# Travail final : Le Jeu « 97 Cartes »

Dans ce travail final, vous devrez développer une application Android qui sera un jeu de cartes virtuelles. Le fonctionnement du jeu est relativement simple et fortement inspiré du jeu *98 Cards* que vous pouvez télécharger sur Google Play.

Vous pouvez également voir le jeu en action dans le vidéo suivant :

<https://www.youtube.com/watch?v=KzLhZz3XHt0>

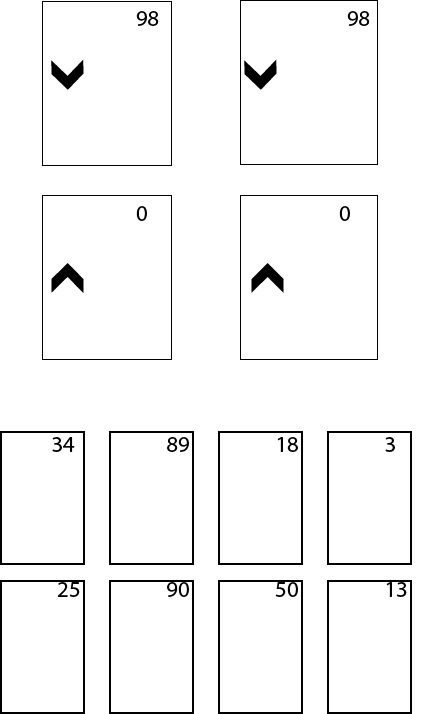
ou une partie que j'ai faite présentée sur LÉA

Vous serez évalué sur :

* **Le modèle objet sur lequel repose l’application**
* Les activités ( 3 ) permettant de jouer au jeu
* Le bon fonctionnement de celui-ci
* La persistance du pointage à l’aide d’un table SQLite

## 1Règles du jeu

1. Le jeu est composé de 4 suites ( 2 croissantes commençant à 0, 2 descendantes commençant à 98 ) et de 8 cartes initiales formant notre jeu :



1. Les 8 cartes initiales sont numérotées **aléatoirement** entre 1 – 97
2. Le but est d’écouler le maximum des 97 cartes ( dont 8 sont présentes actuellement ) sur les piles en respectant l’ascendance ou la descendance.

EX. : On peut placer la carte 3 sur la pile croissante où c’est indiqué 0; elle vaut maintenant 3

On pourrait ensuite placer la carte 25 sur la pile où est la carte 3 car 25 > 3.

1. Quand deux cartes sont libres, 2 autres apparaissent toujours avec un nombre aléatoire ( autres que celles déjà sorties )
2. Une règle spéciale existe : la règle de 10. Si une carte vaut 10 de plus que la carte sur une suite descendante, on peut la placer. De la même façon, on peut placer une carte de 10 unités inférieure à la carte sur une pile ascendante.

EX : On place la carte 13 sur une pile initiale ascendante 13 > 0 OK !

On peut ensuite placer la carte 3 sur la carte 13 même si la pile est ascendante car 13 est exactement de 10 plus grand que 3.

Autre ex : à 2m01 sur le vidéo ci-haut

1. Si on joue une carte sur une pile et qu’elle ne fonctionne pas, elle est retournée à sa place de départ. Si on prend une carte et qu’on ne la dépose pas sur une pile, elle doit également revenir à sa place de départ.
2. La partie se termine si on a réussi à placer les 97 cartes ( Bonne chance ! ) ou si aucun déplacement n’est possible.
3. Comme dans la démonstration une méthode « annuler dernier mouvement » devra retourner la dernière carte jouée dans le jeu de l’utilisateur
4. On doit afficher en tout temps le nombre de cartes restantes à placer et un pointage. Le pointage doit être fonction :

* La proximité de la carte jouée de la carte sur la suite
* La vitesse avec laquelle on a joué la carte
* Le nombre de cartes restantes ( plus de points à mesure qu’on avance dans le jeu )

1. Finalement, un chronomètre ( <https://devstory.net/12625/android-chronometer> ) doit être présent indiquant le nombre de temps en secondes depuis le début du jeu.

## 2,Modèle objet

Vous devez coder des classes représentant les objets qui seront utilisés lors du jeu, en veillant à représenter la réalité le plus fidèlement possible; ces classes doivent contenir des attributs et des méthodes représentant les différentes actions à accomplir durant le jeu.

Toute la logique du jeu doit être représentée dans ces classes ( incluant la fin du jeu ). Donc toutes les opérations présentées de A à H dans la section 1.

Vous utiliserez des objets provenant de ces classes dans vos 3 Activités ainsi que leurs méthodes

## 3.La vue, les 3 Activités :

3 Activités porteront le projet :

1. L’activité de départ sera relativement simple à déployer. Elle doit contenir :

* Le nom du jeu
* Un bouton permettant d’aller à l’activité contenant le jeu
* Le plus haut pointage que vous avez réussi ( en vous référant à votre table SQLite )

1. L’activité de jeu

Dessinez l’interface avec soin; vous pouvez utiliser des ConstraintLayout et des LinearLayout. Pour les cartes, j’ai utilisé de simples TextView mais vous avez le champ libre.

Présentez également des espaces pour le nombre de cartes restantes, le chronomètre et les points.

La gestion des événements est principalement axée sur le « Drag & Drop » qu’on a pratiqué à l’annexe 14. **Faites-en sorte qu’on ne puisse pas « perdre » de cartes si on les dépose ailleurs que sur les 4 piles.**

1. L’activité de conclusion

Lorsqu’on ne peut plus jouer, faites apparaître la 3ème activité qui mentionnera à l’usager que la partie est terminée et qui affichera les différents pointages à ce jour.

## La base de données simple

Lorsque la partie est terminée ( lorsque plus aucun mouvement de cartes n’est possible ), vous devez également écrire le pointage dans une table d’une base de données locale au téléphone. Vous pouvez y entrer d’autres renseignements pertinents à votre guise ( date ). Le plus haut pointage à ce jour doit être affiché dans l’activité de départ.

Un singleton doit être responsable de toutes les opérations relatives à la base de données; d'autres singletons peuvent être utilisés pour d'autres classes du modèle, à vous de voir !

1. **Grille d’évaluation sommaire**

|  |  |  |
| --- | --- | --- |
| Modèle objet | * Classes servant de modèle aux éléments du jeu ( Carte, Pile, Partie, etc ) * Génération des cartes à valeur aléatoire * Attributs * Méthodes représentant les actions à effectuer dans le jeu | 28% |
| Activités | * Présentation * Fonctionnalités * Chronomètre * Pointage * Impression artistique ( 5 % ) | 58% |
| Accès à la base de données |  | 8% |
| Commentaires / indentation / standards de programmation |  | 6% |
| Bonus | Me les mentionner à la remise | +??% |
| Votre version n'est pas compatible avec la version d'AndroidStudio de l'école |  | -5% |
| Retard |  | -10% par jour de retard |
| Le logiciel MOSS développé à l’université Stanford sera utilisé <https://theory.stanford.edu/~aiken/moss/>. Si la comparaison de deux travaux identifie un nombre de lignes identiques considérablement supérieur à la moyenne des comparaisons des autres paires de travaux, les deux travaux obtiendront la note 0. | | |