Rapport du projet

Íssam Jamai , Mohamed Amchemmere, L3 informatique

14 avril 2019

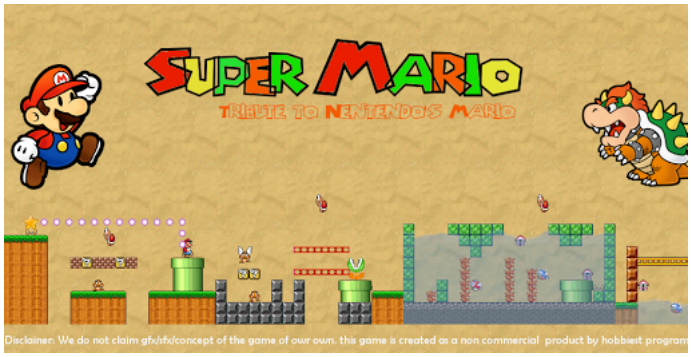
Résumé

Bonjour à tous , dans ce rapport on va essayer de vous rapprochez de notre projet pour bien comprendre ce qu'on a fait , Alors on va vous parlez de l'histoire du jeu qu'on a choisit ,sa Description c'est à dire (Comment jouer ?) et aussi le coté du code source.

1 Introduction

Beaucoup d'entre nous ont grandi en jouant à Super Mario , l'un des jeux les plus célèbres de l'industrie du jeu vidéo. Ce jeu est un clone android des célèbres frères Super Mario de Nintendo. Ce jeu a été lancé comme projet d'apprentissage de la programmation Android, mais il est bientôt devenu aussi proche que le jeu professionnel. Par conséquent, on a travaillé sur ce projet pour rappeler nos souvenirs .

Et voilà vous pouvez regardez le 'screen' de la première interface du jeu : ...



2 Description générale du jeu :

Le jeu Super Mario mettent en scène les aventures de Mario dans une royaume. Mario progresse dans des niveaux variés dans lesquels il saute pour battre des ennemis. Alors Le joueur est le protagoniste du jeu (Mario, ses couleurs dans le jeu sont le marron, le rouge et le jaune). Son frère est Luigi (ses couleurs sont le vert, le blanc, le marron et le jaune), mais contrairement à Mario, il n'a que le second rôle dans la partie. Ils doivent sauver tous deux Peach de Bowser. Le personnage bouge du côté gauche vers le côté droit de l'écran et doit se rendre au drapeau et du petit (ou grand) château qui marquent la fin des niveaux.

Il y a des pièces qui sont partout dans les niveaux, et Mario peut les prendre afin d'augmenter son score et, uniquement au bout de 100 pièces, peut avoir une vie. Les niveaux ont des Blocs " ?" qui sont presque tous (sauf le Bloc/Brique) marqués d'un point d'interrogation qui donnent soit des pièces ou soit des objets lorsque Mario les frappe. D'autres Blocs, parfois non-visibles, peuvent contenir plus de pièces ou un Power-Up. Si le protagoniste obtient un champignon (rouge), Mario grandit et peut prendre un coup par un ennemi ou un des obstacles dangereux sans en mourir, il devient tout simplement petit. Super Mario peut en plus casser les Briques, ce qui est impossible

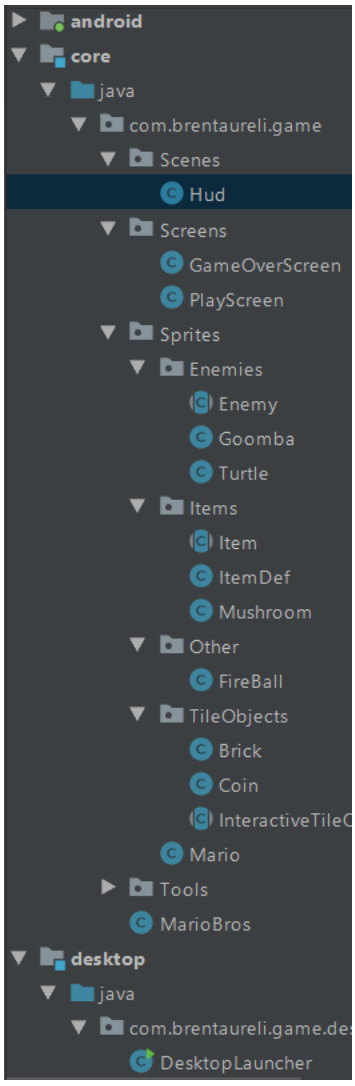
lorsque Mario est tout petit. À chaque début de partie, le joueur obtient 3 vies, qu'il peut faire augmenter en obtenant un Champignon 1-Up, similaire au rouge sauf qu'il est vert. Il peut perdre une vie, en tombant dans une crevasse ou quand le "Timing" est arrivé à 0. Lorsque le joueur n'a plus de vies, un Game Over apparaît alors. L'attaque connue de Mario est le saut mais chaque ennemi est différent : un Goomba sera vaincu par cette attaque, mais un Koopa Troopa se repliera dans sa carapace pendant un certain temps, permettant à Mario de l'utiliser comme petit projectile sur d'autres ennemis qui se situent sur le sol. La carapace peut aussi rebondir sur les murs, ou même tuer Mario si le "gamer" ne fait attention. Une autre façon de les vaincre est la Fleur de feu où Mario lance ses fameuses et connues boules de feu. Un dernier objet, plus rare est le plus fort, c'est la Super Étoile, qui rend Mario Invincible pendant quelques secondes.

3 Description du code

1
[1]
Dans ce paragraphe on va essayer de parler un peu du code afin de vous clarifier quelques notions dans le code , alors on va vous présenter des screens du code en Android et aussi version IOS :

3.1 Architecture

Voilà une image qui nous montre l'hièrarchie de notre projet et les différentes classes qu'on a utilisé :



3.2 Android

Ci dessous est un exemple de code du PlayScreen, c'est le code qui va permettre d'afficher l'écran où on peut controller Mario, voir le score et le temps ...

```
public PlayScreen(MarioBros game){
    atlas = new TextureAtlas( internalPackFile: "Mario_and_Enemies.pack");

    this.game = game;
    //create cam used to follow mario through cam world
    gamecam = new OrthographicCamera();

    //create a FitViewport to maintain virtual aspect ratio despite screen size
    gamePort = new FitViewport( worldWidth: MarioBros.V_WIDTH / MarioBros.PPM,  worldHeight: MarioBros.V_HEIGHT / MarioBros.PPM, gamecam);

    //create our game HUD for scores/timers/level info
    hud = new Hud(game.batch);

    //Load our map and setup our map renderer
    maploader = new TmxMapLoader();
    map = maploader.load( fileName: "level1.tmx");
    renderer = new OrthogonalTiledMapRenderer(map,  unitScale: 1  / MarioBros.PPM);

    //initially set our gamcam to be centered correctly at the start of of map
    gamecam.position.set( x: gamePort.getWorldWidth() / 2,  y: gamePort.getWorldHeight() / 2,  z: 0);

    //create our Box2D world, setting no gravity in X, -10 gravity in Y, and allow bodies to sleep
    world = new World(new Vector2( x: 0,  y: -10),  doSleep: true);
    //allows for debug lines of our box2d world.
    b2dr = new Box2DDebugRenderer();

    creator = new B2WorldCreator( screen: this);

    //create mario in our game world
    player = new Mario( screen: this);

    world.setContactListener(new WorldContactListener());

    music = MarioBros.manager.get( fileName: "audio/music/mario_music.ogg", Music.class);
    music.setLooping(true);
    music.setVolume(0.3f);
    //music.play();
```

Et dans cette image, on peut voir le code qui nous permet d'afficher les labels dans le PlayScreen, comme le score, le world et le temps .

```
public Hud(SpriteBatch sb){
    //define our tracking variables
    worldTimer = 300;
    timeCount = 0;
    score = 0;


    //setup the HUD viewport using a new camera seperate from our gamecam
    //define our stage using that viewport and our games spritebatch
    viewport = new FitViewport(MarioBros.V_WIDTH, MarioBros.V_HEIGHT, new OrthographicCamera());
    stage = new Stage(viewport, sb);

    //define a table used to organize our hud's labels
    Table table = new Table();
    //Top-Align table
    table.top();
    //make the table fill the entire stage
    table.setFillParent(true);

    //define our labels using the String, and a Label style consisting of a font and color
    countdownLabel = new Label(String.format( s: "%03d", worldTimer), new Label.LabelStyle(new BitmapFont(), Color.WHITE));
    scoreLabel =new Label(String.format( s: "%06d", score), new Label.LabelStyle(new BitmapFont(), Color.WHITE));
    timeLabel = new Label( text: "TIME", new Label.LabelStyle(new BitmapFont(), Color.WHITE));
    levelLabel = new Label( text: "1-1", new Label.LabelStyle(new BitmapFont(), Color.WHITE));
    worldLabel = new Label( text: "WORLD", new Label.LabelStyle(new BitmapFont(), Color.WHITE));
    marioLabel = new Label( text: "MARIO", new Label.LabelStyle(new BitmapFont(), Color.WHITE));

    //add our labels to our table, padding the top, and giving them all equal width with expandX
    table.add(marioLabel).expandX().padTop(10);
    table.add(worldLabel).expandX().padTop(10);
    table.add(timeLabel).expandX().padTop(10);
    //add a second row to our table
    table.row();
    table.add(scoreLabel).expandX();
    table.add(levelLabel).expandX();
    table.add(countdownLabel).expandX();
```

3.3 iOS

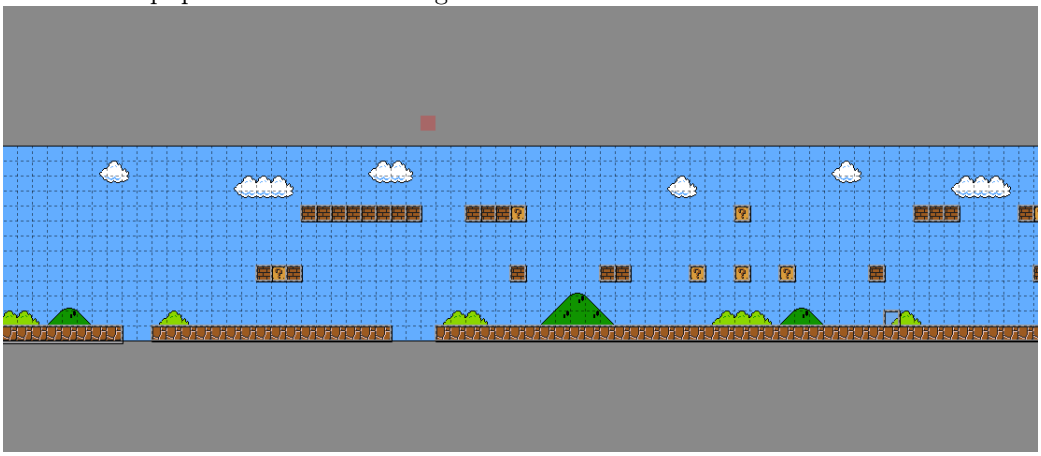
Et comme pour l'Android, voici le code responsable d'afficher le PlayScreen du jeu mais cette fois en version IOS :

```
class GameScene: SKScene, UIGestureRecognizerDelegate, SKSceneDelegate, SKPhysicsContactDelegate {

    let map = JSTileMap(named: "level.tmx");
    var hero = SKSpriteNode(imageNamed: "marioSmall_standing");
    var obstacles = SKNode();
    var swipeJumpGesture: UISwipeGestureRecognizer = UISwipeGestureRecognizer();

    var move: MoveDirection = .None;
    var isJumping: Bool = false;

    required init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder);
        self.commonShitInit();
    }

    override init(size: CGSize) {
        super.init(size: size);
        self.commonShitInit();
    }

    func commonShitInit () -> () {

        // Setting Scene
        self.backgroundColor = UIColor(red: 104.0 / 255.0, green: 136.0 / 255.0, blue: 255.0 / 255.0, alpha: 1.0);
        self.addChild(self.map);

        self.physicsBody = SKPhysicsBody(edgeLoopFromRect: CGRect(x: self.frame.origin.x, y: self.frame.origin.y - 36.0, width:
        self.frame.size.width, height: self.frame.size.height));
        self.physicsBody?.friction = 0.0;
        self.physicsWorld.contactDelegate = self;
        //        self.physicsBody.collisionBitMask = 0;
        //        self.physicsBody.contactTestBitMask = 0;
        //        self.physicsWorld.gravity = CGVectorMake(0,0);
        //        self.physicsWorld.contactDelegate = self;
```

4 Quelques points délicats/intéressants

Parmi les points qu'on veut développer et qui nécessite des clarifications on a décider de parler de :

- "LAYOUTS" ,alors c'est quoi un LAYOUT ? ?pour vous simplifier la réponse on peut dire que c'est comme une sorte de grande boîte dans laquelle on va placer les élements graphique de notre application (Les images , Les textes , Les boutons ....) . Mais la question c'est : Est ce qu'on peut travailler sans cette notion de LAYOUTS ? Normalement oui mais si on ajoute des élements sur notre application sans LAYOUTS on aura tout simplement aucune structure, alors si on change l'appereil ou si on veut faire une rotation d'écran , nos élements ne seront plus disposer correctement .
- Le logiciel "Tiled" : c'est est un éditeur de niveau 2D qui vous aide à développer le contenu de votre jeu. Sa principale fonctionnalité consiste à modifier des cartes de tiles de différentes formes, mais il prend également en charge le placement gratuit d'images ainsi que des moyens puissants d'annoter votre niveau avec des informations supplémentaires utilisées par le jeu. Tiled met l'accent sur la flexibilité générale tout en essayant de rester intuitif. L'image ci-dessous
- LibGD la map qu'on a crée avec ce logiciel :



LIBGDX : c'est un framework de développement de jeux multi-plateformes open source gratuit et de niveau relativement bas. L'objectif du projet est de vous aider à créer des jeux / applications et à le déployer sur des plates-formes de bureau et mobiles sans vous gêner et vous permettre de concevoir comme vous le souhaitez.

5 Conclusion

La création de ce jeu de bout en bout demande énormément de temps et une grande motivation. Nous en étions conscients en nous lançant dans cette tâche que nous avons pris beaucoup de plaisir à accomplir. Seulement l'ambition ne suffit pas pour aboutir et peut conduire facilement à un échec si elle est trop grande. La réalisation de ce premier jeu nous a permet de découvrir toutes les facettes qui composent un tel projet, ainsi que leur lot de problèmes. Nous avons consacré beaucoup de temps à apprendre les bases du langage de programmation "JAVA" car c'est le premier projet qu'on travaille avec ce langage , ainsi qu'on avait pas une grande concsience sur lui par ce que on l'a jamais étudier dans notre parcours , alors c'est la chose qui nous a appris beaucoup de temps , mais en même temps ce projet nous a aider à découvrir plein des nouvelles notions sur le "Développement Mobile ". Nous considérons que nous avons pas bien atteint l'objectif, mais vraiment on a fait le maximum possible pour rendre ce projet comme ça .

Références

[1] Youtube playlist. https://www.youtube.com/channel/UCO9JvZ75Usyzgd1puurLF6A.