Introduction

Ce deuxième travail pratique sert à comprendre ce qu'est une activité au sein d'une application. Une activité est un composant essentiel au bon fonctionnement de toute application Android. Il est primordial de comprendre son importance et son interaction avec le système notamment au travers de son cycle de vie. Avec cette connaissance, vous saurez aussi gérer le changement d'orientation du téléphone et les événements du système afin que l'utilisateur aie une expérience la plus fluide possible.

Objectifs

Voici une liste précise des objectifs à atteindre. Ces objectifs servent de guide sur la matière à savoir et permettent de cibler les recherches pendant l'étude du contenu du cours.

- Comprendre le rôle des activités,
- Comprendre le cycle de vie des activités (états et événements),
- Créer une classe indépendante,
  - Communiquer avec cette classe (modèle de conception delegate),
  - Rendre cette classe consciente du cycle de vie (Lifecycle API),
- Gérer le changement d'orientation du téléphone,
  - Sauvegarde et récupération des données,
  - Interfaces différentes pour chaque orientation,
- Comprendre le contenu et les objectifs du manifest.

Travail pratique

Le travail pratique abordé ici se nomme "Moles Whacker". Il consiste en un jeu de la taupe bien connu dans les salles d'arcade. Le but est de taper, à l'aide d'un marteau, sur un maximum de taupes, sortant de leur trou de manière aléatoire, dans un temps donné.

L'interface graphique comporte un seul écran qui est composé des composants graphiques suivants :

- Un bouton "Start" pour démarrer une partie,
- Douze boutons représentant les taupes,
- Deux labels pour indiquer le score du joueur,
- Deux labels pour indiquer le temps restant (compte à rebours).

Le jeu se déroule en 4 phases qui sont les suivantes (Figures 1 et 2) :

1. Lorsqu'une partie n'est pas en cours, les douzes taupes sont visibles, le compte-à-rebours indique zéro et le dernier score obtenu est affiché (zéro en cas de première partie).
2. Lorsqu'une partie est démarrée avec un clic sur le bouton "Start", le bouton "Start" se désactive, le compte à rebours redémarre (par ex. 30 secondes) et toutes les taupes deviennent invisibles à l'exception d'une.
3. Durant la partie, la taupe visible l'est jusqu'à ce que le joueur clique dessus. Après un clic, la taupe se cache et une autre apparaît (attention : pas la même). Le score est incrémenté et directement affiché à l'écran. Ces actions se répètent jusqu'à la fin du compte à rebours.
4. Lorsque le compte à rebours est terminé, toutes les taupes se rendent visibles, le bouton "Start" se réactive et le joueur peut recommencer une nouvelle partie (phase 1).

Le jeu doit fonctionner lorsque le téléphone est en orientation portrait (vertical) et paysage (horizontal). Il est très important que l'utilisateur retrouve les mêmes informations à l'écran avant et après le changement d'orientation. Le placement des composants doit s'adapter en dépendant de l'orientation (Figures 3 et 4). Si l'utilisateur quitte

l'application au milieu d'une partie, celle-ci doit automatiquement se mettre en pause et se retrouver dans le même état lorsqu'elle est reprise.

Une classe gérant le compte à rebours est mise à disposition. Elle suit le modèle de conception "singleton" qui permet d'avoir une seule et unique instance d'une classe. Toute la logique du compte à rebours est implémentée et il suffit d'appeler les méthodes publiques. Finalement, cette classe doit être modifiée avec l'API Lifecycle pour qu'elle soit consciente du cycle de vie de l'activité qui l'utilise.

Cependant, avant de commencer le travail pratique, il est demandé de réaliser une série d'exercices afin que vous puissiez implémenter le jeu avec plus d'aisance et une meilleure compréhension du cycle de vie des activités.

Cette donnée est aussi disponible sous forme de codelab à l'adresse suivante (connexion à GitLab requise) : https://mobapp.pages.forge.hefr.ch/

Captures d'écran

Voici un exemple du résultat final qui peut servir d'inspiration pour le travail pratique. Il n'est pas nécessaire que l'interface ressemble exactement à ces captures d'écran : le plus important est que l'application soit fonctionnelle et qu'elle respecte la description et les contraintes du travail.

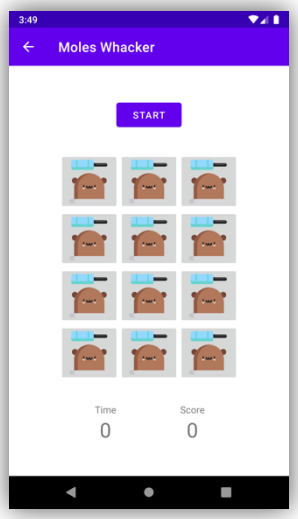

Figure 1 – Accueil (portrait)

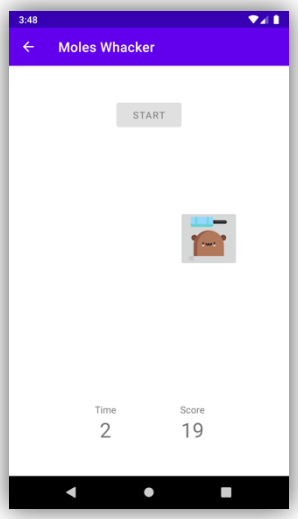

Figure 2 – Partie en cours (portrait)

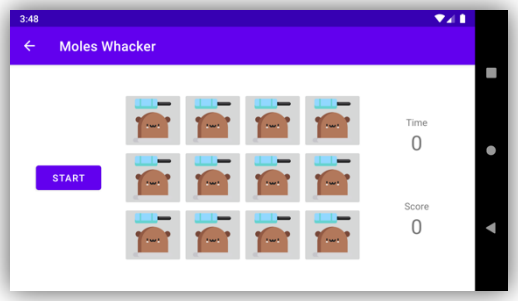

Figure 3 – Accueil (paysage)

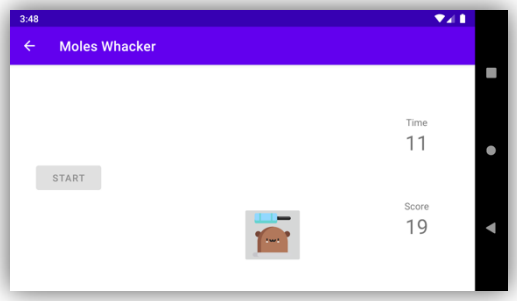

Figure 4 – Partie en cours (paysage)

Contraintes

Voici une liste de contraintes à respecter obligatoirement. Ces contraintes forcent à utiliser certains points précis de la thématique et permettent de bien la comprendre :

- L'interface graphique doit avoir tous les éléments cités dans la description du travail pratique,
- Le comportement de l'application doit être identique à celui décrit dans la description du travail pratique,
- L'application doit fonctionner en orientation portrait et paysage,
- L'API Lifecycle doit être utilisée pour gérer le compte-à-rebours,
- Les composants architecturaux tels que ViewModel et LiveData sont interdits.

Conseils

Afin de vous éviter des problèmes lors de la réalisation de votre travail pratique, nous vous conseillons d'effectuer les points suivants :

- Il est recommandé de créer un projet avec le modèle "Empty Activity",
- Il est recommandé des réaliser les exercices directement dans le même projet,
- Attention à ne pas oublier l'icône d'application.

Ressources disponibles

Afin de permettre une concentration optimale sur les thématiques du travail pratique, les ressources suivantes sont à votre disposition sur Cyberlearn :

- Icônes externes à Android Studio (.svg),
- Classe qui gère le compte à rebours (.java),

Documentation

Voici quelques liens utiles contenant dans un premier temps beaucoup d'informations concernant les différentes thématiques abordées. Il est conseillé d'accéder à ces liens dans l'ordre dans lequel ils sont listés afin de faciliter la compréhension de la matière.

- Implémentation de l'API Lifecycle (jusqu'à la section "LifeCycleOwner" comprise) : https://developer.android.com/topic/libraries/architecture/lifecycle
- Introduction aux activités (complément à la théorie) : https://developer.android.com/guide/components/activities/intro-activities
- Cycle de vie des activités (complément à la théorie) : https://developer.android.com/guide/components/activities/activity-lifecycle
- Aperçu du manifest (complément à la théorie) : https://developer.android.com/guide/topics/manifest/manifest-intro
- Sauvegarde des données lors d'évènements système (complément à la théorie) : https://developer.android.com/guide/components/activities/activity-lifecycle#saras
- Créer des layouts alternatifs (complément à la théorie) : https://developer.android.com/training/multiscreen/screensizes#alternative-layouts

Exercices

Voici une série d'exercices qu'il est demandé de faire et d'expliquer dans le rapport. Ces exercices ont pour but de faciliter la compréhension de certains points importants de la thématique.

Dans un projet Android Studio que vous aurez préalablement créé, surchargez les différentes méthodes du cycle de vie d'une activité. Ces méthodes sont listées aux deux liens suivants. Vous devriez vous retrouvez avec un

total de 8 méthodes. Dans chaque méthode, utilisez le logcat pour savoir quand ces méthodes sont appelées. Effectuez ensuite les scénarios décrits et rapportez l'ordre d'appel de ces méthodes. Quittez complètement l'application entre chaque scénario.

Liens des méthodes :

- https://developer.android.com/guide/components/activities/activity-lifecycle#lc
- https://developer.android.com/guide/components/activities/activity-lifecycle#saras

Scénarios :

1. Ouvrez l'application, quittez-la avec le bouton "back" physique puis ouvrez-la à nouveau,
2. Ouvrez l'application, quittez-la avec le bouton "home" puis ouvrez-la à nouveau,
3. Ouvrez l'application puis changez l'orientation du téléphone.

Rendu

Il est demandé de rendre sur la page Cyberlearn du cours un dossier compressé (.zip) nommé "tp02-<nom du groupe>" contenant les éléments suivants :

- Rapport concis (.pdf) avec les chapitres (nommé "tp02-report-<nom du groupe>")
  - Introduction
  - Exercices
  - Problèmes rencontrés
  - Conclusion
- Code du projet (.zip) nettoyé (nommé "tp02-code-<nom du groupe>")
- Le fichier installable (.apk) de l'application (nommé "tp02-app-<nom du groupe>")

N'hésitez pas à décrire votre ressenti par rapport à ce travail pratique. Par exemple, si la quantité de travail, la durée, la complexité, l'intérêt qu'il sucite vous ont semble adaptés ou non et si possible expliquer pourquoi. Ceci permet d'adapter le travail pratique en cas de problème.

Le rendu du devoir est fixé au **Jeudi 14.10.21 à 23h59** et est à rendre par groupe. Une personne peut déposer le travail pour les deux membres du groupe.

Checklist

Voici une checklist pour éviter d'oublier certains aspects du travail demandé ce qui pourrait enlever des points lors de l'évaluation :

- ☐ Réalisation des exercices,
- ☐ Création d'une interface graphique portrait,
- ☐ Création d'une interface graphique paysage,
- ☐ Implémentation de la logique du jeu,
- ☐ Modification de la classe "GameTimer" avec l'API Lifecycle,
- ☐ Gestion du changement d'orientation (sauvegarde/récupération des données),
- ☐ Gestion de la mise en pause du jeu (sauvegarde/récupération des données),
- ☐ Créer une icône d'application,
- ☐ Rédaction d'un rapport concis avec ses 4 chapitres.