Rapport du Jeu

Brice VIRASSAMY

April 2019

The Forest

Résumé

Dans ce rapport, je fais la présentation d'un jeu que j'ai conçu et que j'ai nommé "The Forest", l'objectif de ce jeu est de frapper le plus d'ennemi possible sans être toucher. Le jeu se finit donc lorsqu'un ennemi nous touche sans avoir pu le frapper auparavant. Le jeu a été conçu pour le système d'exploitation "Android" et pour "iOS". Ce jeu est conçu avec l'aide d'un tutoriel iOS[1] et d'un tutoriel Android[2].

1 Introduction

Nous allons dans ce rapport examiner pour Android et iOS :

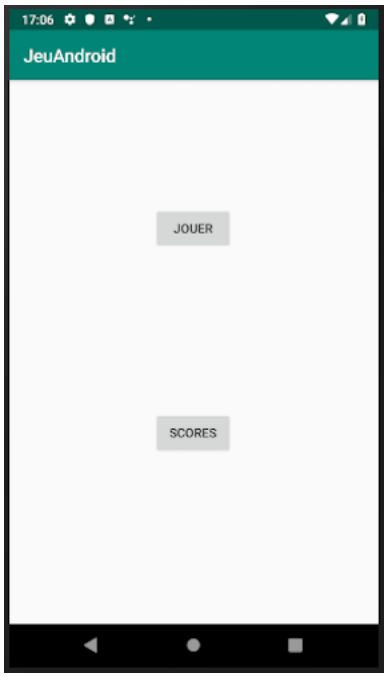
1. Le fonctionnement du jeu
2. l'architecture générale du jeu
3. Quelques points délicats
4. Les futures corrections

2 Le fonctionnement du jeu

2.1 A l'ouverture du jeu

2.1.1 Sous iOS

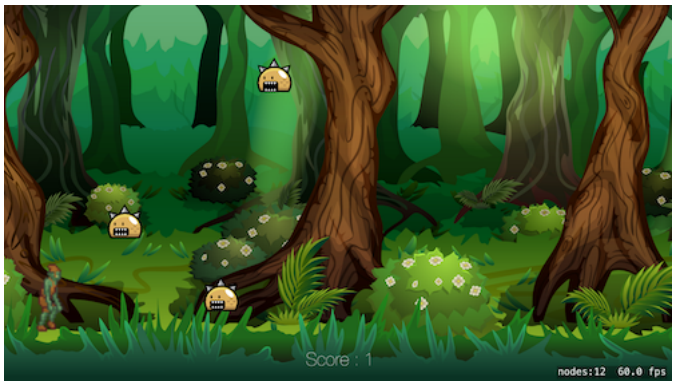
2.1.2 Sous Android



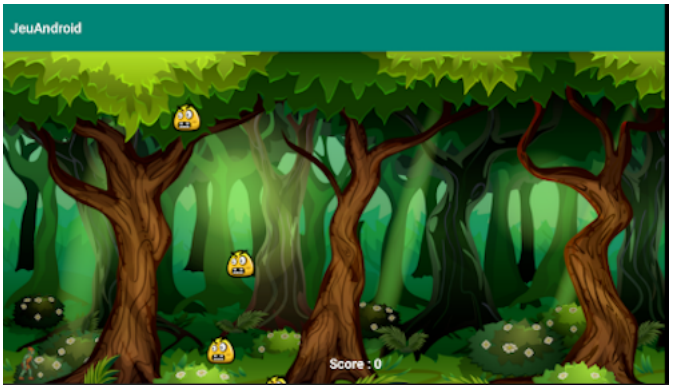
2.2 Jouer

Lorsque l'on clique sur le bouton jouer, nous avons :

2.2.1 Sous iOS



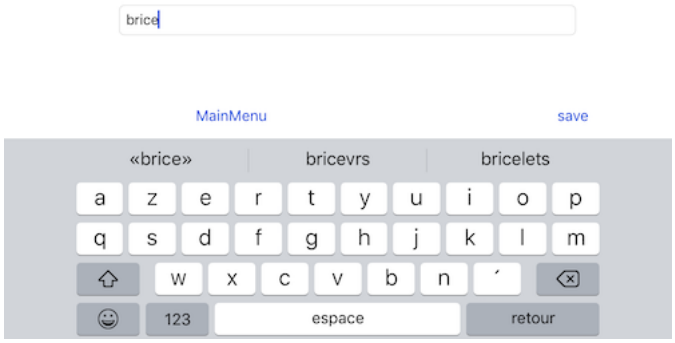
2.2.2 Sous Android



2.3 Sauvegarder son score

A la fin du jeu (lorsque l'on meurt), on peut choisir de sauvegarder ou non son score :

2.3.1 Sous iOS



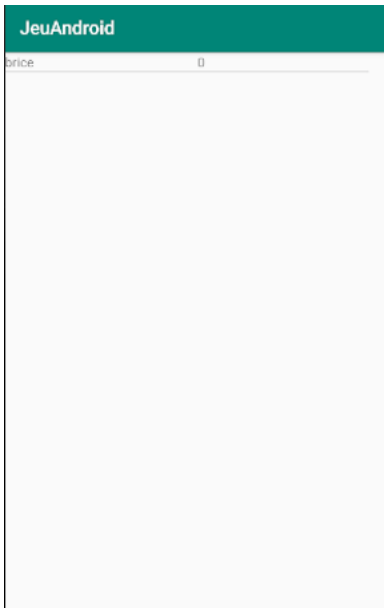
2.3.2 Sous Android

2.4 Tableau des scores

Lorsque l'on sauvegarde (similaire Scores sur le menu principal) le tableau des scores est affiché, si un clic est effectué sur le bouton "del" pour ios et en maintenant la cellule sur Android on procède à une suppression de la cellule :

2.4.1 Sous iOS

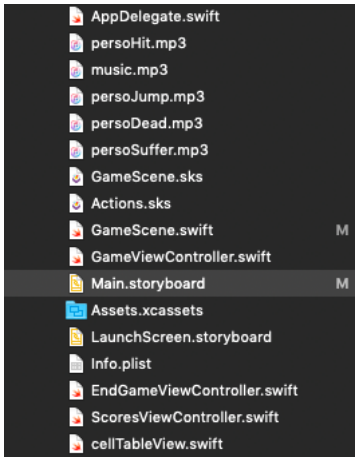
2.4.2 Sous Android



3 L'architecture générale du jeu

Sur iOS et Android l'organisation du code est différente.

3.1 Sous iOS

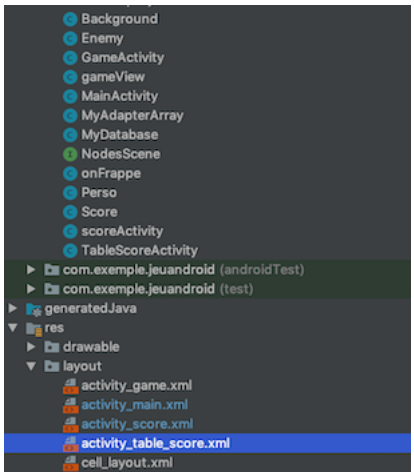


```
override func didMove(to view: SKView) {


    //Musique de fond
    if let musicURL = Bundle.main.url(forResource: "music", withExtension: "mp3"){
        backgroundMusic = SKAudioNode(url: musicURL)
        addChild(backgroundMusic)
    }

    self.anchorPoint=CGPoint(x: 0, y: 0)
    //capture des dimensions de la scene
    sceneX = self.size.width
    sceneY = self.size.height
    physicsWorld.contactDelegate = self
    //Config de la physique concenernant la scene
    physicsWorld.gravity = gravity
    self.physicsBody = SKPhysicsBody(edgeLoopFrom: CGRect(x: frame.minX, y: frame.minY, width: frame.width*2, height: frame.height))
    self.physicsBody?.collisionBitMask = sceneCategory

    scoreLabel = SKLabelNode(text: "Score : 0")
    scoreLabel.position = CGPoint(x:sceneX/2, y: 30)
    scoreLabel.fontSize = 34
    scoreLabel.fontColor = UIColor.white
    self.addChild(scoreLabel)

    addBackgrounds()
    addPerso()
    perso.run(SKAction.repeatForever(marcher()))
```

```
override func update(_ currentTime: TimeInterval) {

    addRandomEnemy()
    moveBackgrounds()
    moveEnemy()
}
```

La majorité de la logique du jeu est située dans la classe "gameScene.swift". A l'intérieur deux méthodes essentielles : didMove() et "update()". didMove() est utilisé pour initialisé le jeu et update() est une fonction qui tourne 60 fois par seconde et qui sert aux divers déplacements et à la génération d'ennemis principalement. "Main.storyboard" est utilisé pour creer les vues additionnelles

3.2 Sous Android




```java
public gameView(Context context) {
    super(context);

    //init score
    scorePaint = new Paint();
    //nbBg = 0;
    scorePaint.setColor(Color.WHITE);
    scorePaint.setTextSize(36);
    scorePaint.setTypeface(Typeface.DEFAULT_BOLD);

    //init perso
    perso = new Perso( v: this);
    frappe = true;


    //init background
    bg = BitmapFactory.decodeResource(getResources(), R.drawable.forest);
    //backgroundMoved
    bgs = new ArrayList<>();
    //init Enemy
    rand = new Random();
    enemies = new ArrayList<>();

    gameOver = false;
```

```java
@Override
protected void onDraw(final Canvas canvas) {
    super.onDraw(canvas);

    //canvas score
    //moveBackground(canvas);
    canvas.drawBitmap(bg, left: 0, top: 0, paint: null);
    //addBackground(canvas);
    //moveBackground(canvas);
    canvas.drawText( text: "Score : "+score,  x: canvas.getWidth()/2,  y: 850, scorePaint);
    //perso.setPosY(canvas.getHeight()-100);
    //canvas.drawBitmap(perso.getMap(), perso.getPosX(),perso.getPosY(),null);
    movePerso(perso,canvas);
    addEnemyInList(canvas);
    addEnemyInCanvas(canvas);
    DetectCollisions(canvas);
    frappe = false;
}
```

Pour Android, la logique est dissociée des vues et des données.

Les vues sont définies grâce à :

1. gameView (fichier de type View)
2. Tous les fichiers Xml (qui sont liés à des activités)

La logique est défini avec toutes les activités

Les données sont modélisés par les autres classes(Perso, Score...)

Extrait de Perso :

```java
public class Perso implements NodesScene{


    private Bitmap perso;
    private Bitmap persoFrappe[];
    private int life;
    private int posX;
    private int posY;
    private boolean dead;
    //rajouter peux etre thread annimation


    public Perso(View v){
        perso = BitmapFactory.decodeResource(v.getResources(),R.drawable.marche5);
        persoFrappe = new Bitmap[4];
        persoFrappe[0] = BitmapFactory.decodeResource(v.getResources(), R.drawable.frappe1);
        persoFrappe[1] = BitmapFactory.decodeResource(v.getResources(), R.drawable.frappe2);
        persoFrappe[2] = BitmapFactory.decodeResource(v.getResources(), R.drawable.frappe3);
        persoFrappe[3] = BitmapFactory.decodeResource(v.getResources(), R.drawable.frappe4);
        //this.hit = hit;
        life = 1;
        posX = 50;
        posY = 750;
        dead = false;
        //hit = new Thread();
        //hit.setDaemon(true);

    }
```

4 Quelques points délicats

4.1 Sous iOS

```swift
let storyboard = UIStoryboard(name: "Main", bundle: nil)
let vc = storyboard.instantiateViewController(withIdentifier: "endScene")
vc.view.frame=(self.view?.window?.rootViewController?.view.frame)!
vc.view.layoutIfNeeded()
UIView.transition(with: (self.view?.window)!, duration: 0.3, options: .transitionFlipFromRight, animations: {
    self.view?.window?.rootViewController=vc
}, completion: { completed in
    self.view?.window?.rootViewController?.dismiss(animated: false, completion: nil)
})
```

Pour accéder à une vue à l'intérieur du "Main.storyboard", j'ai du utilisé la manipulation dans la capture d'écran

4.2 Sous Android

```java
public class MyDatabase {

    private final static String DATABASE_NAME = "myDatabase.db";
    private final static int DATABASE_VERSION = 1;
    private final static String TABLE_SCORE = "score_table";
    private final static String COL_ID = "ID";
    private final static String COL_NAME = "NAME";
    private final static String COL_SCORE = "SCORE";

    private static final String CREATE_DB =
            "create table " + TABLE_SCORE + "("
                    + COL_ID + " integer primary key autoincrement, "
                    + COL_NAME + " text not null, "
                    + COL_SCORE + " integer);";

    private SQLiteDatabase database;
    private MyOpenHelper openHelper;

    public MyDatabase(Context context){
        openHelper = new MyOpenHelper(context, DATABASE_NAME, factory: null,DATABASE_VERSION);
    }

    public void open() { database = openHelper.getReadableDatabase(); }

    public void close() { database.close(); }

    public ArrayList<Score> getAllScore(){
        ArrayList<Score> list_score = new ArrayList<>();
        Cursor c = database.query(TABLE_SCORE, new String[]{COL_ID, COL_NAME, COL_SCORE},  selection: null, selectionArgs: null,
        c.moveToFirst();
        while(!(c.isAfterLast())){
            list_score.add(new Score(c.getLong( columnIndex: 0),c.getString( columnIndex: 1),c.getInt( columnIndex: 2)));
            c.moveToNext();
```

Pour pouvoir faire persister les données on doit pouvoir créer une base de données (créer avec la classe sur la capture ci-dessus, cette classe possède des méthodes pour ouvrir et fermer la base de données, insérer, retirer, afficher les scores

5 Conclusion

Pour conclure sur le jeu, il est correct mais quelques correctifs devront être appliquer.Pour iOS, réorganiser tout le code et adapter correctement les divers éléments de chaque vue.Pour Android, l'organisation est plus correct mais il faut aussi adapter les éléments des vues. Surtout faire attention que quelques sois l'écran du jeu, l'affichage sois correct. Pour les deux systèmes rajouter un bouton pour revenir au menu principal lorsque l'on est dans le tableau des scores. Pour parfaire l'application, rajouter une icône menu pour l'application.

Références

[1] Android tuto. https ://www.youtube.com/watch ?v=pAIC3JI1orglist=PLxefhmF0pcPk6$_jdMtHQ1QICTgqbbUvcIi$n 12.

[2] ios tuto. https ://www.linkedin.com/learning/developper-un-jeu-pour-ios-avec-swift-2-0-et-sprite-kit.