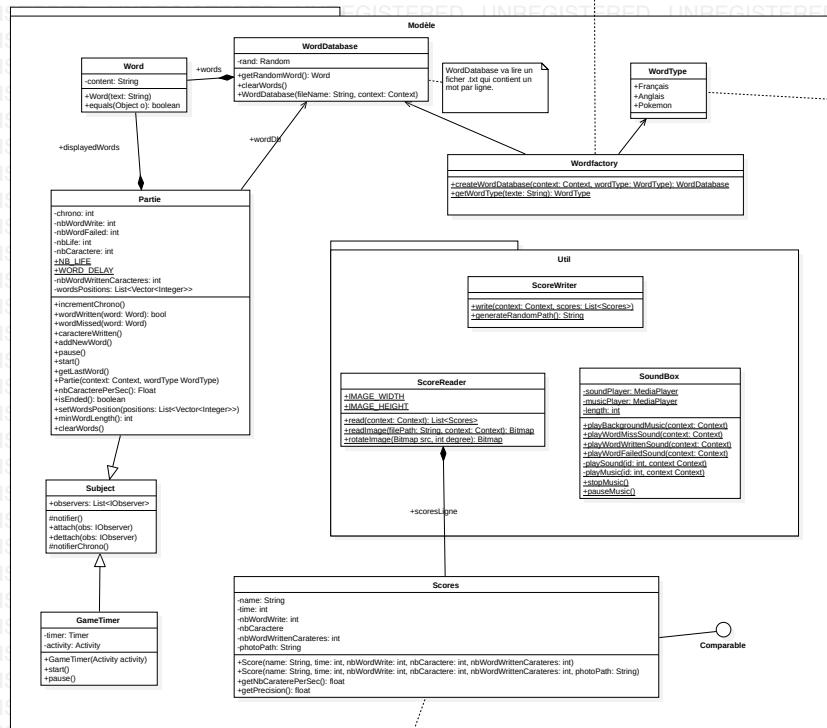
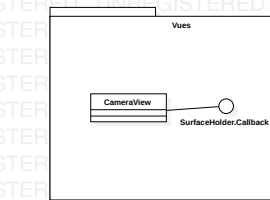
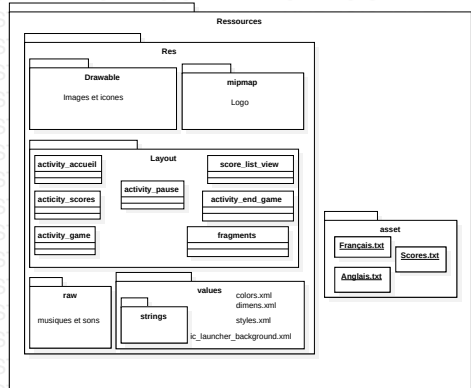
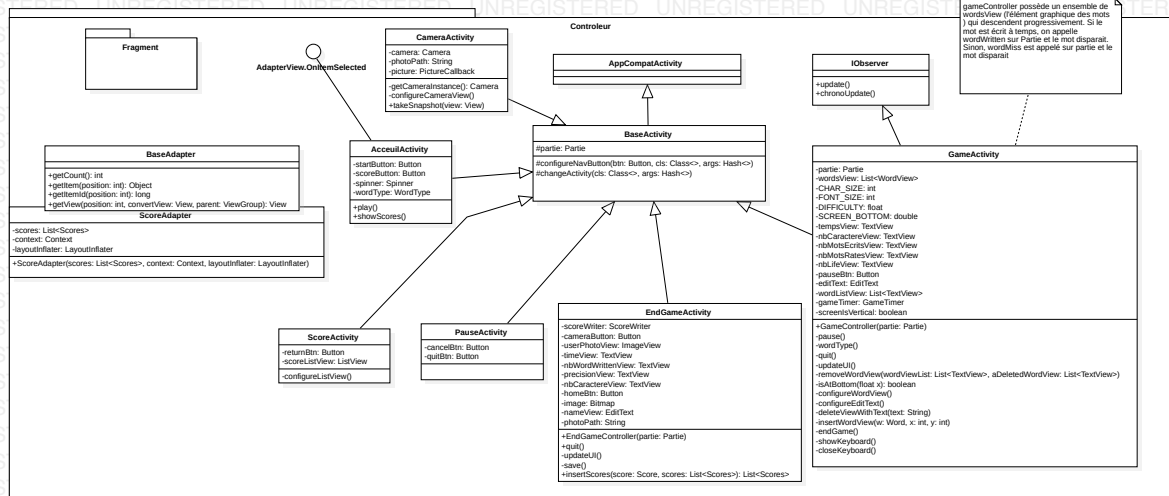


Model



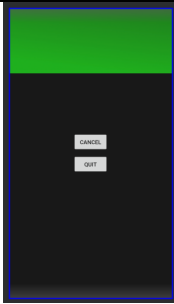
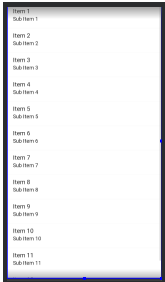
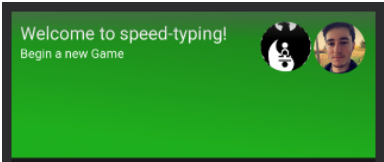
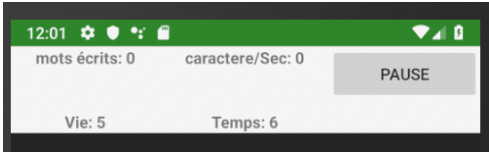
Fiche de compétences

Documentation

| | |
|---|----------------------|
| Je sais décrire le contexte de mon application | X |
| Je sais faire un diagramme de cas d'utilisation | X |
| Je sais concevoir un diagramme UML de qualité | Voir fichier uml.pdf |
| Je sais décrire un diagramme UML | X |

Code

| | |
|--|---|
| Je sais utiliser les Intents | <p>Ils sont utilisés dans la <u>BaseActivity->changeActivity</u>, classe mère des activités pour éviter les redondances.</p> <p>La plupart du temps, ce sont des boutons de navigation qui permettent de passer d'une vue à l'autre. On les configure via <u>configureNavigationButton</u> de BaseActivity qui fait appel à <u>changeActivity</u>.</p> <p>On sauvegarde dans les Intents une instance de la partie en cours (ou null), voir section persistance légère.</p> |
| J'utilise le SDK le plus bas possible | Le minSdkVersion de notre fichier gradle est de 15, pour un support de 99% des téléphones android sur le marché. |
| Je sais distinguer mes ressources en utilisant les qualificatifs | <p>Toutes nos ressources sont situées dans le bon qualificatif :</p> <ul style="list-style-type: none">-Nos images et icons dans le drawable.-Nos fichier de vue xml dans layout.-Notre logo dans mipmap.-Nos musiques dans raw.-Les autres fichiers tel que strings.xml dans values. |
| Je sais modifier le manifest de mon application | <p>Le manifest a été changé pour permettre l'utilisation de la caméra :</p> <pre><uses-permission android:name="android.permission.CAMERA" /> <uses-feature android:name="android.hardware.camera.front" android:required="false"/></pre> <p>Voir section Application.</p> |
| Je sais faire des vues xml en utilisant les composants adéquats | <p>Tous nos choix de layout ont été réfléchis pour être les mieux adaptés au contenant qu'ils affichent.</p> <p>Le ConstraintLayout est conservé pour activity_camera car pratique pour définir des vues superposés (le contenu de la caméra avec un bouton photo par dessus)</p> <div data-bbox="877 1590 1053 1886" data-label="Image"></div> <p>Les LinearLayout pour la vue d'accueil, de pause et de score.</p> |

| | |
|--|--|
| |  <p>La ListView pour l’affichage des scores.</p>  <p>Le RelativeLayout pour certains éléments.</p>  <p>Le GridLayout pour l’affichage des informations de la partie.</p>  |
| <p>Je sais coder proprement mes activités, pour qu’elle ne relaye que les évènements</p> | <p>Nos activités releyent bien les évènements au modèle. Le meilleur exemple est la relation entre GameActivity et Partie. Par exemple ligne 280 de GameActivity.</p> <pre>public void onTextChanged(CharSequence s, int start, int before, int count) { // Informer la partie qu'un caractère vient d'être écrit partie.caractereWritten(); }</pre> <p>Dès l’écriture d’un caractère par l’utilisateur, l’information est envoyée au modèle (partie).</p> <p>De manière générale, GameActivity met à jour Partie dès que besoin. Lorsqu’un mot est écrit, ou bien est descendue en bas, ou encore quand le jeu est se met en pause.</p> |
| <p>J’ai un véritable métier</p> | <p>Notre métier respecte les patron de conception étudiées à la période d’avant, notamment :</p> <p>Le patron observeur, utilisé 2 fois :</p> |

| | |
|--|--|
| | <p>-Pour la GameLoop, tous les abonnés au GameTimer sont notifiés chaque seconde.</p> <p>-Pour faire communiquer la Partie et la GameActivity, par exemple pour prévenir GameActivity qu'un nouveau mot à afficher a été ajouté dans Partie.</p> <p>Ces 2 relations observateurs/observés sont initiées lors de la création de GameActivity, et sont supprimées lorsque l'on change d'activité.</p> <p>GameActivity->onResume() ligne 120 :</p> <pre>partie.attach(this); gameTimer.attach(partie); gameTimer.attach(this);</pre> <p>GameActivity->Quit() ligne 150 :</p> <pre>partie.detach(this); gameTimer.detach(this); gameTimer.detach(partie);</pre> <p>Le patron fabrique : dans WordFactory. Qui permet de créer une instance de WordDatabase (liste de mots) qui va récupérer ses mots dans le bon fichier texte en fonction du WordType. Permet une bonne évolutivité de l'application en rajoutant facilement de nouvelles listes de mots.</p> |
| Je sais séparer la vue du modèle | <p>La vue et le modèle ne communiquent pas.</p> <p>Seul le context est passé à certaines classes qui doivent avoir accès au ressources du projet. Par exemple notre Wordfactory permet de créer dynamiquement une instance de WordDatabase en fonction d'un WordType. De plus, cela permet dans certains cas un meilleur respect du Single Responsibility Principle.</p> |
| Je maîtrise de cycle de vie de l'application | <p>Gestion des onPause, onCreate, onResume etc.</p> <p>Dans GameActivity->onResume(), on vérifie l'orientation de l'écran pour adapter la vue à afficher.</p> <p>Dès la fin de GameActivity, tous les nettoyages nécessaires sont faits dans quit():</p> <ul style="list-style-type: none"> -On ferme les observateurs -On arrête le timer -On ferme le clavier -On enregistre les positions des mots à l'écran |
| Je sais utiliser le findViewById | <p>Utilisé a de très nombreuses reprises.</p> <p>Par exemple dans GameActivity->onCreate()</p> |
| Je sais gérer les permissions dynamiques | <p>On utilise les permissions dynamiques pour l'accès à la caméra dans CameraView->onCreate()</p> <pre>ContextCompat.checkSelfPermission(getApplicationContext(), Manifest.permission.CAMERA)</pre> <p>On vérifie le résultat de la demande grâce à la méthode</p> <pre>public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults)</pre> |
| Je sais gérer la persistance légère | <p>L'instance de Partie est Serializable pour être passée d'une activity à une autre.</p> <p>Dans BaseActivity->changeActivity()</p> <pre>navIntent.putExtra("game", partie);</pre> <p>On la récupère dans BaseActivity->onCreate()</p> <pre>partie = (Partie) getIntent().getExtras().getSerializable("game");</pre> |

| | |
|--|--|
| | <p>Les positions des mots dans la vue de GameActivity sont sauvegardées lorsque l'on va dans PauseActivity (méthode quit()), pour être replacées lorsque l'on revient.</p> <pre>// Sauvegarde des positions des éléments List<Vector<Integer>> wordsPos = new ArrayList<>(); for (TextView wordView : wordViewList) { Vector<Integer> pos = new Vector<>(); pos.add((int) wordView.getX()); pos.add((int) wordView.getY()); wordsPos.add(pos); } partie.setWordsPositions(wordsPos);</pre> |
| Je sais gérer la persistance profonde | <p>Les scores sont enregistrés en interne dans un fichier texte accessible grâce au contexte. Un score possède un chemin d'accès vers la photo du joueur, également enregistrée dans un fichier binaire en interne. On l'utilise dans ScoreReader et ScoreWriter.</p> <pre>FileInputStream ips = context.openFileInput(fileName);</pre> <p>Ces scores seront accessibles jusqu'à la suppression de l'application.</p> |
| Je sais afficher une collection de données | <p>Voir ScoreActivity. On affiche les données dans une ListView.</p> <pre>private void configureListView() { List<Scores> scores = ScoreReader.read(this); ScoreAdapter adapter = new ScoreAdapter(this, scores, getLayoutInflater()); scoresListView.setAdapter(adapter); }</pre> |
| Je sais coder mon propre adaptateur | <p>ScoreAdapter hérite de BaseAdapter. On fournit les données de chacun des scores dans la méthode :</p> <pre>public View getView(int position, View convertView, ViewGroup parent)</pre> <p>qui récupère les éléments de vue de res->layout->score_list_view.xml</p> |
| Je maîtrise l'usage des fragments | <p>Nous avons testés les fragments à but expérimental dans les classes Fragments*.</p> |
| Je maîtrise l'utilisation de GIT | <p>Nous avons une branche chacun que l'on mergeait régulièrement grâce aux outils d'intellij.</p> |

Application

| | |
|----------------------------|---|
| Je sais utiliser la caméra | <p>Nous utilisons la caméra pour prendre en photo le joueur une fois la partie terminée. Cette photo est enregistrée avec le score et visage dans la page des scores.</p> <p>Voir CameraView et CameraActivity.</p> <p>Nous :</p> <ul style="list-style-type: none"> -demandons les permissions d'accès -vérifions l'existence de la caméra frontale -l'affichons correctement peut importe l'orientation (CameraView hérite de SurfaceView) -récupérons et enregistrons l'image grâce au PictureCallback dans CameraActivity. |
|----------------------------|---|