

# Student Invaders



## Table des matières

1	Analyse préliminaire .....	3
1.1	Introduction .....	3
1.2	Objectifs .....	3
1.3	Planification initiale .....	5
2	Analyse / Conception .....	5
2.1	Concept.....	5
2.1.1	Vue d'ensemble.....	5
2.1.2	MCD .....	6
2.2	Stratégie de test .....	7
2.3	Risques techniques .....	7
2.4	Planification.....	8
2.5	Dossier de conception .....	8
2.5.1	Maquettes / Use cases / Scénarios.....	9
2.5.2	MLD.....	17
2.5.3	Particularité 1 – Les blocs pour les mots du professeur .....	18
2.5.4	Particularité 2 – Le changement de scènes .....	20
3	Réalisation .....	22
3.1	Dossier de réalisation .....	22
3.2	Description des tests effectués .....	23
3.3	Erreurs restantes .....	25
3.4	Liste des documents fournis .....	25
4	Conclusions.....	26
5	Annexes .....	27
5.1	Résumé du rapport du TPI / version succincte de la documentation.....	27
5.2	Glossaire .....	28
5.3	Sources – Bibliographie.....	28
5.4	Journal de bord .....	28
5.5	Manuel d'Installation .....	29
5.6	Manuel d'Utilisation .....	29
5.7	Archives du projet.....	29

### NOTE L'INTENTION DES UTILISATEURS DE CE CANEVAS:

*Toutes les parties en italiques sont là pour aider à comprendre ce qu'il faut mettre dans cette partie du document. Elles n'ont donc aucune raison d'être dans le document final.*

*De plus, en fonction du type de projet, il est tout à fait possible que certains chapitres ou paragraphes n'aient aucun sens. Dans ce cas il est recommandé de les retirer du document pour éviter de l'alourdir inutilement.*

# **1 Analyse préliminaire**

## **1.1 Introduction**

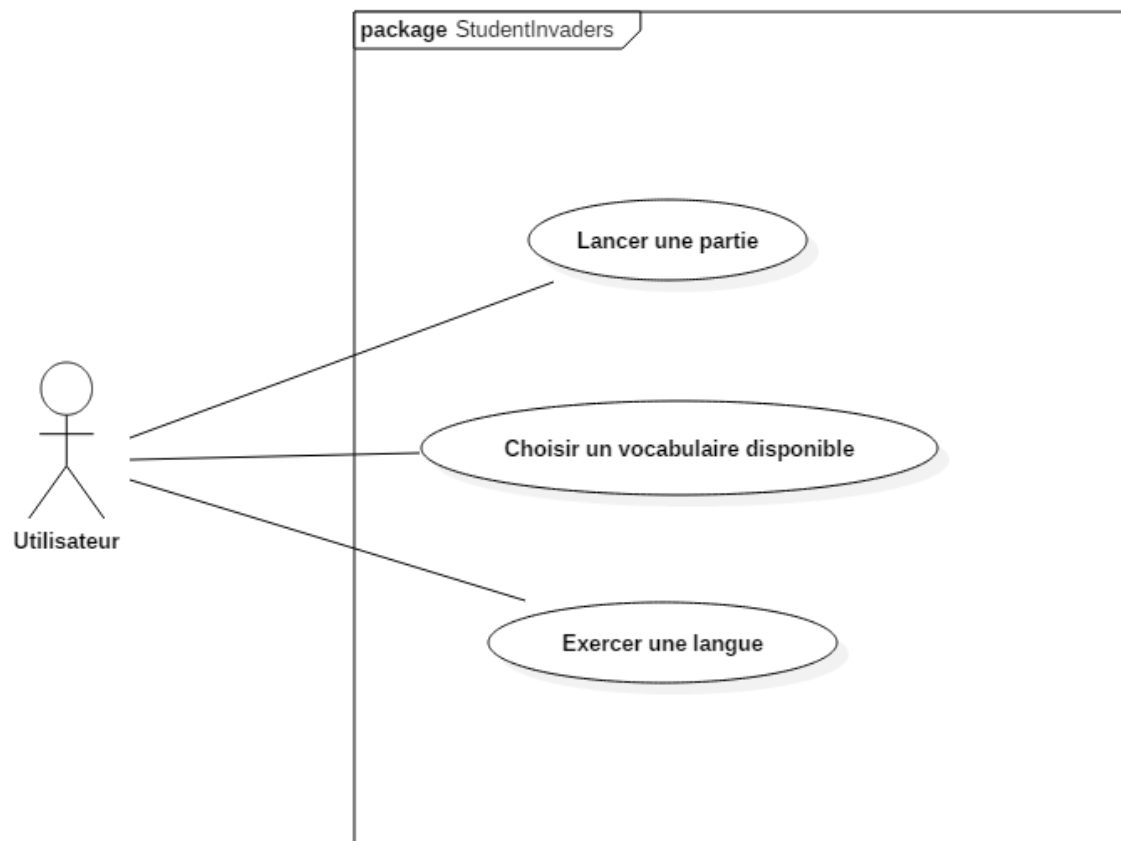
Student Invaders est un jeu didactique d'apprentissage de vocabulaire sur la plateforme Android d'une manière d'un Space Invaders. Dans cette application, le joueur contrôle le professeur. Il a pour but d'envoyer les mots traduits aux élèves avec un avion en papier. Ce projet est une manière assez intuitive d'apprendre et d'exercer une langue. Personnellement, j'apprendrai plus du côté de la programmation mais le jeu peut apporter un côté interactif aux cours. Ce projet reprend le même langage de programmation que mon projet de pré-TPI, AndroidSnake. J'ai donc les bases pour commencer ce projet.

## **1.2 Objectifs**

Les objectifs définis pour ce projet sont les suivants :

- Le joueur peut sélectionner la langue du côté du prof et de l'élève qu'il souhaite exercer ainsi que le vocabulaire.
- Le professeur peut se déplacer latéralement.
- Les élèves avancent aussi latéralement. Cependant, ils avancent en avant quand un élève qui est à l'extrême gauche ou l'extrême droite atteint un bord.
- Un élève part en pause dès qu'il a eu 3 mots traduit correctement.
- Si un élève reçoit une fausse traduction, il avance en avant (uniquement l'élève ayant reçu l'avion en papier avance et non la ligne).
- Si le professeur a envoyé tous les élèves en pause, un écran « Fin de partie » avec le score.
- Si l'élève atteint le prof, un écran « Game Over » s'affiche.

Les use-cases que j'ai défini sont :



### 1.3 Planification initiale

Durant ce TPI, j'évoluerai avec la méthode Agile. Elle a été imposée par le chef de projet. La méthode Agile consiste à mettre en place différents sprint durant lesquels certaines tâches seront effectuées. À chaque fin de sprint, le mandataire doit fournir une démo au chef de projet pour ainsi reporter différents problèmes dans le sprint suivant. La validation est faite par le chef de projet ainsi que les deux experts. L'outil utilisé pour réaliser la planification est Trello. Les détails de chaque sprint seront dessus. Il est accessible via le lien suivant : <https://trello.com/b/KeV2VVap/student-invaders>

Voici un tableau de la planification initiale :

Sprint 1	Mise en place de l'interface	Au 17 Mai
Sprint 2	Mise en place du Prof	Au 24 Mai
Sprint 3	Mise en place des étudiants	Au 29 Mai
Sprint 4	Menu, Scoring, Début, Fin et Pause	Au 1 <sup>er</sup> Juin
Sprint 5	Tests et correction des bugs	Au 5 Juin
Rendu	Rendu final	Au 7 Juin

## 2 Analyse / Conception

### 2.1 Concept

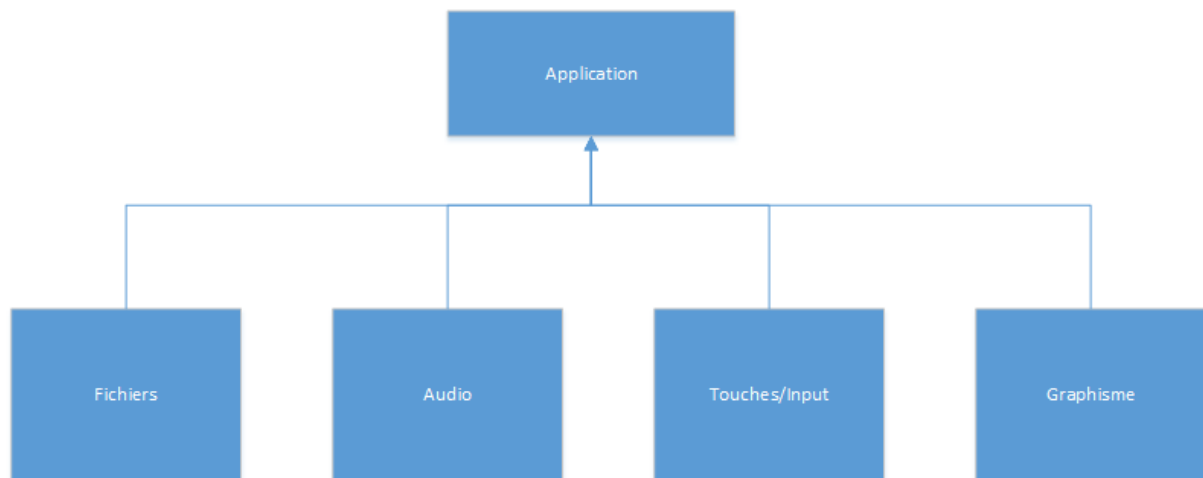
#### 2.1.1 Vue d'ensemble

L'architecture Android se compose ainsi :



Sur ce graphique, notre application sera dans la partie « Applications » et le framework LibGDX dans « Application Framework ».

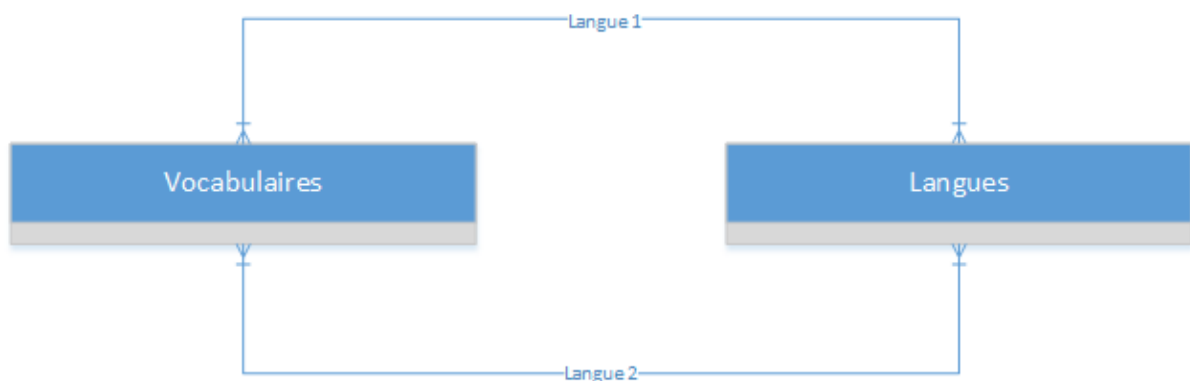
Concernant l'architecture du framework LibGDX :



L'application est le framework. Elle s'occupe de gérer les cycles du jeu ainsi que les boucles. Dans ce bloc contient différents modules utilisables :

- Le module des fichiers : Lecture/Ecriture des ressources ainsi que des assets.
- Le module de l'audio : Jouer de la musique, les effets sonores.
- Le module des touches ou Input/Output : Tous les évènements concernant les clics de souris, l'appui sur une touche du clavier ou le toucher sur un écran mobile.
- Le module Graphisme : Dessiner les sprites/images ou tout autre graphique sur l'écran.

### 2.1.2 MCD



Le MCD ne concerne que les vocabulaires et les langues. Chaque vocabulaire à deux langues et chaque langue a un ou plusieurs vocabulaires. Deux traits distincts ont été dessinés pour montrer qu'il y a une langue 1 qui concerne le prof et une langue 2 qui concerne celui de l'élève.

## 2.2 Stratégie de test

Il y aura 3 types de tests pour ce projet. Le test de robustesse, le test de performance et le test fonctionnel. L'ordre dans lequel ils seront effectués sera :

1. Test fonctionnel.
  - a. Je commence par ce test pour m'assurer que tout fonctionne correctement et que mon application ne plante pas durant les essais.
2. Test de performance.
  - a. Il s'agit de vérifier si mon application a des ralentissements quand il traite les données. Cela peut avoir un impact sur l'expérience de jeu.
3. Test de robustesse.
  - a. Au final, nous allons tester s'il y a une gestion des erreurs suivant des cas anormaux.

Dans les tests fonctionnels, des tests unitaires ~~qui~~ seront effectués tout au long du projet. Les tests systèmes auront aussi lieu pour tester la connexion entre le webservice et le jeu. Ces essais se dérouleront à chaque fin de sprint.

Pour réaliser ces tests, il faut se munir d'une tablette Android ou d'un émulateur Android sur ordinateur. Un téléphone est aussi possible pour l'utilisation mais l'application n'est pas optimisée pour ce dernier. [Mes tests ont été effectués sur une tablette « Samsung Galaxy Tab 2 » sous la version Android 4.2.2 ainsi qu'un smartphone « OnePlus 5 » sous la version Android Oreo 8.0.](#)

Comme données de test, il faudra utiliser des données réelles pour les langues disponibles ainsi que les mots traduits. Ces données sont fournies par Mr. Carrel dans une base de donnée mis à disposition. Même si je peux effectuer les tests moi-même, je vais demander à mes camarades qui ont du temps pour essayer l'application et découvrir les différents problèmes/bugs. [À la fin de chaque sprint, je vais effectuer des tests d'acceptation avec Mr. Carrel.](#)

## 2.3 Risques techniques

Un des risques techniques possible est l'implémentation du web service. En effet, je n'ai jamais appris à implémenter un web service sous le langage Java. Cependant, différents professeurs peuvent me venir en aide si nécessaire. Des tutoriels sont aussi disponibles pour l'implémentation.

Un autre risque est l'utilisation du framework LibGDX. J'ai pu m'entraîner [sur le framework dessus](#) durant le Pré-TPI mais je ne l'avais pas entièrement assimilé. Pour pallier ce manque de connaissance, j'ai acheté deux livres sur la création de jeu sous LibGDX. J'ai aussi revu quelques bases sous Java.

## 2.4 Planification

<i>Sprint</i>	<i>Démo</i>	<i>Résultat</i>
No 1	17 Mai	Manque l'affichage des vocabulaires. Nouvelle classe à créer pour utiliser les données. Deux tâches sont reportées au sprint suivant.
No 2	24 Mai	L'envoi de l'avion et la prise des mots ne sont pas encore en place. Une démo est à faire pour ces deux objectifs le 29 Mai.
No 2	29 Mai	Les deux scénarios restants ont été validés par Mr. Carrel. La prochaine présentation du sprint à lieu le 1 <sup>er</sup> Juin pour les élèves.
No 3	01 Juin	Les scénarios ont été validés par Mr. Carrel. Le 5 <sup>ème</sup> sprint est supprimé au profit du 4 <sup>ème</sup> . Ce dernier sprint aura lieu le 7 Juin.

## 2.5 Dossier de conception

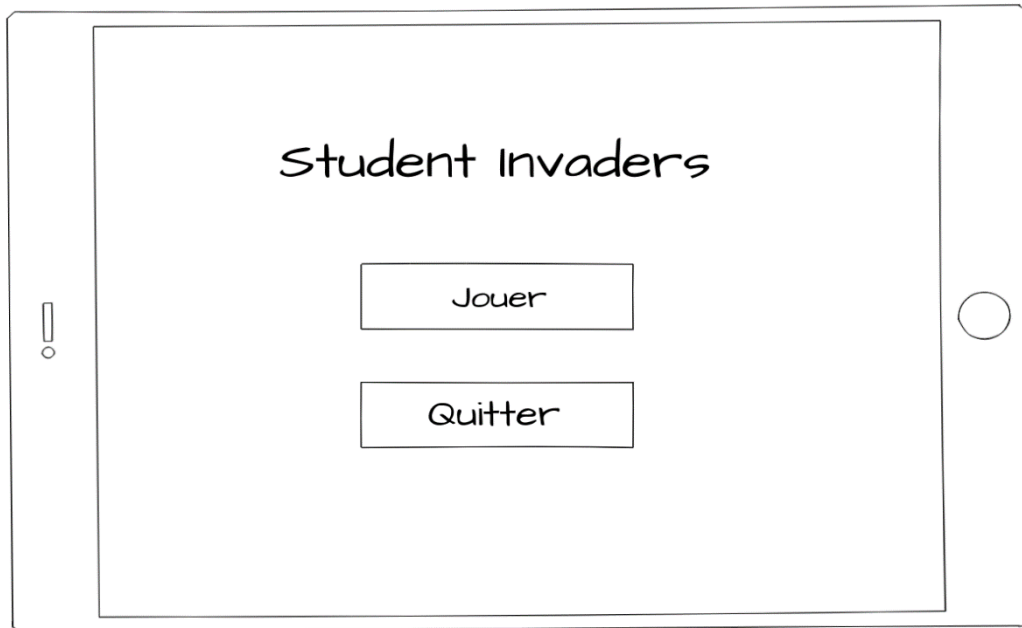
Pour ce projet, J'ai utilisé comme système d'exploitation Windows 7. J'ai ensuite choisis le logiciel Android Studio. Il a été sélectionné car il est simple d'utilisation avec les raccourcis et que c'est un classique pour le développement pour l'environnement Android. Le langage de programmation utilisé pour le jeu est Java. J'ai aussi utilisé le framework LibGDX choisi par mon chef de projet.

J'ai emprunté une tablette Samsung Galaxy Tab 2 sous Android 4.2.2 pour effectuer mes tests directement sur un appareil mobile. J'ai aussi utilisé mon smartphone qui est sous la version Oreo 8.0 pour tester la compatibilité.

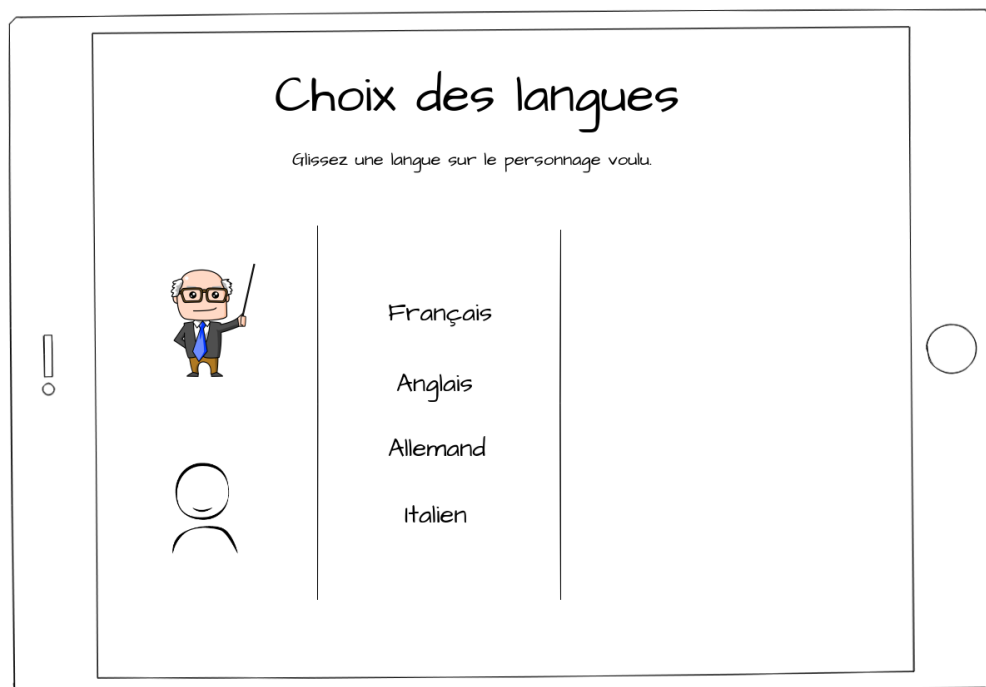


### 2.5.1 Maquettes / Use cases / Scénarios

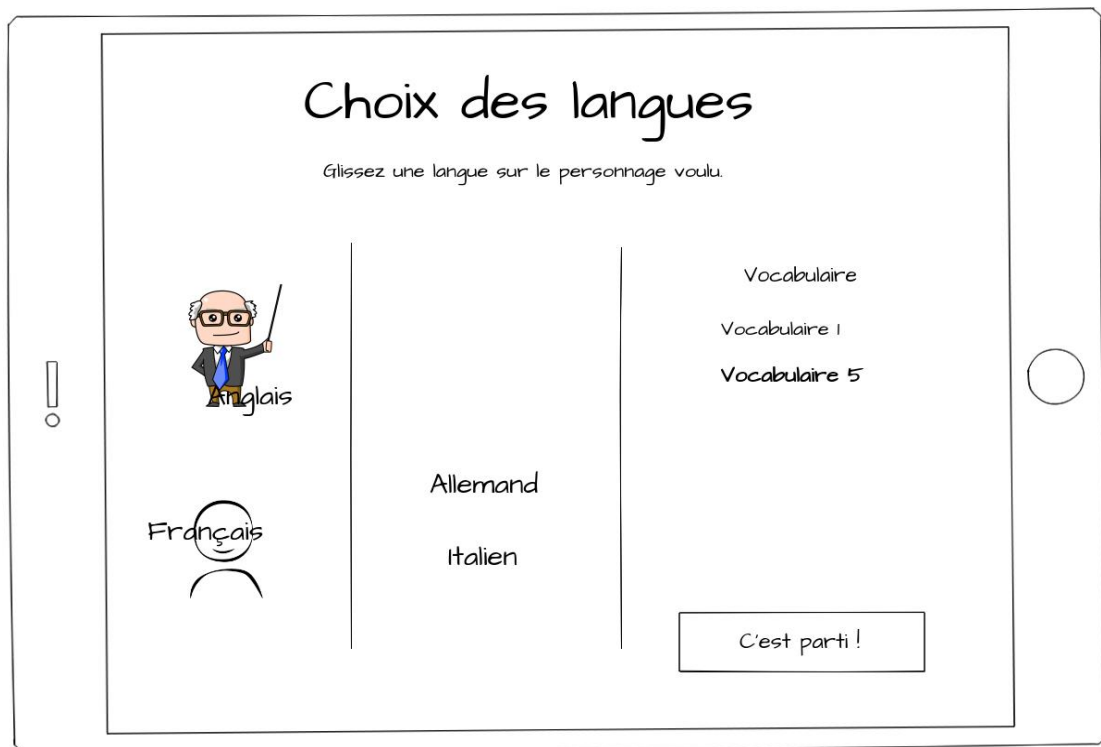
#### 1. Le menu :



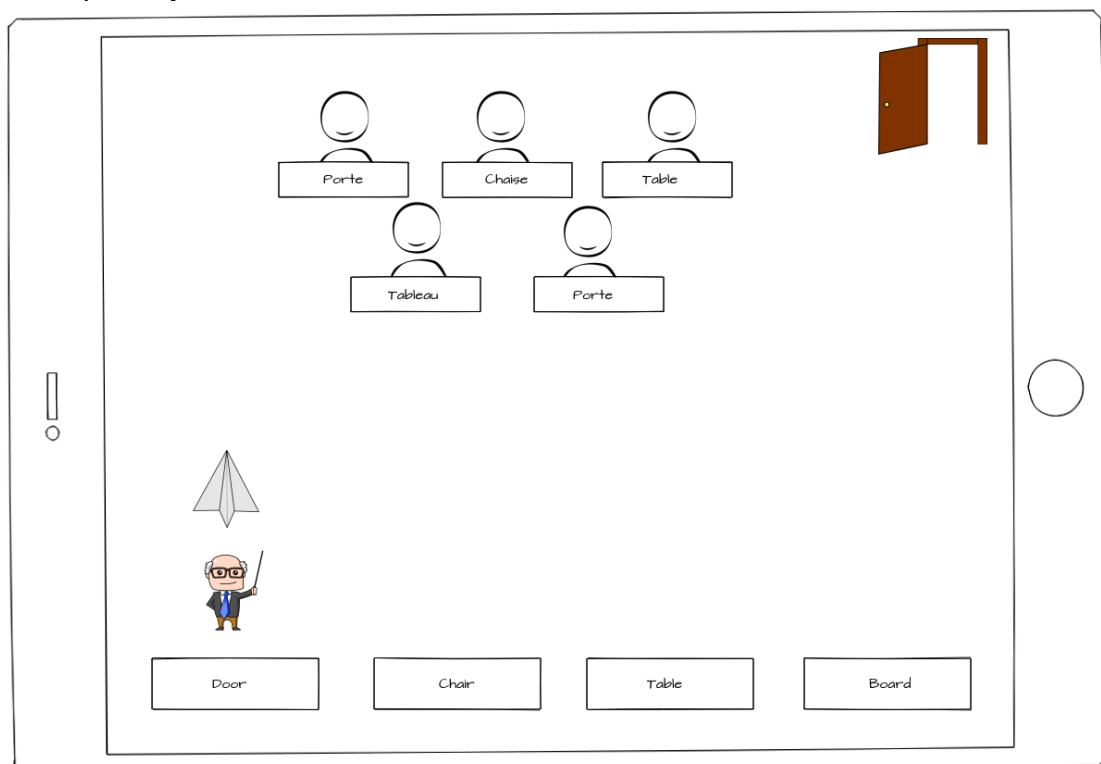
#### 2. La sélection des langues :



Langues choisies ainsi que le vocabulaire :



### 3. La partie jeu :



## Scénarios :

### 1. Menu

#### 1.1. Lancement d'une partie

Action	Condition particulière	Réponse
<b>Le joueur appuie sur le bouton « Jouer »</b>		Le jeu passe sur l'écran « Choix des langues ».

#### 1.2. Quitter le jeu

Action	Condition particulière	Réponse
<b>Le joueur appuie sur le bouton « Quitter »</b>		Retour à l'écran d'accueil de l'appareil.

### 2. Sélection des langues

#### 2.1. Le joueur choisit le français pour le professeur et l'anglais pour l'élève

Action	Condition particulière	Réponse
<b>Le joueur appuie sur le bouton « Jouer »</b>		Le jeu passe sur l'écran « Choix des langues ».
<b>Le joueur glisse avec son doigt le texte « Français » sur le professeur</b>		Un message s'affiche en indiquant que le français a été sélectionné pour le professeur.
<b>Le joueur glisse avec son doigt le texte « Anglais » sur l'élève</b>		Un message s'affiche en indiquant que l'anglais a été sélectionné pour l'élève.
<b>Le joueur clique sur le vocabulaire souhaité</b>		Un bouton « C'est parti » s'affiche.

#### 2.2. Le joueur souhaite changer de langue pour le professeur

Action	Condition particulière	Réponse
<b>Le joueur appuie sur le bouton « Jouer »</b>		Le jeu passe sur l'écran « Choix des langues ».
<b>Le joueur glisse avec son doigt le texte « Français » sur le professeur</b>		Un message s'affiche en indiquant que le français a été sélectionné pour le professeur.
<b>Le joueur glisse avec son doigt le texte « Anglais » sur l'élève</b>		Un message s'affiche en indiquant que l'anglais a été sélectionné pour l'élève.
<b>Le joueur glisse le texte « Français » à côté et met le</b>		Un message s'affiche en indiquant que le l'allemand a été sélectionné pour le professeur.

texte « Allemand » sur le professeur.		
Le joueur clique sur le vocabulaire souhaité		Un bouton « C'est parti » s'affiche.

### 2.3. Le joueur souhaite changer de langue pour l'élève

Action	Condition particulière	Réponse
Le joueur appuie sur le bouton « Jouer »		Le jeu passe sur l'écran « Choix des langues ».
Le joueur glisse avec son doigt le texte « Français » sur le professeur		Un message s'affiche en indiquant que le français a été sélectionné pour le professeur.
Le joueur glisse avec son doigt le texte « Anglais » sur l'élève		Un message s'affiche en indiquant que l'anglais a été sélectionné pour l'élève.
Le joueur glisse le texte « Anglais » à côté et met le texte « Allemand » sur le professeur.		Un message s'affiche en indiquant que le l'allemand a été sélectionné pour le l'élève.
Le joueur clique sur le vocabulaire souhaité		Un bouton « C'est parti » s'affiche.

### 2.4. Pas de vocabulaire pour la langue sélectionnée

Action	Condition particulière	Réponse
Le joueur appuie sur le bouton « Jouer »		Le jeu passe sur l'écran « Choix des langues ».
Le joueur glisse avec son doigt le texte « Français » sur le professeur		Un message s'affiche en indiquant que le français a été sélectionné pour le professeur.
Le joueur glisse avec son doigt le texte « Anglais » sur l'élève		Un message s'affiche en indiquant que l'anglais a été sélectionné pour l'élève.
-	Pas de vocabulaire retourné par le webservice.	Un message s'affiche indiquant qu'aucun vocabulaire n'est disponible pour ces deux langues sélectionnées.

### 2.5. Le joueur change de vocabulaire

Action	Condition particulière	Réponse
--------	------------------------	---------

<b>Le joueur appuie sur le bouton « Jouer »</b>		Le jeu passe sur l'écran « Choix des langues ».
<b>Le joueur glisse avec son doigt le texte « Français » sur le professeur</b>		Un message s'affiche en indiquant que le français a été sélectionné pour le professeur.
<b>Le joueur glisse avec son doigt le texte « Anglais » sur l'élève</b>		Un message s'affiche en indiquant que l'anglais a été sélectionné pour l'élève.
<b>Le joueur appuie sur « Vocabulaire 1 »</b>	Des vocabulaires sont disponibles pour les langues choisies.	Un bouton « C'est parti » s'affiche.
<b>Le joueur appuie sur « Vocabulaire 5 »</b>	Le vocabulaire 1 doit être sélectionné	Changement de vocabulaire. Le bouton « C'est parti » est toujours présent.

### 3. La partie jeu

#### 3.1. Prof :

##### 3.1.1. Le prof se déplace de gauche à droite

Action	Condition particulière	Réponse
<b>Le joueur appuie sur une partie précise à gauche de l'écran.</b>	L'endroit appuyé doit être au même niveau que le prof.	Le prof bouge petit à petit vers sa gauche.
<b>Le joueur appuie sur une partie précise à droite de l'écran.</b>	L'endroit appuyé doit être au même niveau que le prof.	Le prof bouge petit à petit vers sa droite.

##### 3.1.2. Le prof choisit un mot

Action	Condition particulière	Réponse
<b>Le joueur appuie sur une partie précise à gauche de l'écran.</b>	L'endroit appuyé est au même niveau que le prof.	Le prof bouge petit à petit vers sa gauche.
<b>Le joueur appuie sur le mot en dessous.</b>	Le prof n'est pas au-dessus du rectangle.	Le prof continue son déplacement.
<b>Le joueur appuie sur le mot en dessous.</b>	Le prof est au-dessus du rectangle.	L'avion en papier se forme en dessus du professeur et le professeur se tourne.
<b>Le joueur appuie sur une partie précise à droite de l'écran.</b>		Le prof bouge petit à petit vers sa droite avec l'avion en dessus de lui.

##### 3.1.3. Le prof lance un avion en papier

Ce scénario a lieu une fois que le prof a choisi un mot.

Action	Condition particulière	Réponse
<b>Le joueur appuie sur une partie précise à gauche de l'écran.</b>	L'endroit appuyé est au même niveau que le prof.	Le prof bouge petit à petit vers sa gauche.
<b>Le joueur appuie sur une partie précise à droite de l'écran.</b>		Le prof bouge petit à petit vers sa droite avec l'avion en dessus de lui.

Action	Condition particulière	Réponse
<b>Le joueur appuie sur une partie de l'écran au-dessus du prof</b>		L'avion part en ligne droite.
	L'avion atteint le bord de l'écran.	L'avion disparaît et le même mot réapparaît en bas de l'écran.

#### 3.1.4. Le prof a envoyé tous les élèves en pause

Action	Condition particulière	Réponse
<b>Le joueur joue sa partie.</b>		Le prof lance des avions en papier, les élèves avancent.
-	Tous les élèves ont été envoyés en pause.	Un écran « Bien joué ! » s'affiche en indiquant le score du joueur ainsi que la possibilité de rejouer ou de changer la langue.

#### 3.1.5. L'élève atteint le prof

Action	Condition particulière	Réponse
<b>Le joueur joue sa partie.</b>		Le prof lance des avions en papier, les élèves avancent.
-	L'élève atteint le prof.	Un écran Game Over s'affiche en proposant de rejouer ou de retourner au menu.

### 3.2. Elève :

#### 3.2.1. L'élève reçoit un mot correct

Le scénario commence quand le prof a sélectionné un mot

Action	Condition particulière	Réponse
<b>Le joueur appuie sur une partie de l'écran au-dessus du prof</b>		Le prof lance l'avion.
-	L'avion touche l'élève qui porte le bon mot.	L'élève se dirige vers la sortie.
-	L'élève arrive à la porte.	L'élève disparaît.

## 3.2.2. L'élève reçoit un mot incorrect

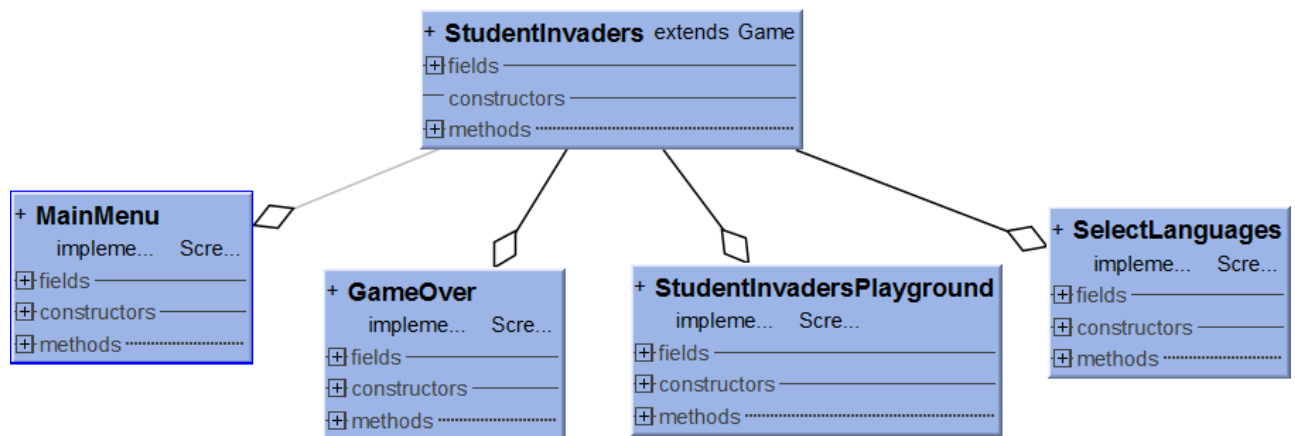
Le scénario commence quand le prof a sélectionné un mot

Action	Condition particulière	Réponse
<b>Le joueur appuie sur une partie de l'écran au-dessus du prof</b>		Le prof lance l'avion.
-	L'avion touche l'élève qui porte un mauvais mot.	L'élève avance vers le bas.
-	L'élève a avancé pendant 2 secondes.	L'élève reprends son déplacement précédent.



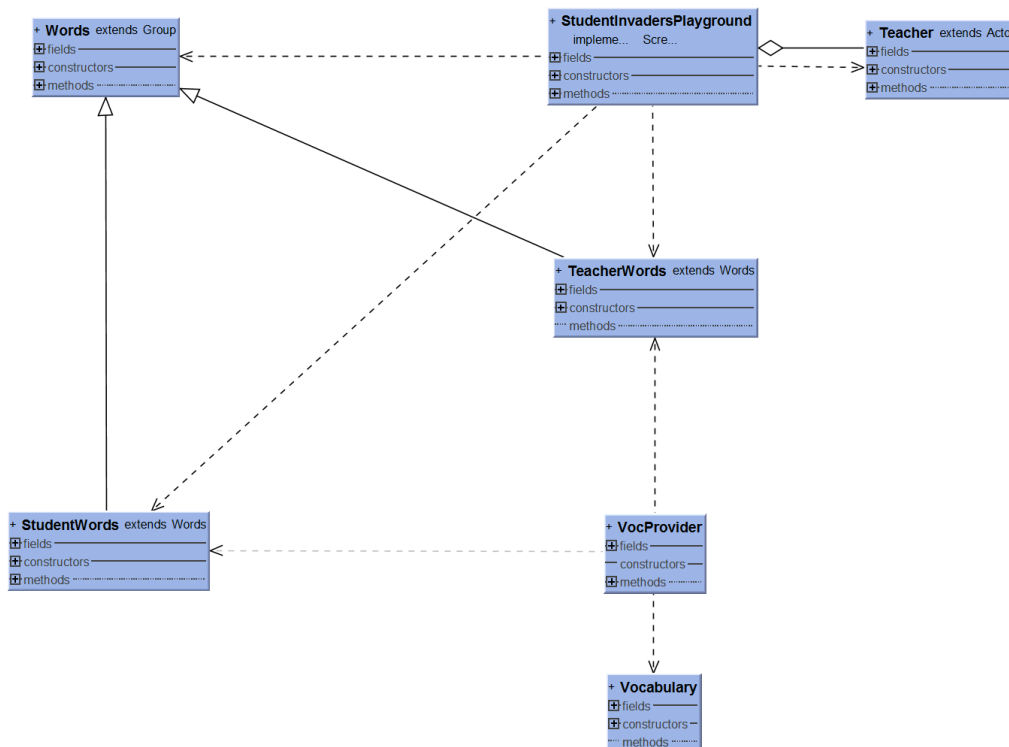
## 2.5.2 MLD

Pour faciliter la lisibilité, j'ai séparé le diagramme de classe en deux parties. La première partie concerne la gestion des scènes.



Chaque classe (**MainMenu**, **GameOver**, **StudentInvadersPlayground**, **SelectLanguages**) utilise la classe principale, **StudentInvaders**. Cette dernière fournit à chacune la taille de l'écran, le stage (un « terrain » qui peut contenir des acteurs) ainsi que différentes méthodes pour changer de scène. Excepté la classe **StudentInvaders**, les autres classes ne dépendent pas des autres.

La deuxième partie concerne les mots pour les profs et les élèves.



La classe « StudentInvadersPlayground » est le terrain de jeu. Il charge les mots ainsi que les boîtes et élèves. Deux classes héritent de la classe « Word ». Il s'agit de « StudentWords » et « TeacherWords ». La particularité est que « StudentWords » a des états et gère donc chaque élève et « TeacherWords » a un booléen pour savoir si le mot du professeur a été appris ou non.

### 2.5.3 Particularité 1 – Les blocs pour les mots du professeur

Arrivé à la moitié du projet, un problème s'imposait au moment d'afficher les mots pour le professeur. En effet, durant la programmation, je me suis posé la question suivante : « Comment faire pour afficher ces boîtes qui sont dans un tableau ? Dois-je afficher ces blocs en continu même s'ils seront cachés car ils dépassent l'écran ou dois-je en mettre 4 et dès qu'un mot a été traduit, on le supprime pour le remplacer par le suivant ? ». J'ai donc posé la question à mon chef de projet et il m'a conseillé de partir avec la première méthode car elle est plus simple. Pour cela, j'ai dû modifier le code en ajoutant une nouvelle classe qui est « TeacherWords ». Cette classe permet de connaître les mots que le prof possède ainsi que de savoir si un mot a été appris.

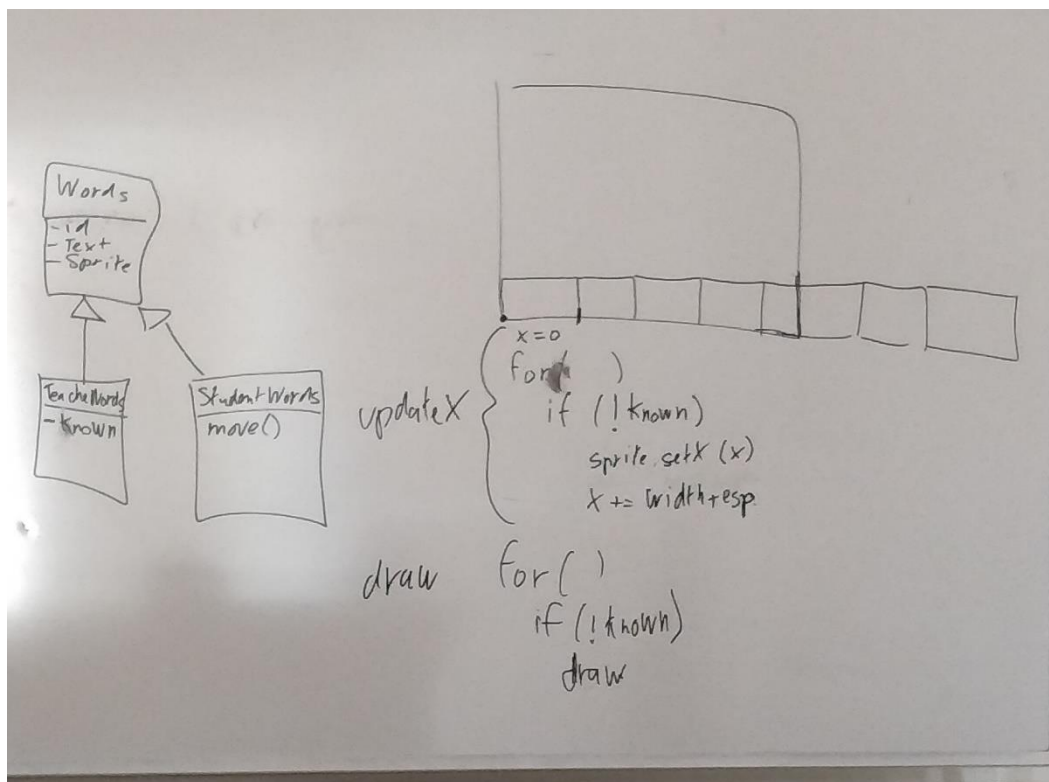


Figure 1. Photo de la discussion avec Mr. Carrel

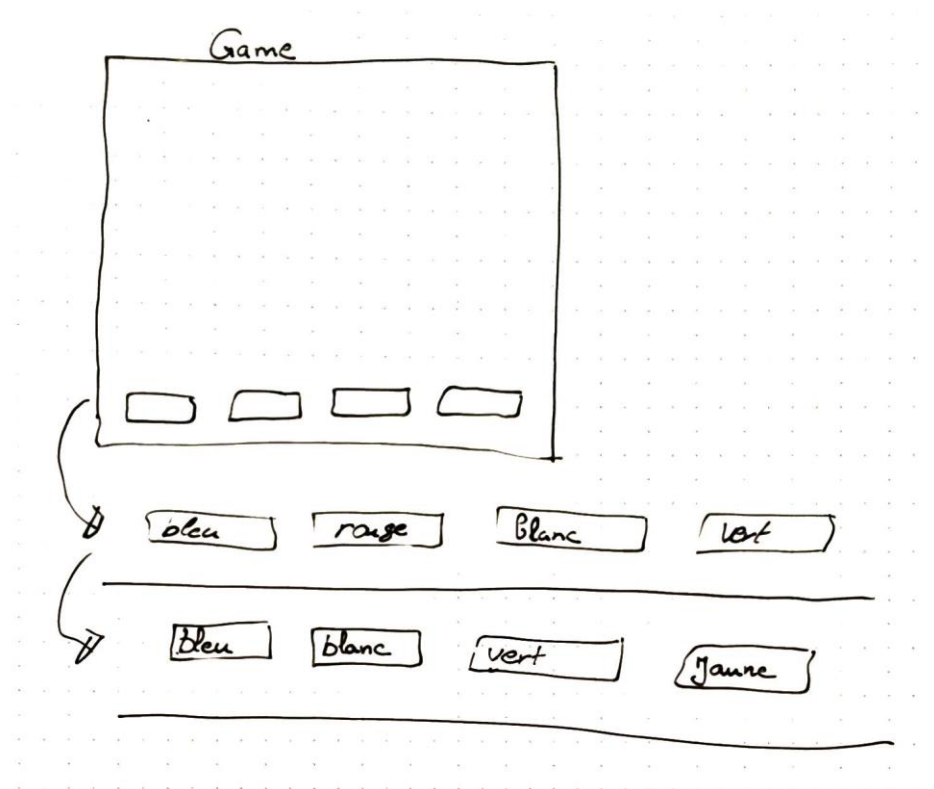


Figure 2. Dessin clair pour l'explication

Pour reprendre la figure 2, voici comment sera exécuté la méthode d’affichage des mots. Les premiers mots affichés, par exemple, seront « Bleu, Rouge, Blanc et Vert ». Si le joueur appuie sur « Rouge », le bloc disparaît à la fin de l’envoi de l’avion. Après cela, les blocs vont se décaler en reprenant l’ancienne position du bloc visible. Avec cette position, on additionne la largeur du bloc et un espace pour espacer les mots. De cette manière, nous évitons de modifier des valeurs dans un tableau et ainsi éviter des bugs qui pourraient intervenir (l’utilisation d’un index récupéré).

## 2.5.4 Particularité 2 – Le changement de scènes

Ayant commencé le projet en créant le menu ainsi que la sélection des langages, un détail de taille m'avait échappé, la fuite de mémoire. Cela consiste à polluer la mémoire de manière croissante sans le vouloir et sans pouvoir le contrôler. Par exemple, un tableau qui continue d'ajouter des valeurs à la suite sans s'arrêter et qui prends de la place dans la mémoire sans le vouloir causant ainsi l'arrêt de l'application. Pour éviter ces fuites, il faut vider la mémoire à chaque changement de scène ou faire attention aux boucles utilisés dans le code qui peuvent tourner en continu.

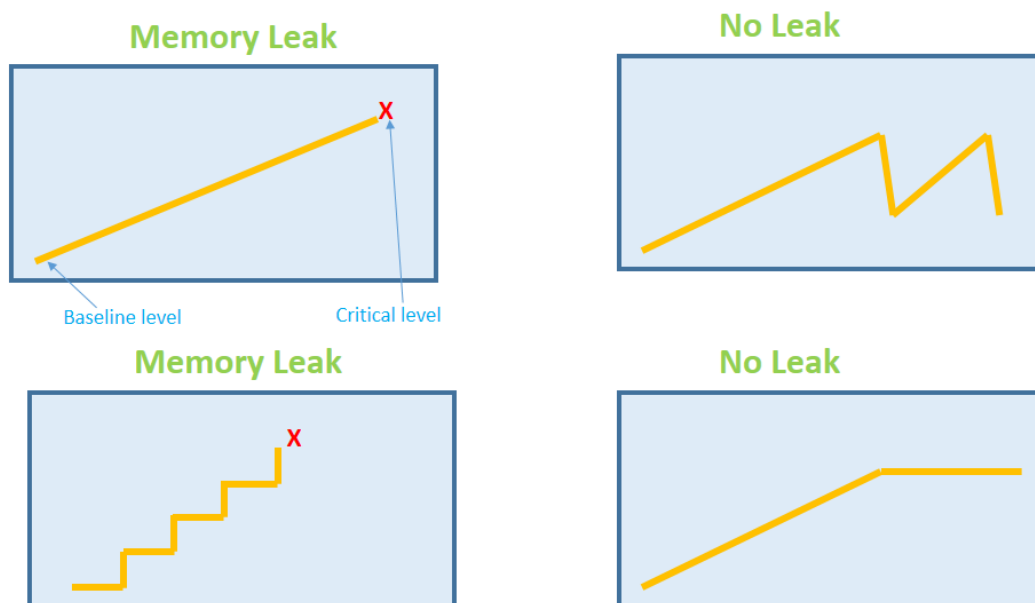


Figure 3. Explications des différentes fuites de mémoire

N'ayant jamais expérimenté ceci dans mes projets précédents, je ne comprenais pas le problème quand cela était arrivé. On peut remarquer des ralentissements sur la tablette mais pas sur mon smartphone. J'ai ainsi commencé à déboguer mon application avec mon téléphone parce que les derniers smartphones permettent de montrer l'utilisation de la mémoire sous Android Studio. J'ai pu identifier le problème et chercher à corriger le problème. Cela ne fût pas si simple mais j'ai réussi à trouver la solution pour cette partie. Après ce souci, j'ai décidé de mener des tests en faisant plusieurs parties d'affilés pour voir si le jeu continuait d'avoir des fuites. Quelques essais plus tard, j'avais effectivement un problème de taille après plusieurs parties effectuées. N'ayant pas utilisé correctement la méthode pour le changement de scène, il s'avérait que ma mémoire continuait d'augmenter car les précédentes ressources n'étaient pas effacées correctement (je supprimais que les acteurs des précédentes scènes). Cela affecte l'expérience du joueur car il ne pourra pas enchaîner ses parties.

J'ai aussi eu ce problème par la suite sur le terrain de jeu (quand le prof doit envoyer des avions sur les élèves). Quand je ramassais un mot, le jeu commençait à utiliser de plus en plus de la mémoire sans aucune raison mais quand je lançais l'avion ramassé, elle n'était plus polluée. Si je laissais l'avion ramassé sans le tirer, la RAM augmentait considérablement jusqu'à l'arrêt de l'application.

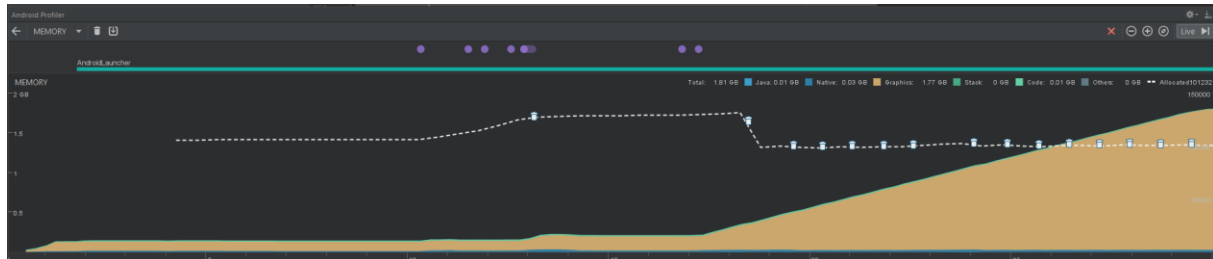


Figure 4. La fuite de mémoire dans mon projet.







Au début, je n'avais pas identifié le problème avant d'avoir fait des tests avec mon smartphone car il y a un système de monitoring (non disponible avec la tablette). Quand j'ai découvert l'origine du problème, j'ai dû regarder à quel moment la mémoire se remplissait. Après quelques heures de recherches, j'ai compris qu'une de mes boucles dans le code continuait à tourner même après avoir l'avion et cela causait cette pollution. J'ai pu ainsi mettre en place une solution pour cette erreur.

### 3 Réalisation

#### 3.1 Dossier de réalisation

Pour ce projet, j'ai utilisé la plateforme Github. Il est accessible au lien suivant : <https://github.com/Seni-J/Student-Invaders>. Cela permet un suivi constant au chef de projet à chaque mise à jour de l'application.

Le repository Github se présente ainsi :

 <b>Code</b>	Ajout des boxes "dynamiques" + Doc
 <b>Documentation</b>	MaJ dossier
 <b>Sprites/Inkscape</b>	les mots peuvent être pris et un avion peut être lancé.
 <b>.gitattributes</b>	Initial commit
 <b>.gitignore</b>	Ajout de la documentation
 <b>README.md</b>	Mise à jour du README

- Le dossier **Code** contient le code source du projet. On y retrouve donc les fichiers de classe Java.
- Le dossier **Documentation** contient le cahier des charges, le journal de travail, la documentation, les maquettes, les différents diagrammes ainsi que les anciennes versions de la documentation.
- Le dossier **Sprites/Inkscape** contient les différents sprites que j'ai créé tout au long du projet.
- Le fichier **README.md** contient le manuel d'installation ainsi qu'une description du projet.

Les logiciels utilisés ainsi que les versions :

Nom	Version
Android Studio	3.0.1
LibGDX	1.6.1
GitHub Desktop	1.2.0
Inkscape	0.92.2
Samsung Galaxy Tab 2	4.2.2
Mockflow (web)	-
Trello (web)	-
Version Android de la tablette	4.2.2

### 3.2 Description des tests effectués

Chaque scénario a été testé par moi ainsi que mon chef de projet durant les sprints review. Les tests ont été effectués sur tablette.

#### Menu et sélection des langues

Scénario	17.5	24.5	1.5	7.5
1.1 Lancement d'une partie	Dév → OK	CdP → OK	CdP → OK	Dév → OK
1.2 Quitter le jeu	Dév → KO	Dév → KO	Dév → KO	Dév → KO
2.1 Le joueur choisit le français...	Dév → OK	CdP → OK	CdP → OK	Dév → OK
2.2 Changement de langue pour le prof	Dév → OK CdP → OK	Dév → KO	Dév → KO	Dév → KO
2.3 Changement de langue pour l'élève	Dév → OK CdP → OK	Dév → KO	Dév → KO	Dév → KO
2.4 Pas de vocabulaire pour la langue sélectionnée	Dév → OK	CdP → OK	CdP → OK	Dév → OK
2.5 Le joueur change de vocabulaire	Dév → OK	Dév → OK	Dév → OK	Dév → OK

#### Professeur

Scénario	17.5	24.5	1.5	7.5
3.1.1 Le prof se déplace de gauche à droite 1.1 Lancement d'une partie	Dév → OK	CdP → OK	CdP → OK	Dév → OK
1.2 Quitter le jeu	Dév → KO	Dév → KO	Dév → KO	Dév → KO
32.1.2 le prof Le joueur choisit un mot le français...	Dév → OK	CdP → OK	CdP → OK	Dév → OK
3.1.32.2 Changement de langue pour le prof lance un avion en papier	Dév → OK CdP → OK	CdP → OK Dév → KO	CdP → OK Dév → KO	Dév → OK KO

3.1.4 Le prof a envoyé tous les élèves en pause	<a href="#">Dév → OK</a> <a href="#">CdP → OK</a>	<a href="#">Dév → KO</a>	Dév → KO	Dév → KO
2.4 Pas de vocabulaire pour la langue sélectionnée	<a href="#">Dév → OK</a>	<a href="#">CdP → OK</a>	<a href="#">CdP → OK</a>	<a href="#">Dév → OK</a>
3.1.5 L'élève atteint le prof	<a href="#">Dév → OK</a>	<a href="#">Dév → OK</a>	<a href="#">CdP Dév → OK</a>	Dév → OK

Élève  
Professeur

Scénario	17.5	24.5	1.5	7.5
3.2.1 L'élève reçoit un mot correct		<a href="#">CdP → OK</a>	CdP → OK	Dév → OK
3.1.1 Le prof se déplace de gauche à droite				
3.1.2.2 l'élève reçoit le prof choisit un mot incorrect		<a href="#">CdP → OK</a>	CdP → OK	Dév → OK
3.1.3 Le prof lance un avion en papier		<a href="#">CdP → OK</a>	<a href="#">CdP → OK</a>	<a href="#">Dév → OK</a>
3.1.4 Le prof a envoyé tous les élèves en pause			<a href="#">Dév → KO</a>	<a href="#">Dév → KO</a>
3.1.5 L'élève atteint le prof			<a href="#">CdP → OK</a>	<a href="#">Dév → OK</a>

*Pour chaque partie testée de votre projet, il faut décrire:*

- les conditions exactes de chaque test
- les preuves de test (papier ou fichier)
- tests sans preuve: fournir au moins une description

Echéance 4  
Echéance 5

Élève



Scénario	17.5	24.5	1.5	7.5
<a href="#">3.2.1 L'élève reçoit un mot correct</a>			<a href="#">CdP → OK</a>	<a href="#">Dév → OK</a>
<a href="#">3.2.2 l'élève reçoit un mot incorrect</a>			<a href="#">CdP → OK</a>	<a href="#">Dév → OK</a>

### 3.3 Erreurs restantes

Différents soucis ont été remarqués durant mes tests :

- Une surcharge de la mémoire
  - Si l'on enchaîne plusieurs parties, la mémoire vive n'est jamais vidée et elle se surcharge petit à petit causant des ralentissements sur l'application voire même le crash de cette dernière.
  - Le joueur pourrait voir son jeu fermé en cours de partie (après 5-6 parties).

Solution : Réécrire une partie du code pour qu'à chaque changement de scène, on supprime les textures, textes ou autre qui ne servent plus.

- Taille des textes
  - S'il y a plus de 5 langages disponibles, les suivants seront en dehors de l'écran.
  - Le texte sur smartphone est petit et peu lisible.

Solution : Pour pallier à ce problème, il faut différencier l'appareil et ainsi changer la taille des sprites ainsi que des textes.

- [Le changement de langue](#)
  - [Cette fonction était disponible dans les premières versions du jeu. Au fur et à mesure des modifications du code \(changements demandés suite à la fin du sprint 1\), cette fonctionnalité ne marchait plus.](#)
  - [L'impact est que le joueur devra relancer le jeu ou lancer la partie pour ensuite perdre et revenir sur le menu de la sélection des langues.](#)

[Solution : Revoir le code de la sélection des langues car le problème est dessus.](#)

### 3.4 Liste des documents fournis

~~Lister~~ Les documents fournis [au client](#) avec [votre produit, en indiquant les numéros de versions ce projet](#) sont :

- Le rapport de projet [en format PDF et papier. Version 1.8.](#)
- [Le code source. Version 1.0.](#)
- [Le journal de travail. Version 1.0.](#)

- [Un lien du projet sur Github avec le manuel d'installation \(en annexe\) d'installation. Version 1.0](#)
  - ~~le manuel d'Utilisation avec des exemples graphiques (en annexe)~~
  - ~~autres...~~

Echéance 5

## 4 Conclusions

*Développez en tous cas les points suivants:*

- *Objectifs atteints / non-atteints*
- *Points positifs / négatifs*
- *Difficultés particulières*
- *Suites possibles pour le projet (évolutions & améliorations)*

## Echéance 5

### 5 Annexes

#### 5.1 Résumé du rapport du TPI / version succincte de la documentation

##### **STUDENT INVADERS**

*Jeu didactique d'apprentissage de vocabulaire sur Android*

##### Situation de départ

Développer une application Java tournant sur les smartphones et tablettes Android qui permet d'apprendre divers vocabulaires fournis par un webservice. Le jeu se déroule de la manière suivante : nous avons des élèves qui souhaitent connaître une traduction d'un mot et le professeur, qui a des mots à disposition, peut prendre le mot correspondant à une traduction du mot de l'élève, faire un avion en papier et l'envoyer sur l'élève correspondant. Ce jeu est dans le même style que « Space Invaders ».

##### Matériels utilisés

- Une tablette Android 4.2.2
- Un poste de développement

##### Logiciels

- Android Studio 3.0.1
- Framework LibGDX

##### Mis en place

L'objectif principal du jeu est d'envoyer tous les élèves sur l'écran en pause. Les autres objectifs étaient l'envoi d'un avion en papier, déplacer le professeur latéralement, prendre différents mots et faire avancer un élève quand la traduction est fausse.

J'ai mis en place le menu du jeu avec la sélection des langues. Par la suite, j'ai mis l'enseignant en premier. N'ayant pas le webservice à disposition avant la fin du projet, j'ai dû mettre en place des données de test. Avec ces données de test, j'ai mis en place les mots pour l'enseignant. Après cela, j'ai fait les élèves. Pour faciliter le changement des mots, chaque élève est indépendant.

##### Conclusion

Sur la dernière version du projet, l'élève ne dispose que d'un seul mot. Le web service a été mis en place. Les autres objectifs demandés dans le cahier des charges ont été réalisés.

## 5.2 Glossaire

Framework	

### 5.25.3 Sources – Bibliographie

« LibGDX Game Development By Example » par James Cook, Août 2015.

La documentation LibGDX :

<https://libgdx.badlogicgames.com/ci/nightlies/docs/api/allclasses-noframe.html>

StackOverflow pour trouver des solutions à des questions déjà posée :

<https://gamedev.stackexchange.com>

<https://stackoverflow.com>

Les tutoriels de « HollowBit » sur Youtube :

<https://www.youtube.com/playlist?list=PLrnO5Pu2zAHKAIjRtTLAXtZKMSA6JWnmf>

Les tutoriels de « Brent Aureli's - Code School » sur Youtube :

<https://www.youtube.com/playlist?list=PLZm85UZQLd2SXQzsF-a0-pPF6IWDDdrXt>

Xavier Carrel – Différentes solutions apportées pour mes problèmes.

Vincent Fagioli – Tests et aide à la recherche de différentes ressources pour le jeu.

Stuart Gueissaz – Aide à la réflexion d'une solution pour le vocabulaire.

*Liste des livres utilisés (Titre, auteur, date), des sites Internet (URL) consultés, des articles (Revue, date, titre, auteur)... Et de toutes les aides externes (noms)*

## 5.4 Journal de bord

Date	Événement
17.05	Revue du Sprint 1. L'affichage du vocabulaire n'est pas fait, il est reporté au sprint suivant.
22.05	Visite de Mr. Tièche pour voir l'avancement du projet.
24.05	Revue du Sprint 2. L'envoi des avions et la prise des mots ne sont pas fonctionnels. Les scénarios ainsi que le MCD/Vue d'ensemble ont été revus et validés. Une deuxième démo a lieu le 29 Mai pour montrer l'envoi des avions ainsi que la prise des mots.
29.05	Revue du Sprint 2 restant. Validation des envois et la prise des mots.

01.06	<a href="#">Revue du Sprint 3. Validé par Mr. Carrel. Le sprint planning a été fait pour le suivant. Le Sprint 5 sera annulé.</a>
-------	---

*Référence au journal de travail externe. Inclus ici seulement si c'est exigé par l'expert.*

Echéance 1  
Echéance 2  
Echéance 3  
Echéance 4  
Echéance 5

## 5.5 Manuel d'Installation

Important !

~~Echéance 4 (Readme dans Git)~~

Le manuel d'installation est disponible sur le projet Github. Il est accessible avec le lien suivant : <https://github.com/Seni-J/Student-Invaders/blob/master/README.md>

## 5.6 Manuel d'Utilisation

Pas important (pour XCL). Ou plutôt : pas prioritaire

## 5.7 Archives du projet

*Media, ... dans une fourre en plastique*