Perret Louis
Mathilde Orillon
B3-A3

Preuves

Documentation

Voir fichier documentation : doc/Documentation_Ultimate.pdf

Code

Je sais utiliser les Intent pour faire communiquer deux activités.

```
/**
 * Lance un combat
 */
public void lancerCombat(){
    Intent intent = new Intent( packageContext: this,FenetreCombat.class);
    intent.putExtra( name: "largeurEcran",largeurEcran);
    intent.putExtra( name: "hauteurEcran", hauteurEcran);
    startActivity(intent);
    finish();
}
```

Figure : Méthode lancerCombat de FenetreJeu

```
@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.fenetre_combat);
    manager = ((App) getApplication()).getManager();
    //On récupère la largeur et la hauteur de la fenêtre
    largeurEcran = getIntent().getIntExtra( name: "largeurEran",getResources().getDisplayMetrics().widthPixels);
    hauteurEcran = getIntent().getIntExtra( name: "hauteurEran",getResources().getDisplayMetrics().heightPixels);
```

Figure : Extrait du onCreate de fenetreCombat

Dans cet exemple, nous utilisons un intent pour faire communiquer l'activité *FenetreJeu* avec l'activité *FenetreCombat* par exemple.

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

Le jeu a été réalisé en SDK 16 ce qui permet de toucher 100% des téléphones d'aujourd'hui.

Je sais distinguer mes ressources en utilisant les qualifier

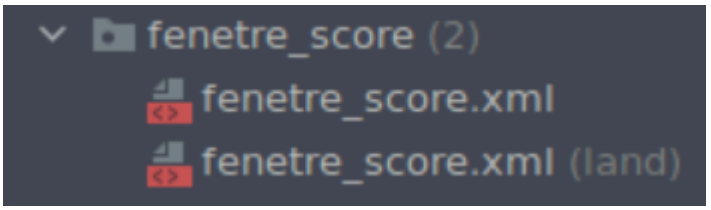

Figure : Extrait de notre package layout/fenetre_score

Nous utilisons un qualifier dans ce cas-ci pour distinguer la vue en format portrait du format paysage par exemple.

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

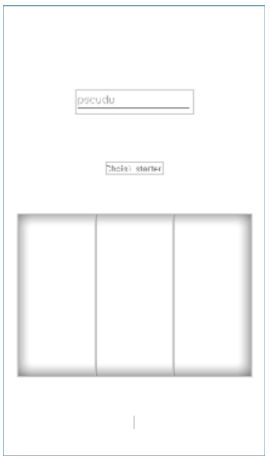

Figure : Notre vue FenetreSelection sous le designer de android studio

Exemple avec notre vue fenetre_selection. Pour pouvoir positionner les éléments comme on le souhaitait nous avons utilisé différents layouts comme un LinearLayout pour pouvoir afficher horizontalement nos éléments.

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

Notre classe FenetreScore fait appel au manager pour pouvoir afficher la liste des joueurs, ce n'est pas elle qui connaît la liste.

Je sais coder une application en ayant un véritable métier

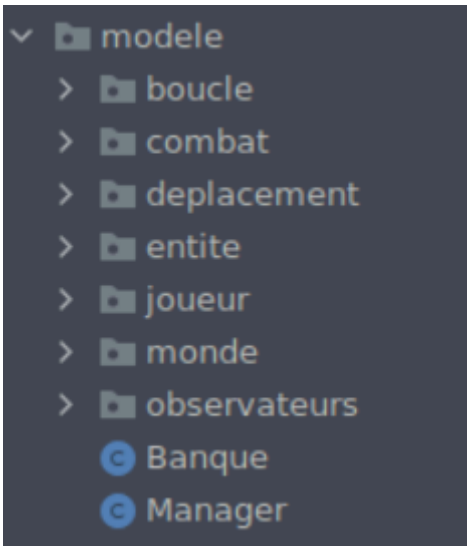

Figure : Notre modèle

Vous en avez une vision plus détaillée dans notre documentation

Je sais parfaitement séparer vue et modèle

Notre modèle n'a aucune référence vers la vue. Seule vue à une référence vers le modèle.

Je maîtrise le cycle de vie de mon application

Nous avons placé stratégiquement chaque recherche, affichage etc soit dans le onCreate, soit dans le onStart, soit dans le onResume afin de parer à toutes éventualités lors du gameplay.

Je sais utiliser le findViewById à bon escient

```
public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
    Bundle bundle = this.getArguments();
    ImageView image = view.findViewById(R.id.imageFragment);
    Button bouton = view.findViewById(R.id.buttonFragment);
```

Figure : onViewCreated de FragmentStarter

Nous utilisons le **findViewById** dans notre fragment pour pouvoir récupérer les 2 éléments de celui-ci, le bouton et l'image, pour pouvoir les changer en fonction des 3 starters. Sans cela nous n'aurions pas pu afficher les 3 différents starter. De plus, dans nos autres vues, nous utilisons le findViewById qu'une seule fois pour chaque élément dans le onCreate de ces dernières.

Je sais gérer les permissions dynamiques de mon application

Nous n'avons pas eu besoin de gérer de permissions dynamiques.

Je sais gérer la persistance légère de mon application

Sur le fenetre_selection nous avons utilisé de la persistance légère. En effet le joueur doit rentrer un pseudo avant de lancer la partie et son pseudo soit toujours apparaître sur la *TextInputEditText* quand il met son téléphone en mode paysage. Pour ce faire nous sommes passés par les éléments xml, et même si ces derniers sont recréés quand l'écran tourne par exemple, leur id permettent de leur faire garder leur ancienne valeur.

Je sais gérer la persistance profonde de mon application

Nous faisons de la persistance lourde avec notre package *persistance.*

Je sais afficher une collection de données

Nous utilisons une RecyclerView dans notre code behind FenetreScore pour afficher la liste des joueurs ayant déjà jouer au moins une fois au jeu

Je sais coder mon propre adaptateur

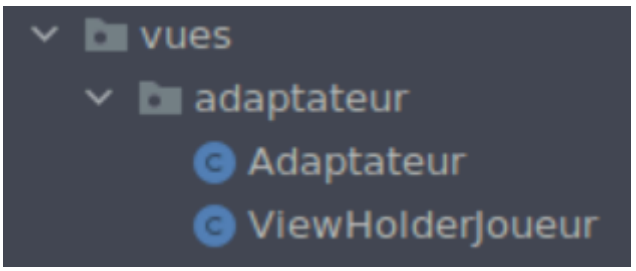

Figure : Notre package adaptateur

Notre adaptateur se trouve dans notre package *adaptateur.* Il nous a permis d'afficher la liste des joueurs.

Je maîtrise l'usage des fragments

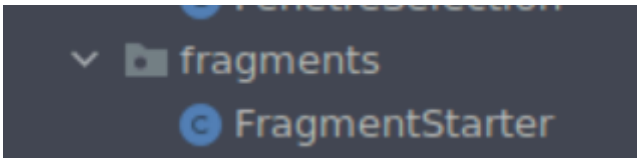

Figure : Extrait de notre package fragments

```
//manager fragment, je donne les bundle en argument
getSupportFragmentManager().beginTransaction()
        .setReorderingAllowed(true)
        .add(R.id.fragment_gauche, FragmentStarter.class, bgauche)
        .commit();

getSupportFragmentManager().beginTransaction()
        .setReorderingAllowed(true)
        .add(R.id.fragment_milieu, FragmentStarter.class, bmilieu)
        .commit();

getSupportFragmentManager().beginTransaction()
        .setReorderingAllowed(true)
        .add(R.id.fragment_droit, FragmentStarter.class, bdroit)
        .commit();
```

Figure : Extrait du code behind FenetreSelection

Nous utilisons un Fragment pour répéter 3 fois le même élément. En effet sur *FenetreSelection* le joueur peut choisir entre 3 starter différents et nous trouvons qu'utiliser un fragment était une bonne solution.

Je maîtrise l'utilisation de Git

Nous avons utilisé 2 branches avec Git durant notre projet : une branche main nous permettait de garder une version stable de l'application, puis une branche dev nous permettait de poursuivre son développement. Puis lorsque des avancements conséquents ont été fait et que l'application était stable sur la branche dev, nous avons merge pour pouvoir update la branche main.

Application

Je sais développer une application sans utiliser de librairies externes

Nous n'avons pas utilisé de librairies externes.

Je sais développer une application publiable sur le store

Nous avons envoyé un mail à M. Bouhours dimanche 27/03 au matin. Nous espérons qu'elle sera publiée lorsque vous lirez ceci.

Je sais développer un jeu intégrant une boucle de jeu threadée observable

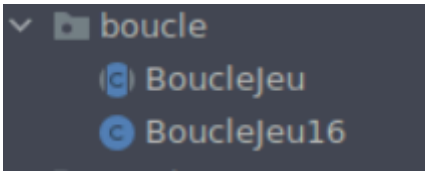

Figure : Extrait de notre package boucle

```
public void update() {
    fenetre.runOnUiThread(() -> { //Equivalent du platform runlater en javafx
        if (manager.isDebutCombat()) {
            manager.terminerBoucleJeu();
            fenetre.lancerCombat();
        }
        else {
            fenetre.updatePosition();
        }
    });
}
```

Figure : Extrait de notre classe ObservateurBoucleVue

Nous avons intégré à notre jeu une boucle de jeu observable utilisant un thread pour permettre le déplacement du personnage côté modèle avec notre ObservateurBoucle puis côté vue avec notre ObservateurBoucleVue.

Je sais développer un jeu graphique sans utiliser de SurfaceView

Nous n'avons à aucun moment utilisé de SurfaceView au sein de nos différents layout xml.