

PROJET INF431 — Set multijoueurs

Objectifs :

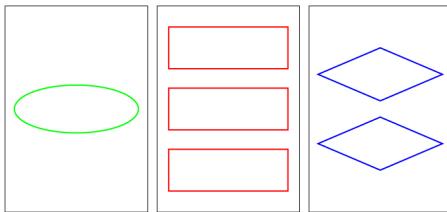
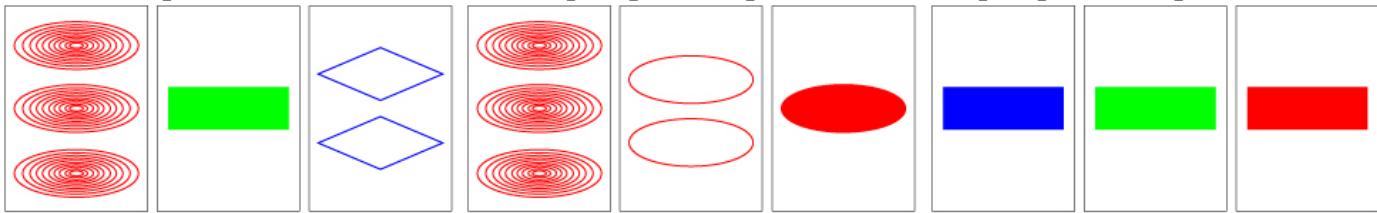
- Développement d'une application (processus de développement, manipulation des outils).
- Manipulation des threads sous Java et/ou Android.
- Manipulation des sockets pour une application distribuée.

1. Présentation du jeu

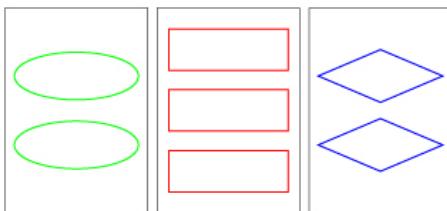
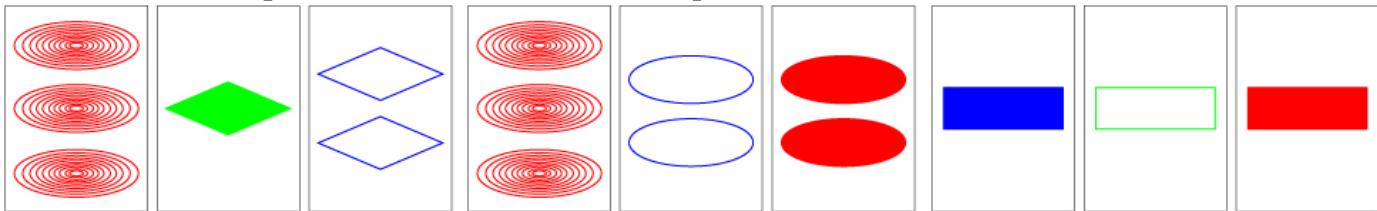
Set! © est un jeu de cartes qui peut se jouer à un ou plusieurs joueurs. Le jeu est constitué de 81 cartes toutes différentes qui se distinguent selon 4 caractéristiques :

- Nombre : 1, 2 ou 3 objets,
- Couleur : rouge, vert ou bleu,
- Remplissage : vide, hachuré ou plein,
- Forme : ovale, rectangle ou losange.

Un set est un ensemble de trois cartes qui sont soit toutes les trois identiques, soit toutes les trois différentes, pour chacune des caractéristiques prises séparément. Voici quelques exemples de sets :



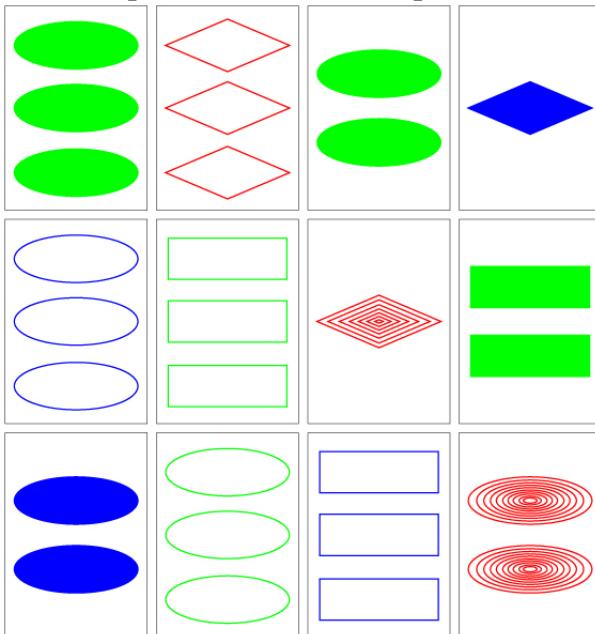
En revanche, les triplets suivants de cartes ne sont pas des sets :



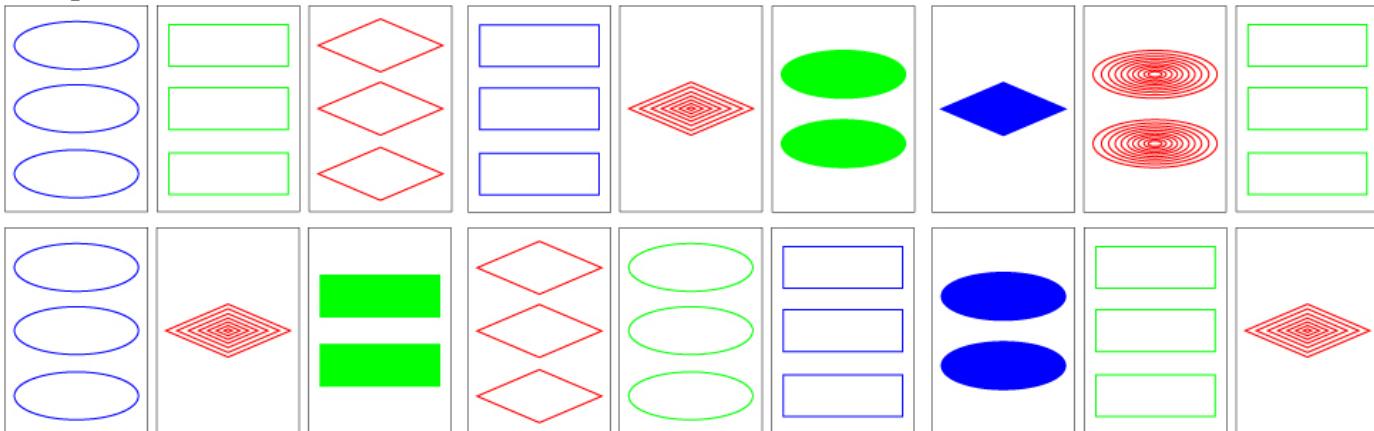
Dans le premier, il y a deux cartes losanges alors que la troisième est une carte ovale ; dans le second, il y a deux cartes rouges alors que la troisième est bleue ; etc.

Le jeu se déroule de la manière suivante : on dispose 12 cartes sur la table (en général sur une grille 3 x 4). Tout le monde joue en même temps. Le premier joueur qui voit un set dit « Set ! » et le montre aux autres. Si le set est bon, le joueur prend les trois cartes et les garde dans sa pile de gain. On complète avec trois cartes de la pioche. S'il n'est pas bon, le jeu continue mais celui qui s'est trompé doit attendre qu'un autre joueur trouve un set avant de pouvoir rejouer. Le gagnant est celui qui a trouvé le plus de sets. Alternativement, le jeu peut se jouer tout seul, en essayant de trouver les sets le plus rapidement possible. On joue alors face à la montre.

Par exemple, combien de sets pouvez-vous trouver dans les douze cartes suivantes ?



La réponse est 6, et voici les 6 solutions :



Notez qu'il peut arriver que 12 cartes ne contiennent aucun set. Dans ce cas, on doit ajouter 3 nouvelles cartes sur la table et chercher dans les 15 cartes alors présentes. Lorsque l'on trouve le premier set, on ne rajoutera alors pas de nouvelles cartes (sauf s'il n'y a à nouveau aucun set sur la table). En fait, on peut montrer que :

- Il existe des ensembles de 20 cartes qui ne contiennent aucun set, mais tout ensemble de 21 cartes contient au moins un set. Voir [ici](#).
- La probabilité qu'un ensemble de 12 cartes tirées aléatoirement ne contienne aucun set est d'environ 1/30, et la probabilité qu'un ensemble de 15 cartes tirées aléatoirement ne contienne aucun set est d'environ 1/2500.
- Au cours d'une partie, ces probabilités tombent respectivement à 1/16 et 1/100 environ. Voir [ici](#) ou

[là.](#)

Dans notre version, on pourra donc toujours supposer que lorsque l'on a 15 cartes sur la table, elles contiennent toujours un set. Voir [ici](#) et [ici](#) ou [là](#) si vous êtes intéressés par la combinatoire de ce jeu.

Ce jeu a été créé par Marsha Falco et auto-édité en 1991. Il est maintenant édité par une grande maison d'édition de jeux de société.

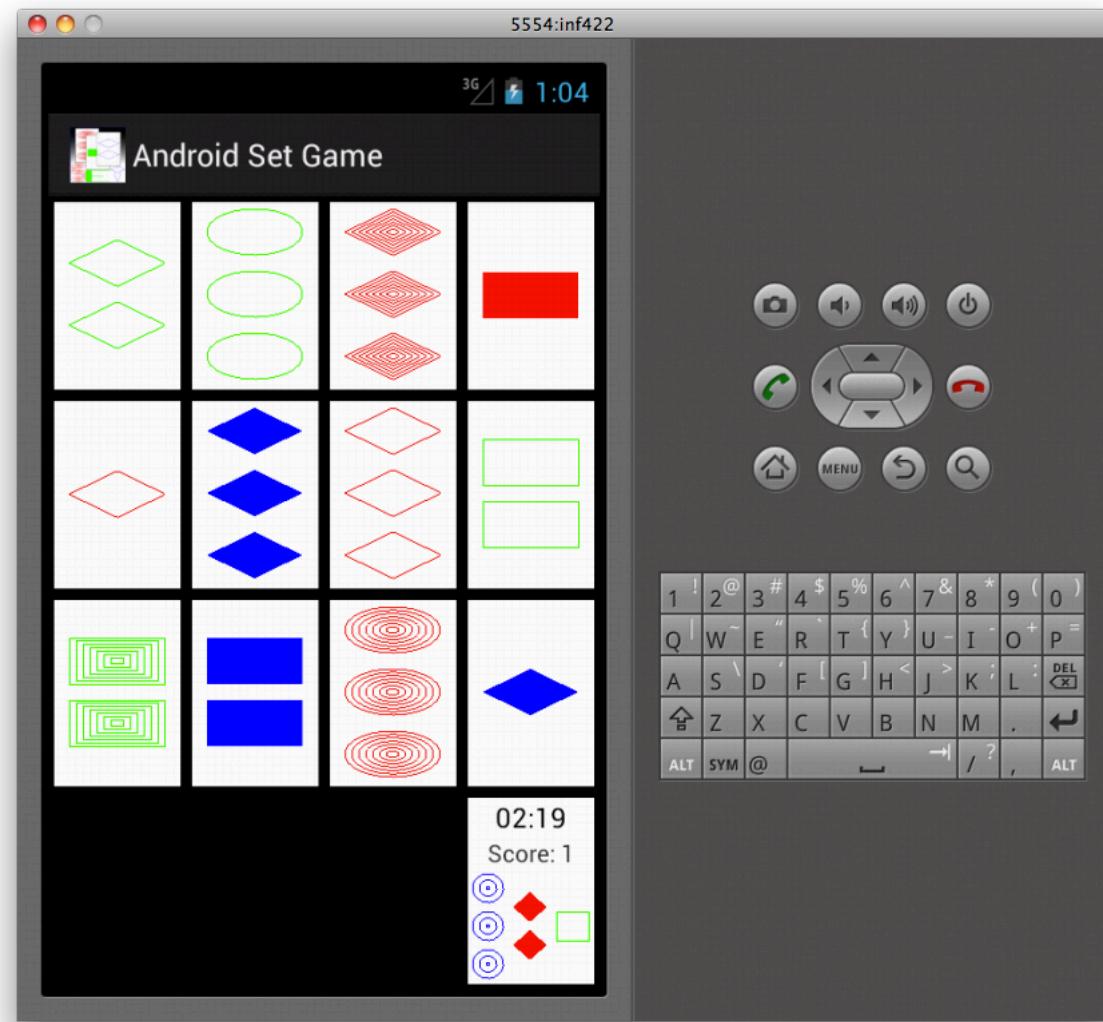
2. Description du projet

Le but du projet est de développer une application multi-joueurs du jeu Set!

2.1. Application à un joueur

Dans un premier temps, vous vous contenterez de développer une application pour un joueur seul. L'objectif ici est de manipuler l'interface graphique d'android ou une interface graphique de votre choix, comme [SWING](#), et les threads utiles pour cela.

Voici à quoi doit ressembler votre application, en tout cas sous Android, ici. Notez la présence d'un panneau de contrôle en bas à droite qui contient le temps écoulé depuis le début de la partie, le score, et le dernier set attrapé.

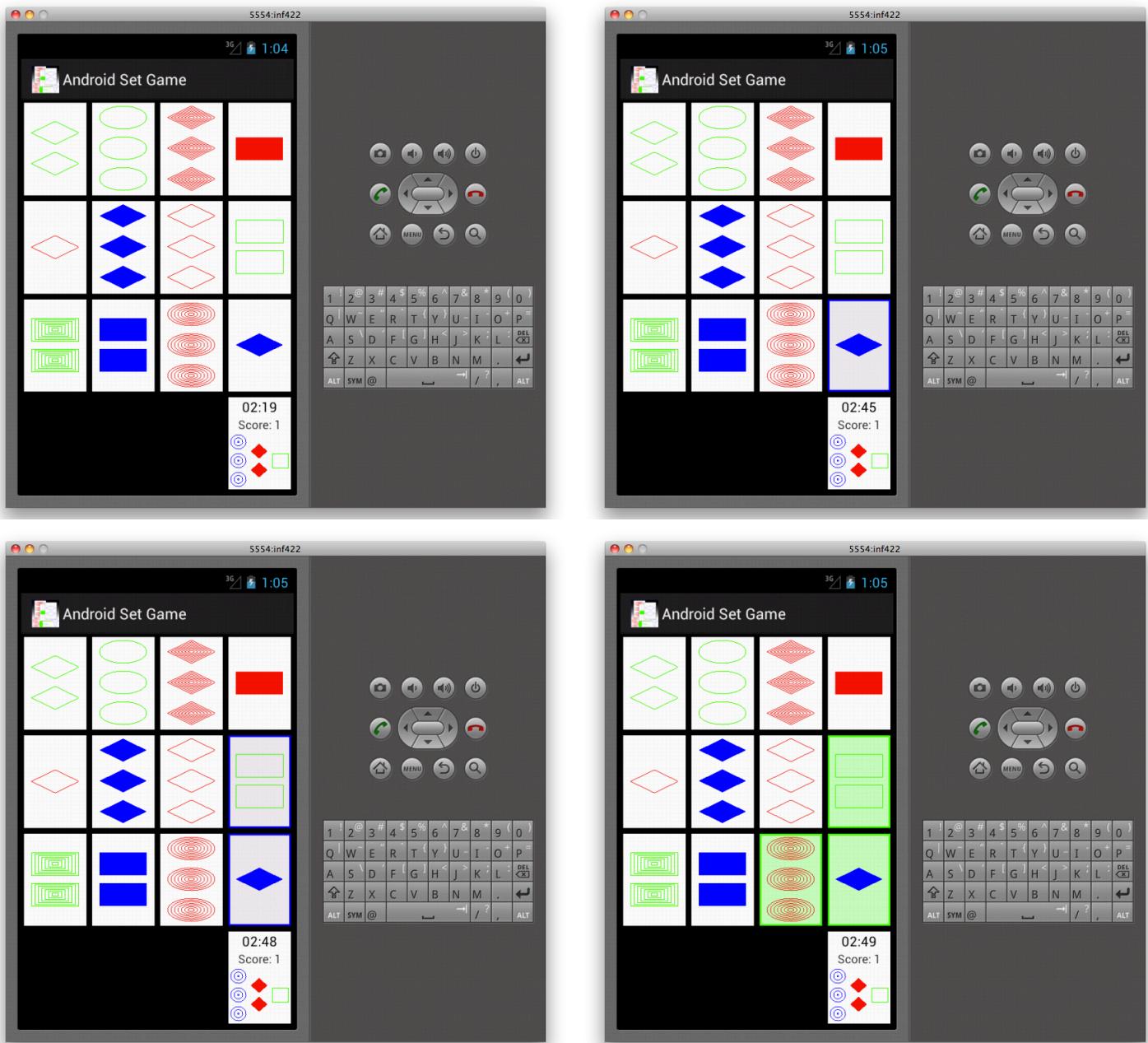


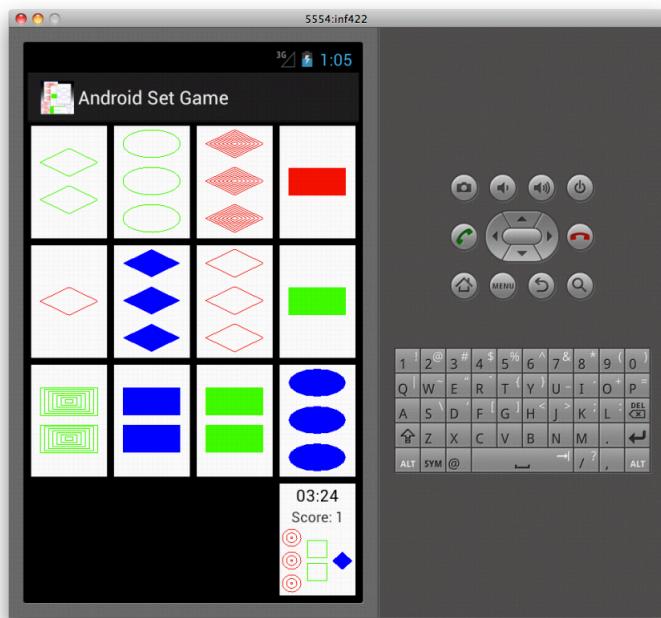
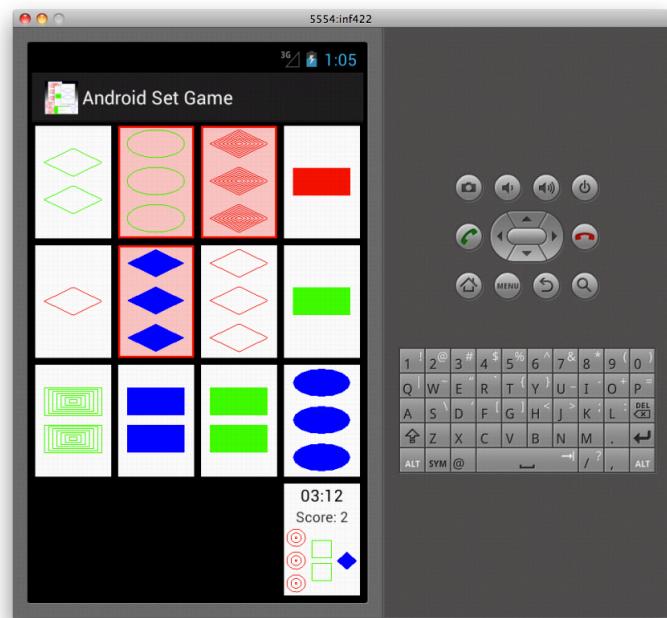
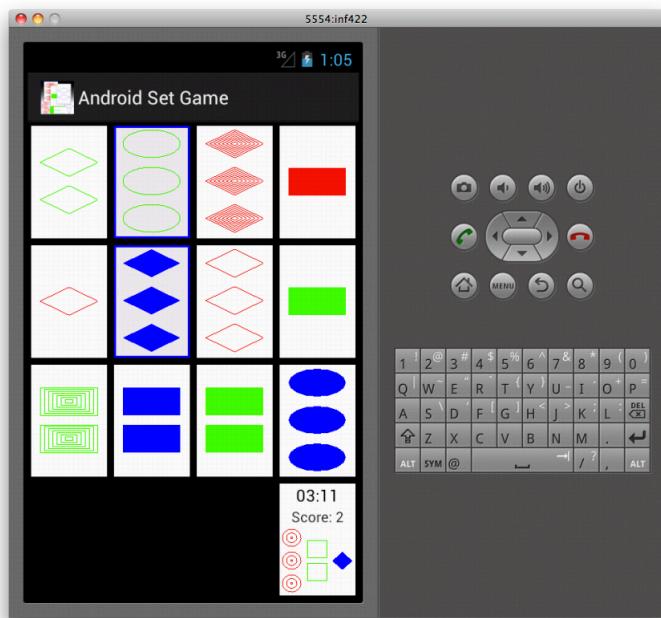
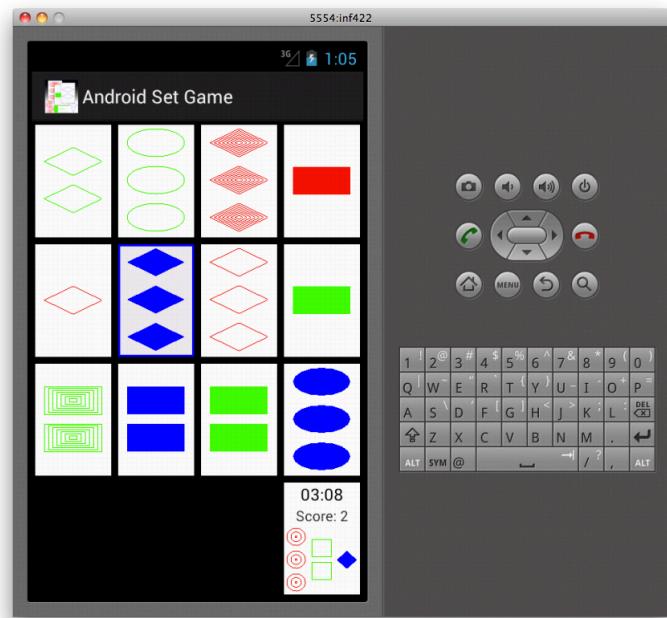
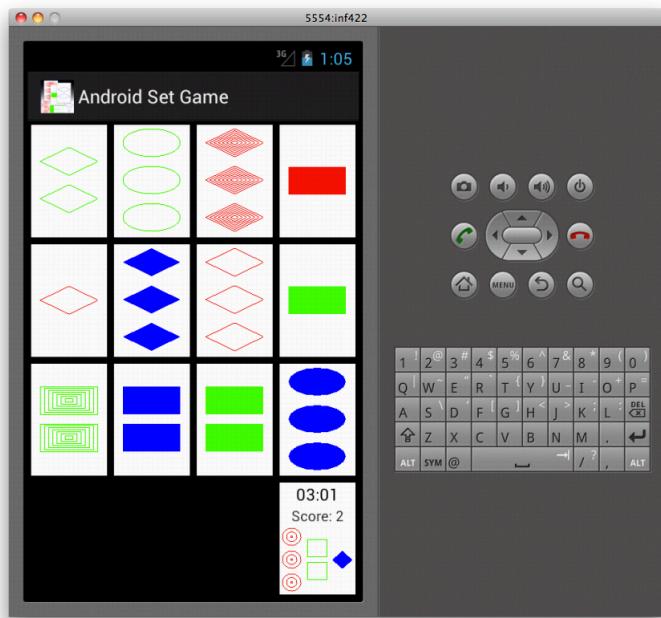
Lors du jeu, votre application doit typiquement se comporter comme suit (voir les illustrations ci-dessous).

- À tout moment, le joueur doit pouvoir sélectionner et désélectionner des cartes. Les cartes sélectionnées apparaissent sur fond bleu.
- Quand trois cartes sont sélectionnées, le jeu teste si ces trois cartes forment ou non un set.
 - Si oui, ces cartes apparaissent sur fond vert pendant quelques instants, puis vont se placer dans le panneau de contrôle comme dernier set attrapé. Le score est incrémenté, et de nouvelles cartes sont replacées.
 - Sinon, ces cartes apparaissent sur fond rouge pendant quelques instants et restent à leur place. Le score est décrémenté.

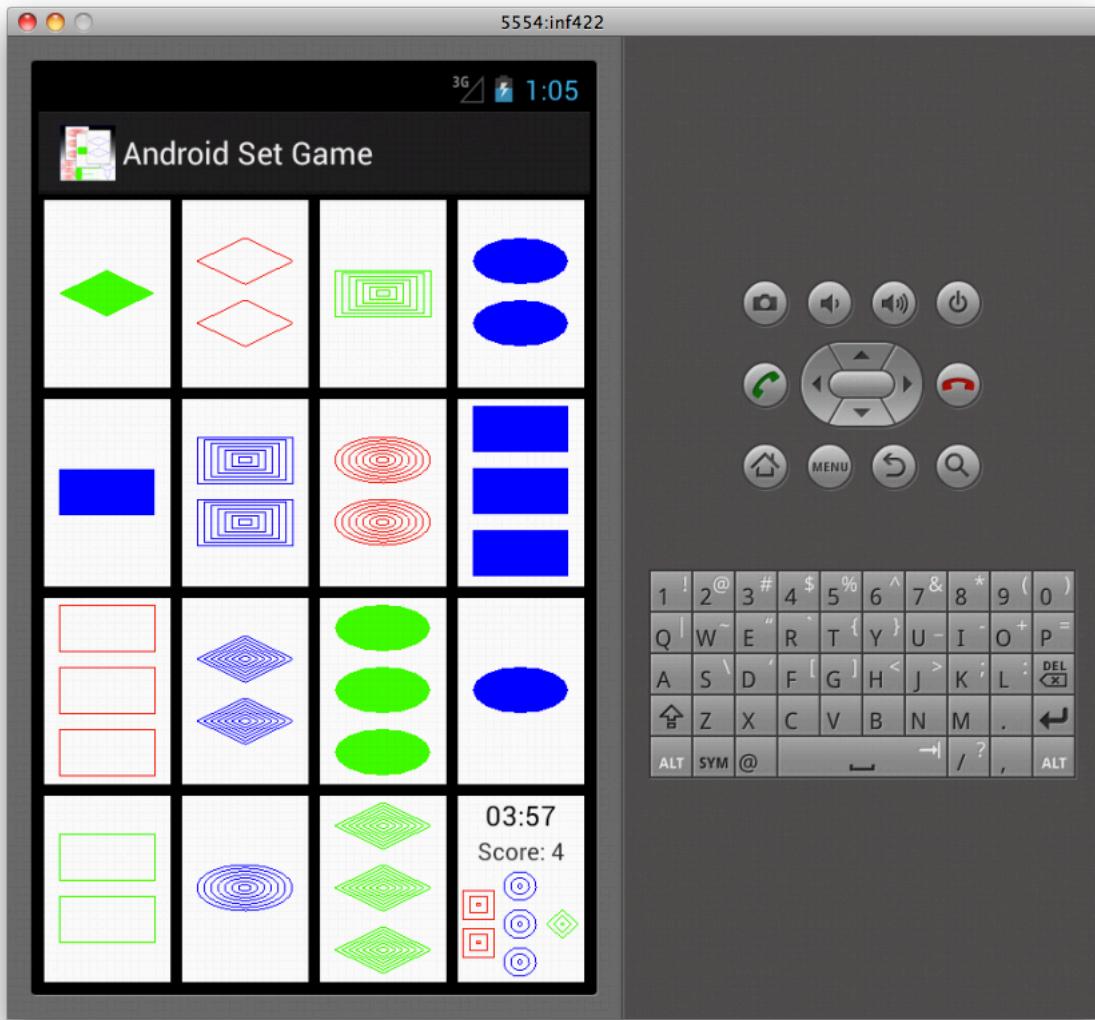
Dans les deux cas, toutes les cartes du plateau ont été désélectionnées.

On illustre ci-dessous le déroulement typique du jeu.





Comme on l'a dit plus haut, il se peut qu'on doive rajouter trois cartes sur le plateau :



Il faudra donc penser à tester chaque fois qu'on remet des cartes s'il existe bien un set dans les cartes qui sont sur le plateau.

2.2. Multi-joueurs

Vous devez développer une version multi-joueur du jeu Set!. Le plus simple sera d'utiliser un serveur central qui tourne sur votre ordinateur. Sous Android, on pourra s'inspirer du [TD 3](#) de l'ancien cours INF422 et du code du [serveur de chat](#) correspondant. Ensuite, chaque utilisateur se connecte à ce serveur central en communiquant par exemple à travers des sockets. Vous pouvez choisir d'autres options d'implémentation (un serveur sur l'un des téléphones, ...).

Vous devez définir le protocole de votre jeu. Par exemple, on pourra élaborer sur le protocole suivant :

- Chaque joueur sélectionne des cartes localement sur son téléphone.
- Lorsqu'un joueur a sélectionné trois cartes, une requête est envoyée au serveur.
- Le serveur évalue les requêtes dans l'ordre d'arrivée et détermine la première requête gagnante.
- L'évaluation d'une requête consiste à tester si les cartes sélectionnées forment un set, et le cas échéant à les retirer du jeu pour en ajouter des nouvelles. Les cartes sélectionnées par tous les autres joueurs doivent être mises à jour en conséquence.

3. Modalités

Vous pouvez préparer ce projet seul ou par binôme. Cette décision doit nous être notifiée par email avant le cours 6.

Pour nous rendre votre travail, vous devez déposer un fichier zip de votre application (ie. un zip du répertoire principal de votre application, qui contiendra donc non seulement le répertoire src, mais aussi tous les autres répertoires nécessaires à l'application, en particulier le répertoire res et le fichier AndroidManifest.xml si vous faites une version Android). Nous devons pouvoir faire tourner votre application sur notre émulateur ou machine virtuelle Android.

4. Conseils et coups de pouce

N'hésitez pas à nous envoyer un email pour prendre rendez-vous et venir nous voir si vous êtes coincés sur votre projet.

Pour créer votre application Android, suivez ce [tutoriel](#). Le plus simple est sans doute d'utiliser eclipse et de suivre les indications qu'il vous donne.

Version initiale, simple

Le code fournit en exemple comporte deux classes : [Cards](#) et [CardDrawable](#). Pour la suite, voici une liste de choses à faire, pas forcément dans cet ordre :

Afficher douze cartes sur un plateau

Il s'agit essentiellement de créer douze vues contenant chacune un objet de type CardDrawable. Un bon point de départ est la vue ImageView et sa méthode setImageDrawable(). Vous pouvez insérer douze vues dans un groupe TableLayout, ou dans des groupes LinearLayout avec des attributs layout_weight appropriés. Voyez aussi la méthode ViewGroup.getChildAt() si vous ne voulez pas nommer toutes les vues individuellement avec un ID.

Mémoriser et gérer la sélection du joueur

En répondant aux événements `onClick()` comme dans le [TD 2](#) de l'ancien cours INF422, modifiant l'état du jeu en conséquence.

Représenter les cartes sélectionnées

Pour cela, vous voudrez changer l'état de vos objets cardDrawable. Il faut modifier le code de la classe pour gérer différents couleurs de fond pour la carte selon un état (sélectionné/non sélectionné). Cet état peut alors être modifié de diverses manières, et la vue contenant la carte doit être *invalidée* afin que le nouveau dessin apparaisse.

Représenter les combinaisons gagnantes et perdantes

Il s'agit d'animations (très simples) : vous voulez afficher une transition graphique indiquant la nature de la combinaison sélectionnée, avant de désélectionner les cartes. Il y a plusieurs manières de procéder ; une façon simple est de poster des tâches à exécuter à des instants précis (p.ex. une seconde après l'instant courant) et qui modifient les vues affichées temporairement. Cf. `Handler.postDelayed()`.

Gérer le tirage de nouvelles cartes depuis une pile

Retenir quelles cartes ont déjà jouées, s'assurer que les cartes tirées contiennent au moins un *set*, et détecter les combinaisons gagnantes pour remplacer les cartes concernées.

Pour ceux qui veulent mieux comprendre le code fourni ou aller plus loin.

- Vous pouvez représenter les cartes de maintes façons différentes. Il s'agit d'encoder d'une manière ou d'une autre les quatre caractéristiques d'une carte : cela tient dans un entier (avec plusieurs codages possibles), ou bien en tant que champs d'un objet. Une classe `Card` offre une meilleure abstraction et des facilités de manipulations, mais un simple entier est plus compact en mémoire et ne nécessite pas d'allocation d'objet, coûteuse sur des systèmes plus petits comme les téléphones. De la même manière, beaucoup de données sont empaquetées dans des simples entiers dans les bibliothèques Android.
- La classe `CardDrawable` est codée de manière à réutiliser une poignée d'objets alloués une fois pour toute à la création, cela afin d'éviter la création d'une multitude d'objets à chaque dessin de la carte.
- Il y a plusieurs façons d'altérer l'état d'une carte affichée à l'écran : entre autres, une vue personnalisée (qui étend `ImageView` p.ex.) peut gérer la modification elle-même (en surchargeant la méthode `performClick()`), ou le code de l'interface principale peut modifier l'état et invalider la vue.

Approche initiale, plutôt plus difficile à suivre

On vous fournit une classe `Card` disponible dans le fichier [Card.java](#). Cette classe modélise une carte du jeu set. Elle contient trois méthodes publiques :

- son constructeur `Card(int value)` qui construit une carte à partir d'un entier entre 0 et 80. (Si nécessaire, les détails du codage des cartes sont expliqués dans la classe.)
- la méthode `boolean equals(Object o)` qui teste l'égalité des quatre caractéristiques des cartes.
- la méthode `int[] characteristics()` qui renvoie les quatre caractéristiques de la carte dans l'ordre nombre, couleur, remplissage, forme.
- la méthode `void draw(Canvas c, int width, int height)` qui dessine le contenu de la carte sur un canevas `c`, aux dimensions `width` x `height`.

Pour gérer la mise en page, vous avez deux options :

- Soit en utilisant xml, en modifiant le fichier `main.xml` et en utilisant `setContentView(R.layout.main);` dans votre classe principale. Dans le xml, vous pouvez même ajouter des objets de vos propres classes qui étendent `View`. Un exemple est donné par la classe `CardPanel` disponible dans le fichier [CardPanel.java](#). Notez comment cette classe redéfinit la méthode `void onDraw(Canvas c)`.
- Soit directement dans le code, en utilisant les classes `TableLayout`, `TableRow` et `CardPanel`, et la méthode `addView` de ces classes. Pour que la mise en page soit correcte, il faudra sans doute utiliser la classe de paramètres `TableLayout.LayoutParams`, et la méthode `setWeightSum` des classes `Layout`.

D'autres coups de pouce viendront s'ajouter ici en fonction de votre travail et intérêt pour ce projet.