M1 Informatique

-

Rapport

Développement Mobile

Etudiant :
Hajanirina Randimbisoa

Enseignant :
Etienne Payet

25 novembre 2019

Table des figures

Table des matières

1 Résumé

Pour compléter l'unité d'enseignement développement mobiles, on a choisi de concevoir un jeu pour titre FlyBirds. Ce rapport montre les étapes et démarches que j'ai suivi pour réaliser enfin ce projet. Le travail est sur deux plateforme Android en utilisant le langage JAVA et iOS en utilisant SWIFT. J'ai utilisé AndroidStudio[1] vesion 3.5.2 et Xcode[2] version 11 comme outils de développement et overleaf pour la rédaction [3].

Selon le cahier de charge et les besoins[4], j'ai utilisé les cours, TD et TP que nous avons appris et fait en classe. Puis j'ai consulté aussi plusieurs tutoriels[5] et forum d'entraides comme stackoverflow[6], github[7] pour certains problèmes que j'ai rencontré au niveau de l'utilisation de capteur. Dans ce rapport,je vais évoquer la structure du projet et sa réalisation en illustrant avec des captures. Prémièrement sur le plateforme Android puis celui de iOS. Ce projet est libre et opensource, pour le télécharger voici un lien vers le dépôt github[8].

2 Introduction

Ce jeu FlyBirds se concentre sur un jeu de tir entre un chasseur et les oiseaux. Le but principal c'est de tirer sur les oiseaux avant qu'ils ne tuent le chasseur. Ce jeu est très amusant et motivant, grâce à la fonction HighScore, on veut toujours jouer pour battre son propre record.

En terme d'utilisation, il est très simple à manier. Il suffit d'utiliser ses doigts pour glisser vers la gauche ou droite. Le chasseur sous forme avion se déplace et orienté facilement aussi à l'aide du sansor que j'ai utilisé.

3 FlyBirds : Plateforme Android

3.1 Architecture générale

Dans ce répertoire, je vais mentionner quelles sont les classes très importantes et nécessaires. Plus precisement, ce sont les composantes qui présente l'architecture du projet.

a - MainActivity.java

Cette classe MainActivity.java est la classe le plus important et essentielles dans le développement de mon projet. J'ai mis dans cette classe, le menu principal de jeu, le score ainsi que le contrôle de son.

On a la méthode onCreate[1] qui est la création des activités qui va nous permetre de relier l'activité avec la vue avec la méthode setContentView. Puis, on a la méthode findViewById pour récupérer un élément de la vue selon son paramètre.

b- GameView.java

Cette classe conserve toutes nos fonctions qu'on aura besoin pour le fonctionnement du jeu. Elle hérite le SurfaceView et constitue plusieurs éléments pour déssiner à l'aide de canvas, bitmap et paint. On peut trouver des fonctions comme :

— update : quand le chasseur tire sur le prédateur il faut renouveler ses munitions, et on fait un random pour les oiseaux et de changement de vitesse en fonction du score.

— draw : pour dessiner les balles, oiseaux et le chasseur.
— saveIfHighScore : si on a un nouveau score on fait un update du precedent et on garde le meilleur score.
— OnTouchEvent : lorsque on tape ou touche l'ecran, le chasseur se déplace suivant l'orientation indiqué.

c - GameActivity

Dans le développement de ce jeu, cette classe est le noyau du jeu. Elle est liée au nombre maximale du boucles, fonctions, la mise en pause, la gestion de l'interruption du jeu et la reprise à l'aide de la méthode onResume. On initialise le GameView dans cette classe pour avoir la vue ou l'interface graphique. Bref, c'est la classe qui va gérer le lancement du jeu.

d - Bird.java

Cette classe contient les parametres pour créer les predateurs, les oiseaux. J'ai crée l'oiseau avec le variable bird en utilisant de Bitmap. C'est dans cette classe que j'ai definit la taille de l'oiseau, la vitesse de l'oiseau. La méthode getCollisionShape est utilisé pour récuper la forme en 2D de l'oiseau.

e - Flight.java

Cette classe assure la création du chasseur et la munition. Comme la création des oiseaux, j'ai utilisé le Bitmap et getCollisionShape pour récupérer la forme de l'avion qui est le joueur. Le shootCounter est utiliser pour faire une boucle de tire. Puis, la méthode getDead récupère le l'avion en 2D cassé, c'est-à-dire, lorsque les oiseaux touchent le chasseur, l'avion se crashe.

f - Bullet.java

Cette classe constitue uniquement la personnalisation et la création des balles. Bitmap pour la création d'image et la Rect getCollisionShape le responsable du retour en 2D. On peut trouver aussi la taille et la largeur de la balle avec les variables width et heigh et l'attribution du son[9] pour ce dernier.

g - Background.java

Pour avoir une aspect attirante et un décor, c'est dans cette classe que tout cela se passe. Pour definir l'arrière plan du jeu, il y a le bitmap pour decoder le ressource qui n'est autre que le fond[**?**].

h - Geolocalisation.java

Pour connaître et récupérer la position du joueur, c'est dans cette classe qu'on va définir tout les paramètres[1] dont on a besoin. Pour simplier les choses, j'ai utilisé une fonction qui va accéder à Google Maps afin d'afficher la position réelle du chasseur.

i - Internationalisation

On connaît tous que l'internationalisation d'une application devient presque une obligation si on veut développer un jeu et le mettre dans PlayStore. Pour cela, il suffit de créer un fichier de langue dans le dossier values des ressources (res). Puis, le nom du répertoire devra avoir pour valeur values-fr pour suivre la norme ISO 639-1 d'où l'abréviation "fr".

j - Capture.java

Pour rendre le jeu interessant, on a ajouté une possibilité de prendre une photo et qui devrait apparaître sur le highscore. Pour être prudent et pour éviter le crash de l'application, on va invoquer une intension de prendre une photo via la méthode StartActivityForResult()[1] afin d'être protéger par la méthode ResolveActivity().

Pour pouvoir enregistrer l'image, on a besoin d'une autorisation de l'utilisation de l'appareil photo via RequestTakePhoto[1] et de stockage getExternalStoragePublicDirectory. Pour ce faire, la lecture et son écriture requièrent ReadExternalStorage et WriteExternalStorage pour le stockage externe. Tout cela devait se passer dans l'android manifest.

3.2 Illustration des codes

a - MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    private boolean isMute;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN, WindowManager.LayoutParams.FLAG_FULLSCREEN);

        findViewById(R.id.play).setOnClickListener((view) → {
                startActivity(new Intent( packageContext: MainActivity.this, GameActivity.class));
        });

        TextView highScoreTxt = findViewById(R.id.highScoreTxt);
        final SharedPreferences prefs = getSharedPreferences( name: "game", MODE_PRIVATE);
        highScoreTxt.setText("HighScore: " + prefs.getInt( key: "highscore",  defValue: 0));

        isMute = prefs.getBoolean( key: "isMute",  defValue: false);
        final ImageView volumeCtrl = findViewById(R.id.volumeCtrl);

        if (isMute)
            volumeCtrl.setImageResource(R.drawable.ic_volume_off_black_24dp);
        else
            volumeCtrl.setImageResource(R.drawable.ic_volume_up_black_24dp);
        volumeCtrl.setOnClickListener((v) → {
                isMute = !isMute;
                if (isMute)
                    volumeCtrl.setImageResource(R.drawable.ic_volume_off_black_24dp);
                else
                    volumeCtrl.setImageResource(R.drawable.ic_volume_up_black_24dp);

                SharedPreferences.Editor editor = prefs.edit();
                editor.putBoolean("isMute", isMute);
                editor.apply();

        });
    }
}
```

FIGURE 1 – Classe MainActivity

b - GameView.java

```
private void saveIfHighScore() {
    if (prefs.getInt( key: "highscore",  defValue: 0) < score) {
        SharedPreferences.Editor editor = prefs.edit();
        editor.putInt("highscore", score);
        editor.apply();
    }
}
```

FIGURE 2 – Sauvegarde highScore dans GameView

c - Bird.java

```
public class Bird {

    public int speed = 20;
    public boolean wasShot = true;
    public int width, height;
    int x = 0, y;
    Bitmap bird1, bird2, bird3, bird4;
    private int birdCounter;

    Bird(Resources res) {

        bird1 = BitmapFactory.decodeResource(res, R.drawable.bird1);
        bird2 = BitmapFactory.decodeResource(res, R.drawable.bird2);
        bird3 = BitmapFactory.decodeResource(res, R.drawable.bird3);
        bird4 = BitmapFactory.decodeResource(res, R.drawable.bird4);

        width = bird1.getWidth();
        height = bird1.getHeight();

        width = bird1.getWidth();
        height = bird1.getHeight();

        width /= 6;
        height /= 6;

        width *= (int) screenRatioX;
        height *= (int) screenRatioY;

        bird1 = Bitmap.createScaledBitmap(bird1, width, height,  filter: false);
        bird2 = Bitmap.createScaledBitmap(bird2, width, height,  filter: false);
        bird3 = Bitmap.createScaledBitmap(bird3, width, height,  filter: false);
        bird4 = Bitmap.createScaledBitmap(bird4, width, height,  filter: false);

        y = -height;
    }
```

FIGURE 3 – Création de la classe Bird

d - Flight.java

```
Flight(GameView gameView, int screenY, Resources res) {

    this.gameView = gameView;

    flight1 = BitmapFactory.decodeResource(res, R.drawable.fly1);
    flight2 = BitmapFactory.decodeResource(res, R.drawable.fly2);

    width = flight1.getWidth();
    height = flight1.getHeight();

    width /= 4;
    height /= 4;

    width *= (int) screenRatioX;
    height *= (int) screenRatioY;

    flight1 = Bitmap.createScaledBitmap(flight1, width, height,  filter: false);
    flight2 = Bitmap.createScaledBitmap(flight2, width, height,  filter: false);

    shoot1 = BitmapFactory.decodeResource(res, R.drawable.shoot1);
    shoot2 = BitmapFactory.decodeResource(res, R.drawable.shoot2);
    shoot3 = BitmapFactory.decodeResource(res, R.drawable.shoot3);
    shoot4 = BitmapFactory.decodeResource(res, R.drawable.shoot4);
    shoot5 = BitmapFactory.decodeResource(res, R.drawable.shoot5);

    shoot1 = Bitmap.createScaledBitmap(shoot1, width, height,  filter: false);
    shoot2 = Bitmap.createScaledBitmap(shoot2, width, height,  filter: false);
    shoot3 = Bitmap.createScaledBitmap(shoot3, width, height,  filter: false);
    shoot4 = Bitmap.createScaledBitmap(shoot4, width, height,  filter: false);
    shoot5 = Bitmap.createScaledBitmap(shoot5, width, height,  filter: false);

    dead = BitmapFactory.decodeResource(res, R.drawable.dead);
    //Resizing the dead bitmap
    dead = Bitmap.createScaledBitmap(dead, width, height,  filter: false);

    y = screenY / 2;
    x = (int) (64 * (screenRatioX));

}
```

FIGURE 4 – Classe Flight

e - Bullet.java

```
public class Bullet {

    int x, y, width, height;
    Bitmap bullet;

    Bullet(Resources res) {
        bullet = BitmapFactory.decodeResource(res, R.drawable.bullet);

        width = bullet.getWidth();
        height = bullet.getHeight();

        width /= 4;
        height /= 4;
        width *= (int) screenRatioX;
        height *= (int) screenRatioY;

        bullet = Bitmap.createScaledBitmap(bullet, width, height, filter: false);

    }

    Rect getCollisionShape() { return new Rect(x, y, right: x + width, bottom: y + height); }

}
```

FIGURE 5 – Création des munitions

4 FlyBirds : Plateforme iOS

4.1 Architecture générale

On trouve dans cette partie la présentation de l'architecture générale de la version iOS du projet. Comme j'ai mentionné au le début, l'outil que j'ai utilisé c'est l'Xcode[10] 11. J'ai fait un choix entre Unity et SpriteKit[11] pour accompagner SWIFT et j'ai choisi SpriteKit. C'est simple à utiliser pour un débutant et qui se concentre sur l'écosystème d'Apple.

Cela nous donne une certaine confiance dans le fait que tous les nouveaux produits d'Apple seront bien pris en charge. Par exemple, on peut utiliser le même code SpriteKit[11] pour que notre jeu fonctionne sur iOS, macOS et tvOS sans accroc.

a - GameScene.swift

Cette classe est la résponsable principale de notre jeu. Elle contient l'ensemble du jeu. Elle comporte aussi des fonctions utiles pour manier le jeu et le geste courant ainsi la double tap.

b - GameViewController.swift

Le GameViewController.swift va nous permettre de lancer la vue en appelant la GameScene.swift. On peut voir la fluidité de jeu aussi en utilisant la méthode showsFPS.

c - Internationalisation

Localizable.strings est utilisé pour gérer l'internationalisation du jeu. Pour avoir la traduction, dans le cas dans ce jeu, francais en anglais, on utilise la KeyValues.

d - Géolocalisation

Comme on a fait sur Android, pour localiser la position du joueur, Geolocalisation.swift est utilisé pour faire cela et en utilisant la méthode CCLocationManager[12]. Chez Apple[2], ce fichier fait appel à une classe qui s'appelle MKMapView qui le responsable d'affichage de la carte sur l'ecran de user.

4.2 Illustration des codes

a - GameScene

```swift
func fireBullet() {
    let bullet = SKSpriteNode(imageNamed: "bullet")
    bullet.setScale(1)
    bullet.position = player.position
    bullet.zPosition = 1
    self.addChild(bullet)

    let moveBullet = SKAction.moveTo(y: self.size.height + bullet.size.height, duration: 1)
    let deleteBullet = SKAction.removeFromParent()
    let bulletSequence = SKAction.sequence([bulletSound, moveBullet, deleteBullet])
    bullet.run(bulletSequence)
}
```

FIGURE 6 – Création des munitions

b - GameViewController

```swift
class GameViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        if let view = self.view as! SKView? {
            // Load the SKScene from 'GameScene.sks'
            let scene = GameScene(size: CGSize(width: 1536, height: 2048))
                // Set the scale mode to scale to fit the window
                scene.scaleMode = .aspectFill

                // Present the scene
                view.presentScene(scene)


            view.ignoresSiblingOrder = true

            view.showsFPS = true
            view.showsNodeCount = true
        }
    }

    override var shouldAutorotate: Bool {
        return true
    }

    override var supportedInterfaceOrientations: UIInterfaceOrientationMask {
        if UIDevice.current.userInterfaceIdiom == .phone {
            return .allButUpsideDown
        } else {
            return .all
        }
    }

    override var prefersStatusBarHidden: Bool {
        return true
    }
}
```

FIGURE 7 – Classe GameViewController

5 Résultats

5.1 Android

a - Accueil
Voici donc à quoi ressemble l'écran principal du jeu :

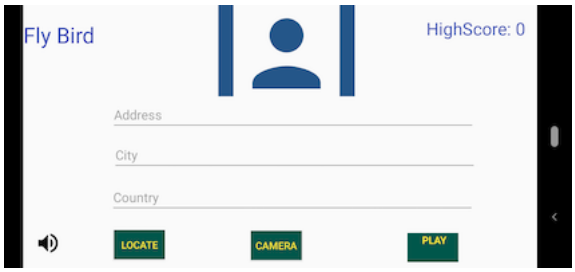

FIGURE 8 – Accueil v2.0

b - Mode Jeu
Capture en plein jeu :

FIGURE 9 – Capture en plein jeu android v1.0

c - Problème rencontré
Dans la partie Android, j'ai eu de difficulté sur la mise en place de la géolocalisation avec la gestion de key maps. En effet, ça ne m'empêcha pas de creuser et trouver des solutions pour résoudre ce problème en utilisant les TP, support de cours et les liens utiles[4] qu'on nous a donné en classe.

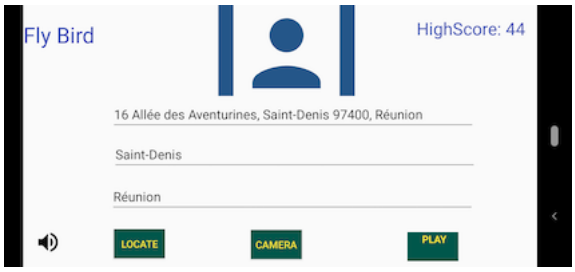

FIGURE 10 – Capture localisation v2.0

5.2 iOS

a - accueil
Pour celui de iOS, j'ai varié un peu le style de l'affichage :

FIGURE 11 – Acceuil du jeu v1.0

b - Mode Jeu
Voilà se qui s'affiche à l'écran quand on joue :

FIGURE 12 – Mode jeu v1.0

C - Problème rencontré
Dans cette partie iOS, je n'ai pas eu l'occasion de tester l'appareil photo sur un téléphone. Vue que, Apple ne permet pas l'utilisation de camera sur le simulateur, je ne sais pas si ca fonctionne vraiment ou pas. Pareil pour le géolocalisation. Malgré tout, le projet est presque terminé et quasi-répond au besoins du cahier de charge.

6 Conclusion

En guise de conclusion, ce projet fut une belle épreuve qui m'a permis plusieurs expérience dans le développement d'applications mobiles que ce soit en Android ou en iOS.

En réalisant ce projet, j'ai rencontré un peu de difficulté au niveau du langage SWIFT[2] et l'utilisation de LATEX[3]. Heureusement que j'ai eu l'occasion de consulter des forums d'aides et on m'a aidé à surmonter les épreuves. Ce n'est pas tout, j'ai aussi le privilège de consulter les TD et l'ensemble des TP qu'on avait fait en cours. Ce projet m'a permis donc d'améliorer mes capacités que ce soit au niveau de la rédaction en LATEX[3] ou en développement mobiles.

Références

[1] Documentation sur android studio. https://developer.android.com/docs/.

[2] Documentation en ios. https://developer.apple.com/documentation/.

[3] Rédaction de la présentation et rapport. https://overleaf.com.

[4] Consultation des cours, td et correction tp. https://moodle.univ-reunion.fr/course/view.php?id=785.

[5] Consultation d'aides. https://www.OpenClassroom.com/.

[6] Consultation de forum d'aide en android. https://stackoverflow.com/users/878126/android-developer.

[7] Documentation sur les capteurs ios. https://github.com/.

[8] Dépôt de la présentation, rapport et projet. https://github.com/HRandi/FlyBirds.

[9] Pour le son de balle et explosion. https://www.sound-fishing.net/bruitages_arme-et-explosion.html.

[10] Pour télécharger xcode 11. https://developer.apple.com/xcode/.

[11] Documentation spritekit. https://www.raywenderlich.com/71-spritekit-tutorial-for-beginners.

[12] Documentation sur la géolocalisation. https://developers.google.com/maps/documentation/ios-sdk/intro.