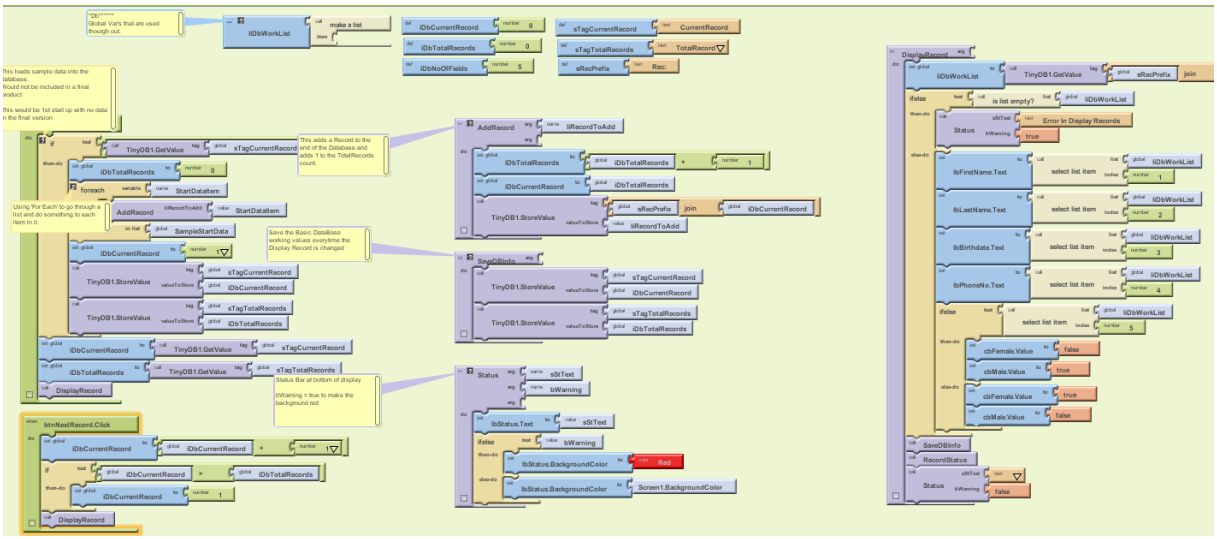

Figure 21 : Editeur de blocks commençant à être illisible

L'application X, qui comporte de nombreuses fonctionnalités, cumules les *blocks* et devient plus difficile à maintenir.

Si quelqu'un doit reprendre un projet avec des dizaines de *blocks* et comportant des dizaines de sous *blocks*, il devra lire quasiment l'intégralité des *blocks* pour comprendre le fonctionnement du programme.

AppInventor souffre aussi d'autres limites, plus techniques. En effet certains composants manquent à l'appel, comme la possibilité de faire appel à des librairies comme OpenGL, d'utiliser des services web (requêtes HTTP pour récupérer des XML, images ou fichiers par exemple) ou même le simple concept d'objets et d'instances est absent.

De plus, un projet est basé sur une *Activity* et ne permet pas d'en créer d'autre. Une *Activity* correspond, dans Android, à un écran d'application ; Il y a donc généralement une *Activity* pour l'accueil, une autre pour la fonctionnalité A, une troisième pour la fonctionnalité B, etc.

Techniquement **une application développée avec AppInventor est limitée à un seul écran**. Il existe des façons de contourner ce problème :

1. Nous pouvons créer plusieurs layouts dans AppInventor[4].
   Chacun correspond à un écran, lors de l'initialisation du programme, il faut tous les mettre en invisible à l'exception d'un. De la même façon, lorsqu'il faut changer d'écran, le layout affiché doit être passé en invisible et celui à afficher devient à son tour visible.
   Ce processus fonctionne mais évidement ce n'est pas très propre et cela complique énormément la programmation en *blocks*.
2. La deuxième solution est qu'il est possible de lancer une *Activity* d'une autre application AppInventor.
   Dans ce cas, il faut donc créer un projet par écran et avoir toutes les applications de ses projets installées sur le téléphone pour pouvoir lancer l'*Activity* d'une autre application qui une fois terminée rendra la main à celle qui l'a lancé.

[4] http://theunlockr.com/2010/09/01/google-appinventor-how-to-create-multiple-screens-for-your-app-as-best-we-can/

Cette solution n'est, elle non plus, pas propre, car les projets ne sont plus des applications bien distinctes mais des sous-projets d'un plus grand projet. De plus, pour que cela fonctionne, il faut que toutes les applications soient installées sur le téléphone.
Nous constatons alors qu'AppInventor n'est pas fait pour concevoir d'importantes applications.

Nous ne pouvons pas non plus utiliser AppInventor pour créer des morceaux de programme ou des interfaces graphiques à utiliser dans une application programmée plus classiquement. En effet, comme souligné dans la partie I, **il est impossible d'exporter un projet AppInventor sous forme de code Java**. Nous pouvons juste télécharger un fichier *.RAR* contenant le projet servant à le partager avec d'autres utilisateurs d'AppInventor. On remarque d'ailleurs que, contrairement à d'autre outils de travail en ligne créé par Google (*Google Document*, *Google Wave*, …), il n'est pas possible de collaborer et de partager le projet en ligne pour le travailler à plusieurs simultanément.

De plus, on remarque de façon plus générale qu'il n'est pas possible de créer des widgets ou des fonds d'écran animés avec AppInventor. Ce sont deux types particuliers d'application :

Figure 22 : Exemples de widget

Les widgets sont en quelques sortes des icônes à placer sur le « bureau » du téléphone et mis à jour (à un délai déterminé) pour afficher diverses informations ou permettre des fonctionnalités.

Les fonds d'écran animés sont, comme leur nom l'indique, des fonds d'écrans, qui sont en arrière-plan du bureau, et animés. Ils sont codés comme une application Android, ce qui leur permet d'utiliser des fonctionnalités du téléphone pour influer sur l'animation – afficher la carte en fonction de l'endroit où vous êtes géolocalisé, changer si on secoue le téléphone, etc.

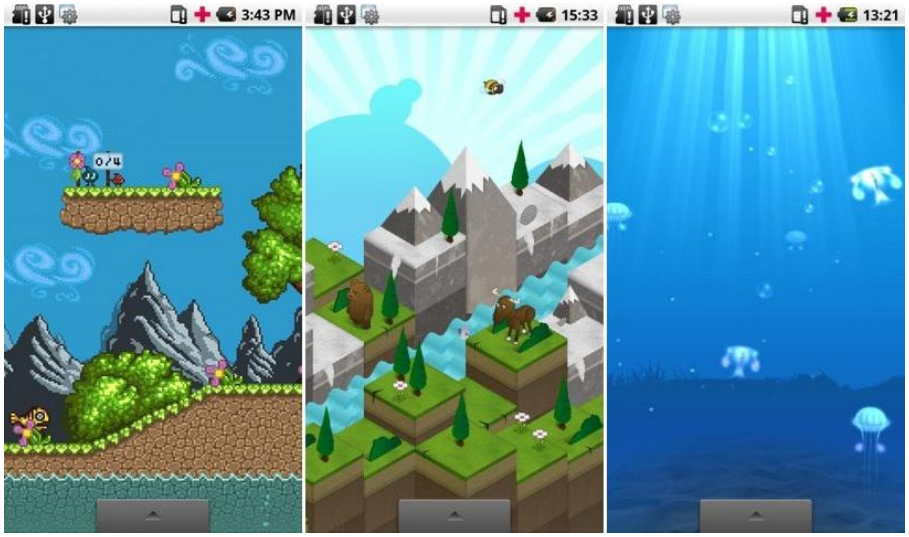

Figure 23 : Exemples de fond d'écran animé

Enfin, dernière limitation et non des moindres : il n'est pas possible de publier des applications créées avec AppInventor sur l'Android Market. Lorsque nous essayons, nous obtenons le message suivant :

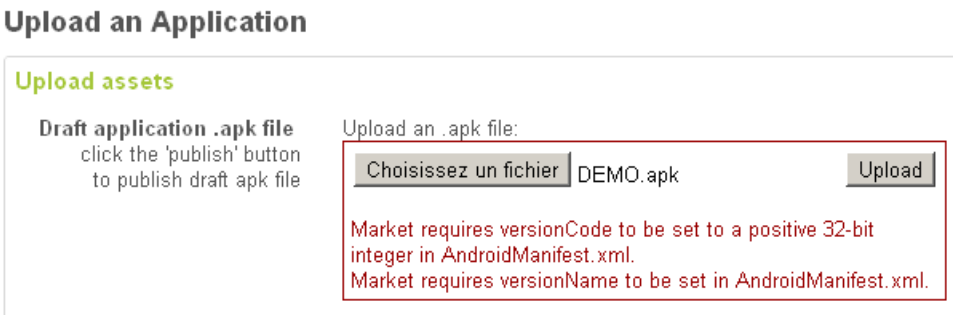

Figure 24 : Message d'erreur lors d'un envoi sur l'Android Market

Google a sans doute voulu ici, éviter que le Market soit pollué d'applications conçues « à la va vite » avec AppInventor. La publication sur d'autres Markets ou la distribution du *.APK* reste néanmoins possible.

Pour résumer, pour un programmeur Android classique, développer une application avec AppInventor n'a pas vraiment d'utilité. En effet AppInventor comporte énormément de limitations par rapport à un développement classique. L'outil peut lui servir à autre chose (maquettage), comme nous le verrons plus loin, mais il perd son utilité première d'outil de développement complet.

AppInventor est cependant très facilement utilisable, on retrouve les components que l'on connait et leurs méthodes. Pour un programmeur ne connaissant pas Android, il est tout aussi utilisable – car il s'abstrait de la programmation Java et se concentre sur la programmation logique par *blocks*, commune à la plupart des langages de programmation –, mais il devient également plus utile, par exemple pour découvrir rapidement une grande partie des fonctionnalités disponibles. Pour un programmeur, AppInventor est donc très utilisable et cela même si il est peu utile pour faire une application finalisée, on peut lui trouver d'autres usages comme nous le démontrerons plus loin.

B. Pour un novice

Nous avons vu que AppInventor était très limité pour un programmeur désireux de créer une « vraie application », mais qu'en est-il des non informaticiens qui veulent s'essayer à la programmation ? AppInventor a était conçu pour eux et nous allons voir dans cette partie si il est utilisable et utile pour un novice.

Nous avons trois cobayes qui vont essayer de réaliser deux petits exercices de programmation avec l'outil de Google. Les trois cobayes ont des profils forts différents :

1. Quentin – 21 ans – étudiant en DUT biologie – profil scientifique.
2. Emilie – 23 ans – diplômée d'une licence d'anglais – profil littéraire.
3. Lise – 8 ans – possède de bonnes notes en maths et en français.

Les exercices sont les suivants :

1. Réaliser une application comportant une zone de saisie de texte (*TextBox*), un bouton et une zone d'affichage de texte (*Label*, non éditable). Après avoir saisi un texte, si on clique sur le bouton, il faut que le texte s'affiche dans le label.

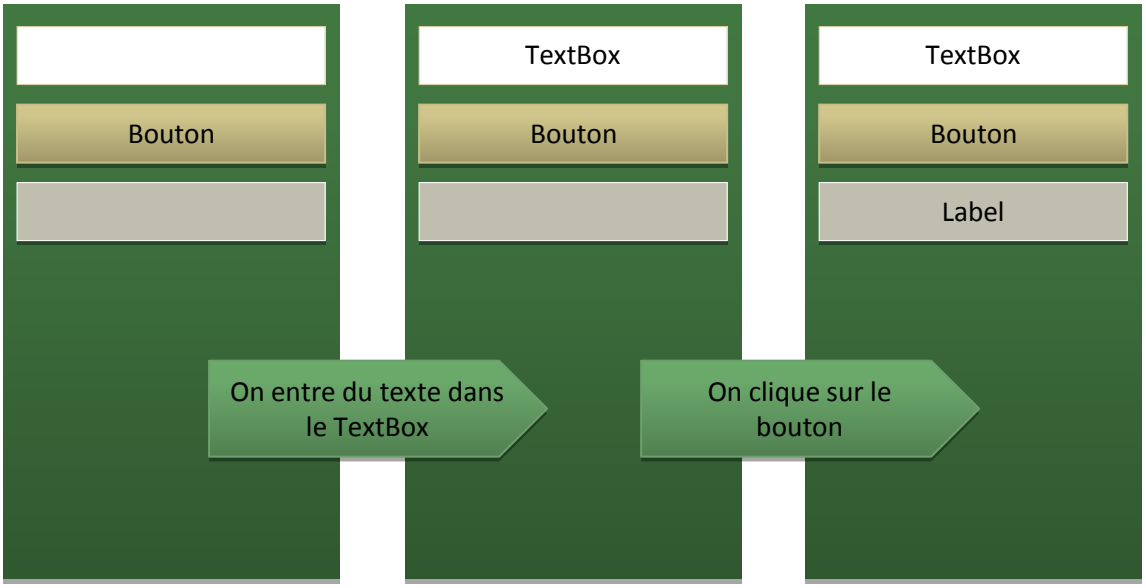

Figure 25 : Schéma de la première application test à réaliser

2. Réaliser une application qui affiche les plus grandes valeurs obtenues avec le capteur accéléromètrique.

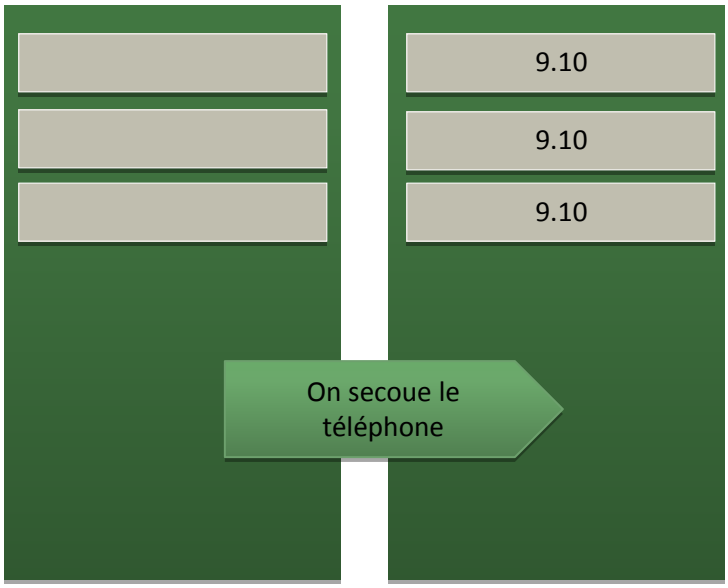

Figure 26 : Schéma de la seconde application test à réaliser

Dans l'ensemble, **aucun des cobayes n'a réussi à réaliser ces exercices seul**. Tout d'abord lors de la phase de design, la première contrainte est le vocabulaire, à la fois la langue et les termes utilisés. En effet, AppInventor est en anglais, ce qui pose problème aux non anglophones. Même pour les anglophones, certains termes ne sont pas familiers aux non informaticiens – par exemple : le terme « label » qui se traduit par « étiquette », représente une zone de texte non éditable.

Une fois les différents composants expliqués, la phase de design est généralement réussie. Le fait d'effectuer des Drap-and-Drop de composants puis de voir l'écran tracer un trait de positionnement en temps réel aide à la compréhension. Néanmoins, l'utilisation de composants non visibles n'est pas claire au début. Ici, le fait de placer un *AcceleratometerSensor* sur l'écran pour y avoir accès par la suite dans l'éditeur de *blocks* n'est pas évident à comprendre pour des non informaticiens qui ne sont pas habitués à faire des importations dans leur code.

Une fois l'interface désignée et tous les composants ajoutés, les choses se gâtent. Effectivement, bien que la programmation par *blocks* via l'interface Scratch soit beaucoup plus abordable que la programmation en classiques lignes de code, elle reste complexe à assimiler. Elle reste basée sur l'appel de méthode, le passage de paramètres, … cette logique n'est pas intelligible pour les non informaticiens. Ils comprennent le fonctionnement de code assemblé quand on leur explique mais ont beaucoup de mal à en composer par eux même. Trouver les bons *blocks* et savoir comment les utiliser leur prend du temps et encore une fois la langue n'aide pas.

La première application qui consiste à transférer le texte du *TextBox* vers un label lorsque l'on clique sur un bouton a été réussie (avec aide) par tous les cobayes, à l'exception de Lise. On ne peut pas faire d'un cas d'étude une généralité. Mais en essayant de faire comprendre le fonctionnement d'AppInventor à cette enfant de 8ans, nous nous sommes aperçus qu'un grand nombre d'obstacles se dressaient face à elle – la langue dans un premier temps ainsi que le concept des composants. Ils sont difficilement surmontables pour quelqu'un n'utilisant pas de téléphone/Smartphone. S'agissant des autres testeurs, la conception de la première application a aussi nécessité notre aide : pour expliquer le rôle et le fonctionnement des composants, les méthodes à utiliser dans la partie éditeur de *blocks* et, plus globalement, les étapes à suivre pour faire fonctionner AppInventor.

La deuxième application nécessite plus de logique. Elle utilise des variables et des conditions. Ici aussi ces concepts logiques propres aux informaticiens ne sont pas connus de tous. Il a donc fallut expliquer le système de `IF ELSE THEN`, le type booléen, etc.

Pour conclure, on peut dire qu'**AppInventor est inutilisable sans aide par des non informaticiens** ; ceci à cause de la langue, des termes techniques et des logiques propres à la programmation. Cependant avec l'aide de quelqu'un, cela permet d'**avoir une première approche de cette logique** ainsi que d'avoir un résultat immédiat de son travail puisqu'on peut rapidement tester sur son téléphone et créer des applications assez intéressantes (utilisation de TextToSpeech, GPS,…).

C. Quels usages pour AppInventor ?

Nous avons vu dans les parties précédentes qu'AppInventor était utilisable mais peu utile pour les informaticiens et à l'inverse utile mais peu utilisable par les non informaticiens. On peut donc se demander quels sont les usages d'AppInventor ? En fait, il y en a quelques-uns, pour les différents types d'utilisateur.

Tout d'abord pour les informaticiens, même si l'outil est limité en matière de développement d'applications entièrement fonctionnelles, il peut être utilisé pour créer rapidement des maquettes d'applications en partie fonctionnelles. En effet, l'interface visuelle peut être employée pour designer des applications : il est rapide à prendre en main, facilement modifiable ; on peut établir des comportements grâce aux *blocks* et enfin on peut tester et exporter rapidement sur le téléphone. Cela peut donc permettre de **créer rapidement une maquette**, pour tester des concepts ou présenter une idée à ses collaborateurs.

Pour les informaticiens, non initiés à Android, AppInventor donne l'occasion de découvrir les composants. Ils peuvent alors **s'en servir pour créer rapidement de petites applications** correspondant à un besoin facilement réalisable avec AppInventor, donc relativement limité en fonctionnalités.

Concernant les **non informaticiens, Appinventor se trouve être difficile à utiliser**. Il peuvent tout de même s'en sortir s'ils comprennent l'anglais et qu'ils suivent à la lettre les tutoriaux disponibles[5].

Pour finir, **avec l'aide d'un professeur**, cela permet de découvrir les bases de la programmation, ou tout du moins ses logiques. La programmation par *blocks* ressemble beaucoup à la programmation « en pseudo-code » qui peut être vue lors de premières années d'études informatiques. Utiliser AppInventor offre en prime la possibilité d'apprendre les logiques de la programmation, de mettre en place facilement une interface graphique et surtout de pouvoir tester soit même son code avec un téléphone.

[5]http://appinventor.googlelabs.com/learn/tutorials/index.html

Conclusion

Ce projet nous a permis d'étudier AppInventor de plus près. Nous avons réalisé **trois applications, plus ou moins complexes**, fait tester cette plate-forme à des néophytes pour en évaluer l'utilisabilité, et enfin nous avons recherché des usages à ce site.

Les avantages de cette solution sont nombreux. Comme nous avons pu l'observer tout au long de ce rapport, **AppInventor est extrêmement bien réalisé**. Il existe peu de produits comparables pour Android. Il renferme beaucoup de fonctionnalités au travers de ces nombreux composants, se trouve être bien documenté, est totalement gratuit et de surcroit possède un forum assez actif. Toutefois, **il comporte aussi des limitations et inconvénients notables**. Effectivement,même si de nombreuses fonctionnalités sont présentes, elles n'y sont pas toutes. De plus, nous ne pouvons pas réaliser n'importe quels types de projet Android (pas de widgets ou de fonds d'écran) et nous sommes limités à un seul écran par projet.

Plus généralement, AppInventor est uniquement en anglais, ce qui peut s'avérer être un problème. En effet, les principales cibles sont les élèves de collège et lycée. Ceux-ci risquent d'avoir du mal à s'en sortir s'ils ne sont pas anglophones. Enfin, cet outil ne permet pas de publier les applications créées sur l'Android Market, alors qu'il est tout de même possible de la proposer sur d'autres *markets*, non officiels. Malheureusement l'Android Market est le plus visité.

De notre point de vue, **AppInventor est donc un outil parfaitement conçu mais qui manque d'utilité**. Il reste trop compliqué pour les non informaticiens et trop simple pour les informaticiens si bien qu'il est difficile de savoir à qui il pourrait réellement être à la fois utile et utilisable. Néanmoins il y a quand même certains usages où il peut s'avérer fort utile, comme nous avons pu le voir plus précédemment, notamment dans le design et le maquettage d'applications et l'apprentissage des bases de l'informatique.

En outre, cet outil est jeune et acquiert régulièrement des mises à jour de la part de Google. À notre avis, pour le futur, il serait intéressant qu'il s'améliore en deux points. D'une part, la traduction du site simplifierait la prise en main pour les jeunes et pour les non anglophones. D'autre part, **rendre AppInventor un peu plus utile pour les développeurs**. Par exemple, la possibilité d'**ajouter ou d'exporter du code Java et permettre le travail collaboratif instantané** afin d'accélérer le développement d'application serait un axe de développement intéressant.

Annexes

Annexe 1 : Serveur, fichier db.sql 23
Annexe 2 : Serveur, fichier connexion.php 24
Annexe 3 : Serveur, fichier _conexim p.inc 25
Annexe 4 : Serveur, fichier glihm/index.php 26
Annexe 5 : Serveur, fichier glihm/send.php 30
Annexe 6 : Serveur, fichier glihm/get.php 31
Annexe 7 : Serveur, fichier glihm/form.php 33
Annexe 8 : Serveur, fichier glihm/_exit.php 34
Annexe 9 : Appication Java, ficher ActivityEnvoi.java 35
Annexe 10 : Appication Java, ficher ActivityRecois.java 38
Annexe 11 : Appication Java, ficher CommunicationServeur.java 40
Annexe 12 : Appication Java, ficher Main.java 44
Annexe 13 : Mail de confirmation de Google 45

Annexe 1 : Serveur, fichier db.sql

```
CREATE TABLE `glihm_message` (
`idmessage` int(11) NOT NULL auto_increment,
  `sender` varchar(45) NOT NULL,
  `content` text NOT NULL,
  `geoloc_lat` varchar(45) NOT NULL,
  `geoloc_long` varchar(45) NOT NULL,
  `date` timestamp NOT NULL default CURRENT_TIMESTAMP on update
CURRENT_TIMESTAMP,
PRIMARY KEY (`idmessage`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```


Annexe 2 : Serveur, fichier connexion.php

```
<?php
// Paramètres persos
$host = ""; // voir hébergeur
$user = ""; // vide ou "root" en local
$pass = ""; // vide en local
$bdd = ""; // nom de la BD
// connexion
$link = mysql_connect($host,$user,$pass)
or die( include("_coneximp.inc") );
@mysql_select_db("$bdd")
or die( include("_coneximp.inc") );
?>
```


Annexe 3 : Serveur, fichier _coneximp.inc

```
<h1>Erreur</h1>
            <p>Serveur en maintenance pour le moment.</p>
            <p>Vous être prié de revenir plus tard&#1333; merci!</p>
```


Annexe 4 : Serveur, fichier glihm/index.php

```
<?php

include('../connexion.php');

// renvoi la distance en mètres
functionget_distance_m($lat1, $lng1, $lat2, $lng2) {
  $earth_radius = 6378137;   // Terre = sphère de 6378km de rayon
$rlo1 = deg2rad($lng1);
  $rla1 = deg2rad($lat1);
  $rlo2 = deg2rad($lng2);
  $rla2 = deg2rad($lat2);
  $dlo = ($rlo2 - $rlo1) / 2;
  $dla = ($rla2 - $rla1) / 2;
  $a = (sin($dla) * sin($dla)) + cos($rla1) * cos($rla2) * (sin($dlo) *
sin($dlo));
  $d = 2 * atan2(sqrt($a), sqrt(1 - $a));
return ($earth_radius * $d);
}

//echo (round(get_distance_m(48.856667, 2.350987, 45.767299, 4.834329) /
1000, 3))." km"; // affiche 391.613 km

?>

<h1>Projet de GLIHM</h1>

<div style="width: 800px; margin: 0px auto; background-color: #EEEEEE;">
      <div   style="width:   100px;   background-color:   #DDDDDD;   float:
left;">ID</div>
      <div   style="width:   100px;   background-color:   #DDDDDD;   float:
left;">sender</div>
      <div   style="width:   250px;   background-color:   #DDDDDD;   float:
left;">content</div>
      <div   style="width:   100px;   background-color:   #DDDDDD;   float:
left;">lat</div>
      <div   style="width:   100px;   background-color:   #DDDDDD;   float:
left;">long</div>
      <div   style="width:   150px;   background-color:   #DDDDDD;   float:
left;">date</div>
      <div style="clear: both;"></div>
<?php
$query = "SELECT * FROM glihm_message ORDER BY idmessage ASC;";
$result = mysql_query($query);
while ( $var = mysql_fetch_array($result) )
{
echo "      <div           style=\"width:           100px;           float:
left;\">".$var[0]." </div>
      <div style=\"width: 100px; float: left;\">".$var[1]." </div>
      <div style=\"width: 250px; float: left;\">".$var[2]." </div>
      <div style=\"width: 100px; float: left;\">".$var[3]." </div>
      <div style=\"width: 100px; float: left;\">".$var[4]." </div>
```


```
      <div style=\"width: 150px; float: left;\">".$var[5]." </div>
      <div style=\"clear: both;\"></div>\n";
}

?>
</div>

<h2>Géolocalisation</h2>

<?php
$rayon = 50;
$lat = 2.00524;
$long = 0.99568;
if (isset($_GET['long'])) {
      $long = $_GET['long']+0;
}
if (isset($_GET['lat'])) {
      $lat = $_GET['lat']+0;
}
if (isset($_GET['rayon'])) {
      if ( $_GET['rayon'] > 10000 ) {
            $rayon = 10000;
      } else if ( $_GET['rayon'] < 0 ) {
            $rayon = 100;
      } else {
            $rayon = $_GET['rayon'];
      }
}
?>

<div style="width: 800px; margin: 0px auto; padding-top: 15px; padding-
bottom: 15px; background-color: #EEEEEE;">
<form method="get">
<div style="text-align: center;">
      <div style="margin: 0px auto; text-align: left; width: 300px;">
      <p>Rayon : <select name="rayon" style="float: right; text-align:
right;">
      <option value="1">1 m</option>
      <option value="2">2 m</option>
      <option value="5">5 m</option>
      <option value="10">10 m</option>
      <option value="20">20 m</option>
      <option value="50">50 m</option>
      <option value="100">100 m</option>
      <option value="200">200 m</option>
      <option value="500">500 m</option>
      <option value="1000">1 km</option>
      <option value="2000">2 km</option>
      <option value="5000">5 km</option>
      <option value="10000">10 km</option>
      </select></p>
      <p>Latitude : <input type="text" name="lat" style="float: right;"
value="<?php echo $lat; ?>"></p>
```


```
      <p>Longitutde : <input type="text" name="long" style="float: right;"
value="<?php echo $long; ?>"></p>
      </div>
<input type="submit" value="Chercher">
</div>
</form>
</div>

<h3>Résultat :</h3>

<div style="width: 800px; margin: 0px auto; background-color: #EEEEEE;">
      <div  style="width:  800px;  margin:  0px  auto;  background-color:
#FFFFFF; text-align: center;">
      Rayon de cherche : <?phpecho $rayon; ?>m.
      </div>
      <div style="clear: both;"></div>
      <div   style="width:   50px;   background-color:   #DDDDDD;   float:
left;">ID</div>
      <div   style="width:   100px;   background-color:   #DDDDDD;   float:
left;">sender</div>
      <div   style="width:   150px;   background-color:   #DDDDDD;   float:
left;">content</div>
      <div   style="width:   100px;   background-color:   #DDDDDD;   float:
left;">lat</div>
      <div   style="width:   100px;   background-color:   #DDDDDD;   float:
left;">long</div>
      <div   style="width:   150px;   background-color:   #DDDDDD;   float:
left;">date</div>
      <div   style="width:   150px;   background-color:   #DDDDDD;   float:
left;">distance</div>
      <div style="clear: both;"></div>
<?php
$query = "SELECT * FROM glihm_message ORDER BY idmessage ASC;";
$result = mysql_query($query);
while ( $var = mysql_fetch_array($result) )
{
$dist = get_distance_m($lat, $long, $var[3], $var[4]);
$color = "";
if ($dist<= $rayon) { $color = " background-color: #AAFFAA;"; }
echo "      <div            style=\"width:            50px;            float:
left;".$color."\">".$var[0]." </div>
      <div              style=\"width:              100px;              float:
left;".$color."\">".$var[1]." </div>
      <div              style=\"width:              150px;              float:
left;".$color."\">".$var[2]." </div>
      <div              style=\"width:              100px;              float:
left;".$color."\">".$var[3]." </div>
      <div              style=\"width:              100px;              float:
left;".$color."\">".$var[4]." </div>
      <div              style=\"width:              150px;              float:
left;".$color."\">".$var[5]." </div>
      <div              style=\"width:              150px;              float:
left;".$color."\">".get_distance_m($lat, $long, $var[3], $var[4])." m</div>
```


```
      <div style=\"clear: both;\"></div>\n";
}
echo "</div>";
?>

<?php

include('_exit.php');

?>
```


Annexe 5 : Serveur, fichier glihm/send.php

```
<?php

include('../connexion.php');

$nbVar = 4;
if ( isset($_POST['sender']) ) {
      $nbVar--;
}
if ( isset($_POST['content']) ) {
      $nbVar--;
}
if ( isset($_POST['lat']) ) {
      $nbVar--;
}
if ( isset($_POST['long']) ) {
      $nbVar--;
}

if ( $nbVar> 0 ) {
      // Si tous les paramètres requis ne sont pas présents, on retourne
"nok".
      echo "nok";
} else {
      // Si tous les paramètres requis ne sont pas présents, on retourne
"ok" et on ajoute le message dans la base de données.
      echo "ok";
      $query = "INSERT INTO `glihm_message` (`idmessage` , `sender` ,
`content` , `geoloc_lat` , `geoloc_long`) VALUES ('',
'".$_POST['sender']."', '".$_POST['content']."', '".$_POST['lat']."',
'".$_POST['long']."');";
      $result = mysql_query($query);
}

include('_exit.php');

?>
```


Annexe 6 : Serveur, fichier glihm/get.php

```
<?php

include('../_conex.php');
//include('../connexion.php');

// Fonction calculant la distance entre deux points géolocalisés.
function get_distance_m($lat1, $lng1, $lat2, $lng2) {
  $earth_radius = 6378137;   // Terre = sphère de 6378km de rayon
  $rlo1 = deg2rad($lng1);
  $rla1 = deg2rad($lat1);
  $rlo2 = deg2rad($lng2);
  $rla2 = deg2rad($lat2);
  $dlo = ($rlo2 - $rlo1) / 2;
  $dla = ($rla2 - $rla1) / 2;
  $a = (sin($dla) * sin($dla)) + cos($rla1) * cos($rla2) * (sin($dlo) *
sin($dlo));
  $d = 2 * atan2(sqrt($a), sqrt(1 - $a));
  return ($earth_radius * $d);
}

//Valeurs par défaut
$nbVar = 2;
$rayon = 1000;
$lat = 2.00524;
$long = 0.99568;
if ( isset($_POST['lat']) ) {
      $lat = $_POST['lat']+0;
      $nbVar--;
}
if ( isset($_POST['long']) ) {
      $long = $_POST['long']+0;
      $nbVar--;
}
if (isset($_POST['rayon'])) {
      if ( $_POST['rayon'] > 1000 ) {
            $rayon = 1000;
      } else if ( $_POST['rayon'] < 0 ) {
            $rayon = 100;
      } else {
            $rayon = $_POST['rayon'];
      }
}

if ( $nbVar > 0 ) {
      // Si tous les paramètres requis ne sont pas présents, on retourne
"_nok".
      echo "_nok";
} else {
      $haveRes = false;
      $result = "";
      $marqueur = ";";
```


```
	// Listage des résultats.
	$query = "SELECT * FROM `glihm_message` WHERE date >
DATE_SUB(CURDATE(),INTERVAL 8 DAY) ORDER BY idmessage ASC;";
	$resultQuery = mysql_query($query);
	while ( $var = mysql_fetch_array($resultQuery) )
	{
		$dist = get_distance_m($lat, $long, $var[3], $var[4]);
		if ( $dist <= $rayon ) {
			$haveRes = true;
			//date;qui;dist;mess;
			$result = $result.$var[5].$marqueur;
			$result = $result.$var[1].$marqueur;
			$result = $result.$dist.$marqueur;
			$result = $result.$var[2].$marqueur;
		}
	}
	echo $test;
	if ( $haveRes == true ) {
		// Si il y a des résultats, les retournes.
		echo $result;
	} else {
		// Si il n'y a aucun résultat, on retourne "_nnok".
		echo "_nnok";
	}
}

include('_exit.php');

?>
```


Annexe 7 : Serveur, fichier glihm/form.php

```
<div style="width: 500px; float: left;">
<form method="post" action="./send.php">
<label>
<input name="sender" type="text" id="sender" value="0606060606" />
</label>
<p>
<label>
<textareaname="content" id="content" cols="45" rows="5">Voila le texte que
j'envoie maintenant.</textarea>
</label>
</p>
<p>
<label>
<input name="lat" type="text" id="lat" value="36.55487" />
</label>
</p>
<p>
<label>
<input name="long" type="text" id="long" value="+150.4578893" />
</label>
</p>
<p>
<input type="submit" value="Envoyer" />
</p>
</form>
</div>

<div style="width: 500px; float: left;">
<form method="post" action="./get.php">
<p>
<label>
<input name="lat" type="text" id="lat" value="36.55487" />
</label>
</p>
<p>
<label>
<input name="long" type="text" id="long" value="+150.4578893" />
</label>
</p>
<p>
<input type="submit" value="Envoyer" />
</p>
</form>
</div>
```


Annexe 8 : Serveur, fichier glihm/_exit.php

```
<?php

mysql_close($link);

?>
```


Annexe 9 : Appication Java, ficher ActivityEnvoi.java

```
package com.app.e.service.APP_E_SERVICE_COMM_SERV;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;

/**
 * Activité qui sert à envoyer les données au serveur.
 * Envoie un message localisé au serveur.
 *
 *  A son lancement elle récupère les infos passés en paramètres
(l'expediteur du message, le message et les coordonnées satélite).
 *  Puis elle envoie ces informations au serveur et rend la main à
l'activité qui l'a lancée.
 *
 * @author Administrateur
 *
 */
public class ActivityEnvoi extends Activity {

	/**
	 * Réponse que l'activityEnvoi vas retourner.
	 */
	private String reponse;
	/**
	 * Numéro de Téléphone de l'expéditeur du message.
	 */
	private String qui;
	/**
	 * Message.
	 */
	private String quoi;
	/**
	 * Longitude.
	 */
	private String longitude;
	/**
	 * Latitude.
	 */
	private String latitude;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //  On  récupère  les  informations  transmisent  au  serveur  par
l'intermédiaire du Bundle.
	String[]                              all                              =
this.getIntent().getExtras().getString("all").split(";");
        boolean success=true;
        reponse = "Bien envoyé !";
        // Initialisation des variables.
```


```
        qui = "0";
        quoi = "";
        longitude = "0.0";
        latitude = "0.0";
        // On met à jour les variables.
        try{
            qui = all[0];
            longitude = all[1];
            latitude = all[2];
            quoi = all[3];
            for (int i = 4 ; i < all.length ; i++ )
            {
                  quoi += ";"+all[i];
            }
        } catch(Exception e) {
            success=false;
        }


        if(success)
        {
            // Si les variables ont bien été mise à jour on envoie les
données au serveur.
            sendtoserveur();
        }else{
            reponse = "Erreur lors du lancement de l'application.\nAvez-
vous bien activité votre GPS ?";
        }

        // Une fois fini l'activité rend la main en transmettant la réponse
au serveur.
        Intent intent = new Intent();
        Bundle bundle = new Bundle();
        bundle.putString("activity1", reponse);
        intent.putExtras(bundle);
        setResult(RESULT_OK, intent);
        finish();

    }

    /**
     * Envoie le message au serveur.
     *     On     utilise     comme     l'adresse          pour     l'envoie
CommunicationServeur.adresseServeurSend  qui  est  l'adresse  de  base  du
serveur.
     */
    private void sendtoserveur()
    {
      try {
            //    On    utilise    comme    l'adresse        pour    l'envoie
CommunicationServeur.adresseServeurSend  qui  est  l'adresse  de  base  du
serveur.
```


```
                CommunicationServeur.SetPosition(latitude,      longitude,
quoi, qui, CommunicationServeur.adresseServeurSend);
            } catch(Exception e) {
                reponse = "Echec de connexion au serveur";
            }
    }
}
```


Annexe 10 : Appication Java, ficher ActivityRecois.java

```
package com.app.e.service.APP_E_SERVICE_COMM_SERV;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.widget.Toast;

/**
 * Activité qui sert à récupérer les informations sur le serveur.
 * on lui donne des coordonnées satélite et elle retourne les éventuelles messages qui ont été postés à cet endroit.
 *
 * A son lancement elle récupère les infos passés en paramètres (les coordonnées satélite).
 * Puis elle envoie ces informations au serveur, attend le retour de celui ci, enfin elle rend la main à l'activité qui l'a lancée (en lui retournant les informations récupérées).
 * @author Administrateur
 *
 */
public class ActivityRecois extends Activity {

	/**
	 * Réponse que l'activityEnvoi vas retourner.
	 * (les messages proches ou un message d'erreur).
	 */
	private String reponse;
	/**
	 * Longitude.
	 */
	private String longitude = "0";
	/**
	 * Latitude.
	 */
	private String latitude = "0";
	/***
	 * Si estlocalise est vrai, alors la longitude et la latitude ont été correctement initialisées.
	 */
	private boolean estlocalise = false;

    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
	      // Récupére les arguments placés en paramètre.
      Toast.makeText(this.getApplicationContext(),       "Refresh       !", Toast.LENGTH_SHORT).show();
      String[]                              all                                  = this.getIntent().getExtras().getString("all").split(";");
	      try {
	            longitude = all[0];
```


```
                latitude = all[1];
                estlocalise = true;
            } catch(Exception e) {
                estlocalise=false;
            }
            // Tente de récuperer les messages proches sur le serveur.
            refreshlist();

            //TODO à virer
      //Toast.makeText(this.getApplicationContext(),     ">>     "+reponse,
Toast.LENGTH_SHORT).show();

      // Une fois fini l'activité rend la main en transmettant la réponse
au serveur.
      Intent intent = new Intent();
        Bundle bundle = new Bundle();
        bundle.putString("activity2", reponse);
        intent.putExtras(bundle);
        setResult(RESULT_OK, intent);
        finish();

    }

    /**
     * Met à jour le message à renvoyer (les messages proches ou un message
d'erreur)
     *
     *    On    utilise    comme    l'adresse    serveutr    pour    l'envoie
CommunicationServeur.adresseServeurGet  qui  est  l'adresse  de  base  du
serveur.
     */
    public void refreshlist()
    {
      String myreponse = "_Il n'y a aucun message ici. Chercher ailleur.";
      if( !estlocalise )
      {
            myreponse = "_Vous n'êtes pas géolocalisé.";
          } else {
            try{
                  //   On   utilise   comme   l'adresse     pour   l'envoie
CommunicationServeur.adresseServeurGet  qui  est  l'adresse  de  base  du
serveur.
                  myreponse    =    CommunicationServeur.Refresh(longitude,
latitude,CommunicationServeur.adresseServeurGet);
                  } catch(Exception e) {
                        myreponse = "_Echec de la connection au serveur.";
                  }
                  reponse = myreponse;
      }
    }
}
```


Annexe 11 : Appication Java, ficher CommunicationServeur.java

```
package com.app.e.service.APP_E_SERVICE_COMM_SERV;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.List;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import org.apache.http.HttpResponse;

/**
 * Cette class permet de gérer les communications avec un serveur distant.
 * Les méthodes de cette class sont indépendantes et static, ell ne posède
donc pas de constructeur.
 * @author Administrateur
 *
 */
public class CommunicationServeur {


	/**
	 * Adresse de base du serveur, page où l'on envoie les données.
	 */
	public static String
adresseServeurSend="http://glihm.nesscorp.fr/send.php";
	/**
	 * Adresse de base du serveur, page où l'on recois les données.
	 */
	public static String
adresseServeurGet="http://glihm.nesscorp.fr/get.php";



	/**
	 * Méthode qui prend en paramètre des coordonées satélite et une
adresse.
	 * Envoie ces coordonnées au serveur avec url type :
	 * ADRESSE_SERVEUR/?lat=VALEUR_LATITUDE&long=VALEUR_LONGITUDE
	 * exemple :
	 * http://glihm.nesscorp.fr/get.php/?lat=2.00524&long=0.99568
	 *
	 * et retourne le résultat renvoyé par ce serveur.
	 */ public static String Refresh(String latitude, String
longitude,String adr)
	{
```


```
            // Initialisation de la variable de retour.
            String s="rien connection";
            HttpPost httppost = new HttpPost(adr);

            // On crée la liste qui contiendra tous nos paramètres.
            List<NameValuePair>         nameValuePairs         =         new
ArrayList<NameValuePair>();
            // Et on y rajoute nos paramétres.
            nameValuePairs.add(new BasicNameValuePair("lat", latitude));
            nameValuePairs.add(new BasicNameValuePair("long", longitude));

            try {
                  httppost.setEntity(new
UrlEncodedFormEntity(nameValuePairs));
                  HttpClient httpclient = new DefaultHttpClient();
                  // On envoi la requette http et on récupère la réponse.
                  HttpResponse response=httpclient.execute(httppost);
                  BufferedReader    reader    =    new    BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
                  String tmp=reader.readLine() ;
                  s="";
                  // On met la réponse buffer sous forme de String.
                  while(tmp!=null)
                  {
                        s = s +tmp+ "\n";
                        tmp=reader.readLine() ;
                  }
                  s.substring(0, s.length()-1);
            } catch (UnsupportedEncodingException e) {
                  //  En  cas  d'erreur  on  retourne  le  message  d'erreur
correspondant.
                  e.printStackTrace();
                  s=e.getMessage();
            } catch (ClientProtocolException e) {
                  //  En  cas  d'erreur  on  retourne  le  message  d'erreur
correspondant.
                  e.printStackTrace();
                  s=e.getMessage();
            } catch (IOException e) {
                  //  En  cas  d'erreur  on  retourne  le  message  d'erreur
correspondant;
                  e.printStackTrace();
                  s=e.getMessage();
            }
            return s;
      }

      /**
       *  Méthode  qui  prend  en  paramètre  des  coordonées  satélite,  un
message, un expéditeur et une adresse.
       * Envoie ces paramètres au serveur avec url type :
```


```
       *
ADRESSE_SERVEUR/?sender=EXPEDITEUR&lat=VALEUR_LATITUDE&long=VALEUR_LONGITUD
E&content=MESSAGE
       * exemple :
       *
http://glihm.nesscorp.fr/send.php/?sender=2.00524&lat=0.99568&long=0.88880&
content='testmessage'
       *
       * et retourne le résultat renvoyé par ce serveur.
       * @param latitude
       * @param longitude
       * @param message
       * @param qui
       * @param adr
       * @return
       */
      public     static     String     SetPosition(String     latitude,String
longitude,String message, String qui, String adr)
      {
            String s = "Aucune connexion";
            HttpPost httppost = new HttpPost(adr);
            // On crée la liste qui contiendra tous nos paramètres.
            List<NameValuePair>         nameValuePairs         =         new
ArrayList<NameValuePair>();

            // Et on y rajoute nos paramétres.
            nameValuePairs.add(new BasicNameValuePair("sender", qui));
            nameValuePairs.add(new BasicNameValuePair("lat", latitude));
            nameValuePairs.add(new BasicNameValuePair("long", longitude));
            nameValuePairs.add(new BasicNameValuePair("content", message));
            try {
                  httppost.setEntity(new
UrlEncodedFormEntity(nameValuePairs));
                  HttpClient httpclient = new DefaultHttpClient();
                  // On envoie la requette http et on récupère la réponse.
                  HttpResponse response=httpclient.execute(httppost);
                  BufferedReader    reader    =    new    BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
                  s="";
                  // On met la réponse buffer sous forme de String.
                  String tmp=reader.readLine() ;
                  while(tmp!=null)
                  {
                        s = s +tmp+ "\n";
                        tmp=reader.readLine() ;
                  }
            } catch (UnsupportedEncodingException e) {
                  //  En  cas  d'erreur  on  retourne  le  message  d'erreur
correspondant.
                  e.printStackTrace();
                  s=e.getMessage();
            } catch (ClientProtocolException e) {
```


```
                  //  En  cas  d'erreur  on  retourne  le  message  d'erreur
correspondant;
                  e.printStackTrace();
                  s=e.getMessage();
            } catch (IOException e) {
                  //  En  cas  d'erreur  on  retourne  le  message  d'erreur
correspondant.
                  e.printStackTrace();
                  s=e.getMessage();
            }
            return s;
      }
}
```


Annexe 12 : Appication Java, ficher Main.java

```
package com.app.e.service.APP_E_SERVICE_COMM_SERV;

import android.app.Activity;
import android.os.Bundle;

/***
 * Class Main vide
 * @author Administrateur
 */
public class Main extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
    }

}
```


Annexe 13 : Mail de confirmation de Google

From: **App Inventor Team** <noreplyappinventor@google.com>
Date: 2010/7/30
Subject: Getting Started with App Inventor

Welcome to App Inventor for Android!

The Google account that you are receiving this email on has been given access to App Inventor.

We recommend you start your app building adventures by working through theGetting Started material. You might also want to read moreabout App Inventor and take a look at somesample apps. Finally, you can ask questions and get help by signing up for theApp Inventor Google Group.

Thanks!
Google's App Inventor Team