

Preuves compétences projet Android

Maël CHAUMONT

Jaden COUCHOT

1 - Je sais décrire le contexte de mon application, pour que n'importe qui soit capable de comprendre à quoi elle sert.

Dans le dossier « documentation », voir fichier Contexte.pdf

2 - Je sais concevoir et décrire un diagramme de cas d'utilisation pour mettre en avant les différentes fonctionnalités de mon application.

Voir fichier CasUtilisation.pdf

3 - Je sais concevoir un diagramme UML de qualité représentant mon application.

Voir fichier DiagrammeClasses.pdf

4 - Je sais décrire mon diagramme UML en mettant en valeur et en justifiant les éléments essentiels.

La description de ce diagramme se trouve dans DescDiagrammeClasses.pdf

5 - Je sais utiliser les Intent pour faire communiquer deux activités.

```
public void settingsClick(View view) {
    Intent monIntent = new Intent( packageContext: this, HighscoreActivity.class);
    startActivity(monIntent);
}

public void rulesClick(View view) {
    Intent monIntent = new Intent( packageContext: this, RulesActivity.class);
    startActivity(monIntent);
}
```

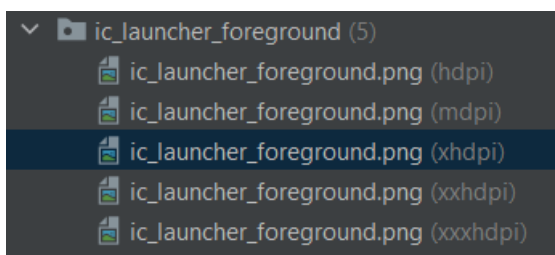
Voici une partie du code de la classe MainActivity : selon la méthode appelée, l'Intent enverra soit vers la classe HighscoreActivity soit vers RulesActivity.

6 - Je sais développer en utilisant le SDK le plus bas possible.

```
defaultConfig {  
    applicationId "com.example.myapplication"  
    minSdk 16  
    targetSdk 31  
    versionCode 1  
    versionName "1.0"  
  
    testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"  
}
```

Dans le build.gradle : le SDK minimum pour que notre application fonctionne est le SDK 16.

7 - Je sais distinguer mes ressources en utilisant les qualifier



Ici on peut voir le drawable du premier plan de l'icône de notre application (un poisson avec hameçon). Nous l'avons généré en plusieurs formats grâce à différents qualifieurs (hdpi, mdpi, xhdpi, ...)

8 - Je sais faire des vues xml en utilisant layouts et composants adéquats

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
  
    <ImageView  
        android:id="@+id/imageView"  
        android:layout_width="0dp"  
        android:layout_height="0dp"  
        android:adjustViewBounds="false"  
        android:scaleType="centerCrop"  
        android:scrollX="-100dp"  
        android:scrollY="0dp"  
        app:layout_constraintBottom_toBottomOf="parent"
```

Nous avons utilisé de nombreux fichiers .xml dans notre dossier layout. Voici notre fichier home.xml, qui comme beaucoup de nos fichiers possède comme élément racine un ConstraintLayout (éléments placés les uns par rapport aux autres selon des contraintes, par exemple la ligne *app:layout_constraintBottom_toBottomOf='parent'* signifie que le bas de l'ImageView sera collé au bas de son élément parent, ici le ConstraintLayout).

9 - Je sais coder proprement mes activités, en m'assurant qu'elles ne font que relayer les évènements

```
@Override
protected void onDestroy() {
    super.onDestroy();
    theGM.stopGame();
}
```

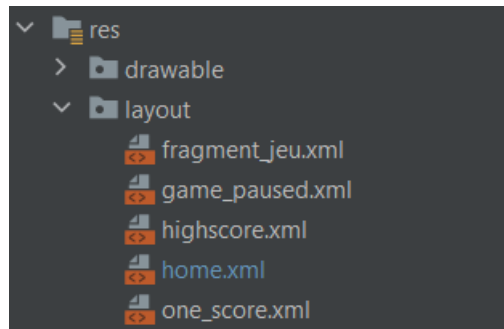
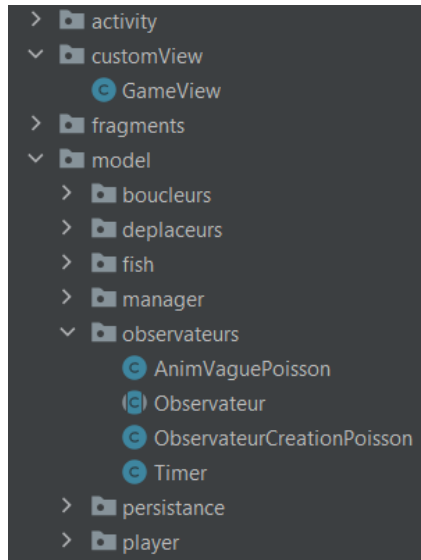
```
@Override
protected void onPause() {
    super.onPause();
    game_paused.setVisibility(View.VISIBLE);
    theGM.stopGame();
}
```

Dans l'activité GameActivity, on peut voir que quand cette activité est mise en pause ou détruite c'est la méthode stopGame() de notre classe GameManager qui est appelée. L'arrêt d'une partie ne se fait donc pas directement dans notre Activité.

10 - Je sais coder une application en ayant un véritable métier

Cette application répond bien à un besoin spécifique : celui de passer du temps et apprécier de jouer au jeu. Ce jeu est donc bien une application métier.

11 - Je sais parfaitement séparer vue et modèle



Ma CustomView, mes fragments et mes activités sont bien placés en dehors de notre Modèle. Les layouts et drawables utilisés pour la vue sont situés dans le dossier de ressources res.

12 - Je maîtrise le cycle de vie de mon application

```
@Override
protected void onDestroy() {
    super.onDestroy();
    theGM.stopGame();
}

@Override
protected void onResume() {
    super.onResume();
    Button btnContinue = (Button) findViewById(R.id.btn_continue);
    bindButtonContinue(btnContinue);
}

@Override
protected void onPause() {
    super.onPause();
    game_paused.setVisibility(View.VISIBLE);
    theGM.stopGame();
}
```

Le cycle de vie de notre jeu est géré : en effet, notre activité de jeu gère les appels à onCreate(), onPause(), onResume(), onDestroy() et délègue ensuite l'action à effectuer au GameManager. Par exemple, onPause stoppera le jeu et fera apparaître à l'écran un menu de pause.

13 - Je sais utiliser le findViewById à bon escient

GameActivity

V

```
game_paused = findViewById(R.id.game_paused);
```

```
@Override  
protected void onPause() {  
    super.onPause();  
    game_paused.setVisibility(View.VISIBLE);  
    theGM.stopGame();  
}
```

viewcustomjeu.xml

V

```
<!-- include permet de mettre une vue par-dessus  
<include  
    android:id="@+id/game_paused"  
    layout="@layout/game_paused"  
    android:layout_width="758dp"  
    android:layout_height="389dp"  
    android:visibility="invisible" />
```

Nous avons utilisé le findViewById à plusieurs endroits pour récupérer des éléments du xml, par exemple ici dans GameActivity où nous avons récupéré l'élément avec l'id « game_paused » dans le xml, pour en changer la visibilité plus tard.

14 - Je sais gérer les permissions dynamiques de mon application

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

Des permissions d'accès aux fichiers afin d'enregistrer les scores sont demandées dans le manifeste.

15 - Je sais gérer la persistance légère de mon application

```
Intent intent = getIntent();  
myNickname = intent.getExtras().getString( key: "nickName");
```

Nous avons utilisé la persistance légère pour faire passer le pseudo du joueur depuis l'activité d'accueil jusqu'à l'activité de jeu pour qu'il soit affiché.

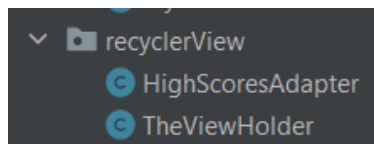
```
getIntent().putExtra( name: "monGameManager", theGM);
FragmentManager fm = getSupportFragmentManager();
Fragment fragment = new MyFragment();
fm.beginTransaction().replace(R.id.frag, fragment).commit();
```

Nous avons aussi utilisé la persistance légère plus loin, pour faire passer le GameManager actuel à notre fragment et donc permettre l'affichage du score et pseudo du joueur.

16 - Je sais gérer la persistance profonde de mon application

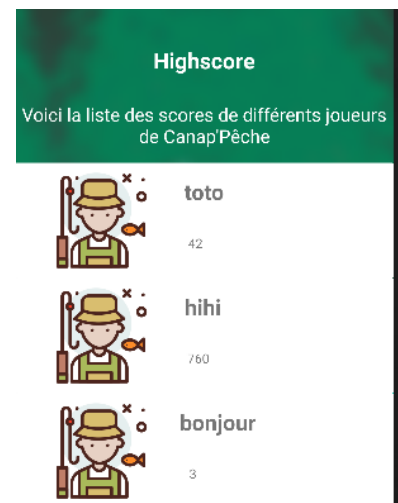
Nous avons tenté de faire une persistance profonde pour pouvoir enregistrer les scores, mais qui ne marche pas pour le moment.

17 - Je sais afficher une collection de données

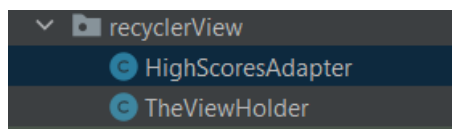


Nous avons implémenté une RecyclerView et son adaptateur pour afficher notre collection de scores.

Elle est visible en cliquant sur le bouton HIGHSCORES depuis le menu d'accueil.



18 - Je sais coder mon propre adaptateur



Nous avons dû créer un adaptateur appelé HighScoresAdapter afin de faire fonctionner notre RecyclerView des Highscores. (on peut y accéder en cliquant sur le bouton HIGHSCORE de la page d'accueil).

19 - Je maîtrise l'usage des fragments

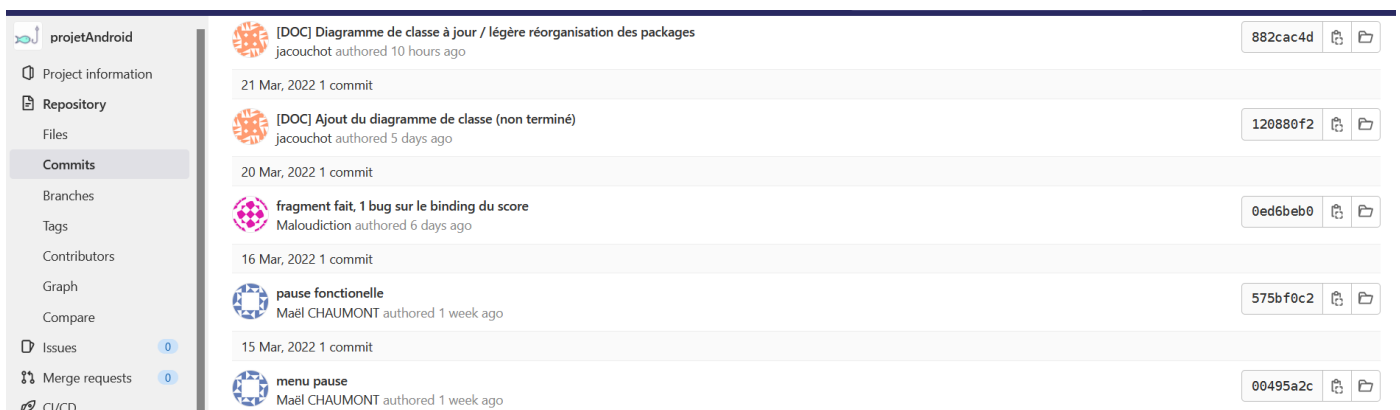
```
public class MyFragment extends Fragment {
    private GameManager monGameManager;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable
        Intent i = getActivity().getIntent();
        monGameManager = (GameManager) i.getSerializableExtra( name: "monGameManager");

        FragmentJeuBinding fragmentJeuBinding = DataBindingUtil.inflate(inflater, R.layout.fragment_jeu,
```

Pour notre vue de jeu, nous avons créé un fragment appelé MyFragment, qui s'affiche en bas de l'écran et qui contient le pseudo du joueur ainsi que son score actuel.

20 - Je maîtrise l'utilisation de Git



Nous avons utilisé le Gitlab de l'IUT pour gérer notre projet et en push de nouvelles versions.

21 - Je sais développer une application sans utiliser de librairies externes.

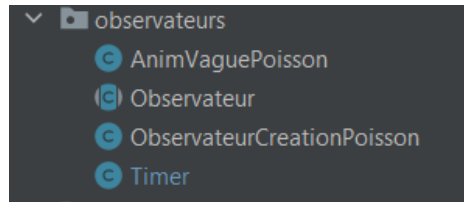
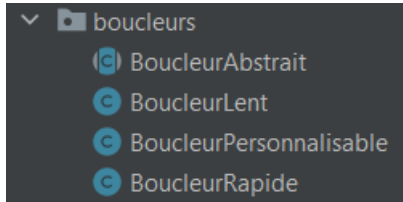
```
dependencies {

    implementation 'androidx.appcompat:appcompat:1.4.1'
    implementation 'com.google.android.material:material:1.5.0'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
}
```

Notre application ne possède aucune dépendance vers des librairies autres que celles déjà présentes à la création du projet.

22 - Je sais développer une application publiable sur le store.

23 - Je sais développer un jeu intégrant une boucle de jeu threadée observable.



Notre jeu possède un système de boucleurs lancés par 2 threads, qui notifient régulièrement les différents observateurs qui y sont attachés.

```
public void startNewGame() {
    vP = new VaguePoissons( nbPoissons: 6);
    lePecheur.setScorePecheur(0);
    animatorVP = new AnimVaguePoisson( gm: this);
    boucleurRapide = new BoucleurRapide(millisSleepBoucleurRapide, vP, animatorVP,
    boucleurLent = new BoucleurLent(millisSleepBoucleurLent, vP);
    thread1 = new MyThread(boucleurRapide);
    thread2 = new MyThread(boucleurLent);
    thread1.start();
    thread2.start();
}
```

Ces threads sont lancés au démarrage du jeu, et seront stoppés si l'activité de jeu reçoit un onPause() ou onDestroy().

24 - Je sais développer un jeu graphique sans utiliser de SurfaceView

```
public class GameView extends View {

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        Rect imageBounds = canvas.getClipBounds();

        mCustomImage.setBounds(imageBounds);
        mCustomImage.draw(canvas);
    }
}
```


Aucune `SurfaceView` n'a été utilisée puisque notre vue de jeu est la classe `GameView`, il s'agit d'une custom view qui agit comme un canvas grâce à sa méthode `onDraw()`.