[Tutoriel AndEngine] Partie 5 - Arrête ton char !

Bonsoir à tous, nous n'allons pas parler du film Ben-Hur malgré le titre en concordance avec cette expression, mais comment faire bouger un char d'assaut dans une scène :biggrin:

Tout d'abord nous allons apprendre à séparer notre scène de l'activité principale pour plus de propreté, nous continuerons en ajoutant un contrôle analogique fournit par l'andengine (oui oui ça existe !) et pour terminer nous allons faire bouger notre petit char (la classe non ?).

Comme d'habitude on commence par créer notre projet, puis à définir les propriétés de notre activité, comme ceci :

```
public class MonActivite extends BaseGameActivity {

 public static Camera camera;

 public final static int CAMERA_LARGEUR = 480;
 public final static int CAMERA_HAUTEUR = 320;

 @Override
 public Engine onLoadEngine() {
  // Initialisation de la caméra
  camera = new Camera(0, 0, CAMERA_LARGEUR, CAMERA_HAUTEUR);

  // Retourne le moteur de jeu
  return new Engine(new EngineOptions(true,
   ScreenOrientation.LANDSCAPE,
   new RatioResolutionPolicy(CAMERA_LARGEUR, CAMERA_HAUTEUR),
  camera));
 }

 @Override
 public void onLoadResources() {

 }

 @Override
 public Scene onLoadScene() {

 }

 @Override
 public void onLoadComplete() {

 }
```

```
}
```

Puis nous allons créer notre propre classe héritant de scène pour pouvoir travailler dedans et nous permettre de rendre notre code plus propre, vous remarquerez que setBackground, qui permettra de d'afficher une couleur verte en arrière plan (j'aime la pelouse sous les roues d'un char pas vous ?) :whistle:

```
public class SceneJeu extends Scene {

 /**
 * Constructeur
 *
 * @param pLayerCount
 */

 public SceneJeu(int pLayerCount) {
  super(pLayerCount);
  setBackground(new ColorBackground(0.52f, 0.75f, 0.03f));
 }

}
```

Voilà nous avons notre scène séparée de notre activité, nous allons donc l'ajouter à notre activité pour la charger

```
// Déclaration de la scéne du jeu
private SceneJeu maScene;
```

Puis dans le onLoadEngine(...)

```
// Initialisation de la scéne du jeu
maScene = new SceneJeu(1);
```

Et pour finir dans le onLoadScene()

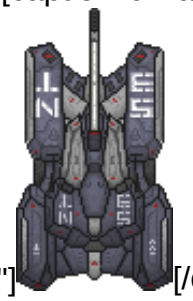
```
@Override
public Scene onLoadScene() {
 // Retourne la scéne
 return maScene;
}
```

Nous allons maintenant créer une classe Tank héritant de Sprite

```
public class Tank extends Sprite{

 public Tank(float pX, float pY, float pWidth, float pHeight,TextureRe
gion pTextureRegion) {
  super(pX, pY, pWidth, pHeight, pTextureRegion);
 }

}
```

Jusque la rien d'insurmontable, mais nous avons tout ce qu'il faut pour commencer à travailler, c'est moi qui vous le dit ! Prenons une belle photo d'un char d'assaut futuriste [caption id="attachment_358" align="aligncenter" width="73" caption="Char d'assaut"]  [/caption] Rajoutez le dans le dossier gfx, puis déclarons celui-ci dans notre scène

```
// Texture du tank
private Texture texture;
private TextureRegion textureRegionTank;
```

Petit soucis dans une scène nous n'avons pas accès au contexte ni au moteur pour charger les textures ! Hé bien feintons en rajoutant une méthode à notre scène nommée LoadResources avec 2 paramètres (Engine et Context)

```
/**
 * Chargement des resources
 *
 * @param engine
 * @param context
 */
public void LoadResources(final Engine engine, Context context) {
 // Chargement des textures du tank
 texture = new Texture(128, 256);
 textureRegionTank = TextureRegionFactory.createFromAsset(texture, con
text, "tank.png", 0, 0);

        // Chargement des textures dans le texture manager
        engine.getTextureManager().loadTextures(texture);

        // Lance l'initialisation de la scéne
        init();
```

```
}
```

Et pour finir dans notre activité puis dans la méthode onLoadResources appelons notre petite méthode

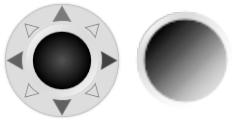
```
@Override
public void onLoadResources() {
 // Spécifie le répertoire de chargement des images
 TextureRegionFactory.setAssetBasePath("gfx/");
 // Chargement des textures de la scéne
 maScene.LoadResources(getEngine(), this);
}
```

Voilà notre tank texture est chargée en mémoire, plus qu'a l'afficher, pour cela nous allons rajouter la méthode init() à notre classe Scène

```
/**
 * Initialisation de la scéne
 */
private void init() {
 // Initialisation de notre tank
 tank = new Tank(0, 0, textureRegionTank.getWidth(), textureRegionTank
.getHeight(), textureRegionTank);
 // Redimensionne notre tank
 tank.setScale(0.5f);

 // Ajout de notre tank à la scéne
 getTopLayer().addEntity(tank);
}
```

Si nous lançons l'application vous verrez notre cher petit char d'assaut, super mais il bouge pas on va pas aller bien loin avec ça ! Qu'à cela ne tienne, on va arranger ce petit soucis. L'équipe de développement d'AndEngine à tout prévu en ajoutant un contrôle analogique dans le moteur de jeu, nous allons le rajouter pour pouvoir travailler avec. Commençons par télécharger les deux images du contrôle et plaçons les dans le répertoire GFX



Continuons en déclarant les textures dans notre scène

```
// Textures des contrôles pour se déplacer
private Texture mOnScreenControlTexture;
private TextureRegion mOnScreenControlBaseTextureRegion;
private TextureRegion mOnScreenControlKnobTextureRegion;
```

Puis dans notre LoadResouces

```
// Chargement des textures du contrôle analogique
mOnScreenControlTexture = new Texture(256, 128, TextureOptions.BILINEA
R_PREMULTIPLYALPHA);
mOnScreenControlBaseTextureRegion = TextureRegionFactory.createFromAss
et(this.mOnScreenControlTexture, context, "onscreen_control_base.png",
 0, 0);
mOnScreenControlKnobTextureRegion = TextureRegionFactory.createFromAss
et(this.mOnScreenControlTexture, context, "onscreen_control_knob.png",
 128, 0);

// Chargement des textures dans le texture manager
engine.getTextureManager().loadTextures(texture,mOnScreenControlTextur
e);
```

Nous avons chargé nos images en mémoire, y'a plus qu'a définir notre contrôle analogique

```
// Controle analogique
private AnalogOnScreenControl analogOnScreenControl;
```

Et pour finir dans notre méthode init

```
// Initialisation du controle analogique
analogOnScreenControl = new AnalogOnScreenControl(0,
 MonActivite.CAMERA_HAUTEUR - mOnScreenControlBaseTextureRegion.getHei
ght(),
 MonActivite.camera,
 mOnScreenControlBaseTextureRegion,
 mOnScreenControlKnobTextureRegion,
 0.1f,
 200,
 this);

// Spécifie que le contrôle analogie sera transparent
analogOnScreenControl.getControlBase().setBlendFunction(GL10.GL_SRC_AL
PHA, GL10.GL_ONE_MINUS_SRC_ALPHA);
// Valeur de la transparence
analogOnScreenControl.getControlBase().setAlpha(0.5f);
// Taille du centre
analogOnScreenControl.getControlBase().setScaleCenter(0, 128);
// Redimensionnement de la base du controle
analogOnScreenControl.getControlBase().setScale(0.5f);
// Redimensionnement du joystick
analogOnScreenControl.getControlKnob().setScale(0.5f);
// Rafraichit le controle
analogOnScreenControl.refreshControlKnobPosition();
```

```
// Ajout du controle à la scéne enfant
setChildScene(analogOnScreenControl);
```

Je n'explique pas plus, tout est dans les commentaire du code, passons à la gestion du déplacement du joystick, pour cela nous rajoutons l'interface à notre classe

```
public class SceneJeu extends Scene implements IAnalogOnScreenControlL
istener
```

Eclipse vous propose alors d'ajouter automatiquement les méthodes manquantes, faites le vous verrez ces deux méthodes supplémentaires :

```
@Override
public void onControlChange(BaseOnScreenControl pBaseOnScreenControl,f
loat pValueX, float pValueY) {

}

@Override
public void onControlClick(AnalogOnScreenControl pAnalogOnScreenContro
l) {

}
```

La première permet de savoir la position du joystick La deuxième si l'on clique dessus Nous allons nous intéresser uniquement à la première, après je vous laisserais faire ce que vous voulez de la deuxième :happy:

```
@Override
public void onControlChange(BaseOnScreenControl pBaseOnScreenControl,f
loat pValueX, float pValueY) {
 // Lorsque l'on bouge notre joystick le tank se déplace à une vitesse
 graduelle (la vélocité)
 tank.setVelocity(pValueX * 100, pValueY * 100);

 // Fait pivote notre tank selon la direction du joystick
 if(pValueX != 0 && pValueY != 0)
  tank.setRotation(MathUtils.radToDeg((float)Math.atan2(pValueX, -pVal
ueY)));
}
```

Quand nous allons bouger notre joystick le tank va se déplacer, puis il va pivoter selon l'orientation du joystick. Vous avez plus qu'a faire un petit jeu avec tout ça non ? [caption id="attachment_361" align="aligncenter" width="300" caption="Ça roule"]

 [/caption] Bon allez je vous dit à la prochaine pour un nouveau tutoriel (on va faire tirer notre tank !) Lien vers les sources : FormationDeplacementAndEngine