Desenvolvendo jogos com AndEngine

Diego Lopes Marques

http://sites.google.com/site/lmsdiego

@diegolms

AndEngine

- Uma pequena palestra sobre AndEngine

Montando o ambiente para o desenvolvimento

- SDK Android
  - http://developer.android.com/sdk/index.html
- Biblioteca AndEngine
  - https://sites.google.com/site/lmsdiego/cursos/desenvolvendo-jogos-para-android-com-andengine/andengine.jar?attredirects=0&d=1
- Instale o SDK Android(Atenção para o diretório de instalação, iremos precisar dele mais à frente)

Montando o ambiente para o desenvolvimento

- ADT Tools (Plugin para Eclipse)
  - Ao abrir o eclipse, selecione HELP > Install New Software
  - Clique em Add
  - Digite o nome (Ex: ADT Plugin) e em location informe a seguinte URL:

    https://dl-ssl.google.com/android/eclipse/,

    Se tiver problemas com o https, utilize o http: **http://dl-ssl.google.com/android/eclipse/**
  - Pressione OK

Montando o ambiente para o desenvolvimento

- Você verá na lista o novo site adicionado
  - Developer Tools
- Selecione e clique em FINISH.

  *O plugin ADT não é assinado, mas você pode aceitar a instalação de qualquer maneira clicando em Install All.
- Reinicie o eclipse

Montando o ambiente para o desenvolvimento

- Após o reinicio do eclipse, vamos atualizar as preferências.
  - Selecione Window > Preferences
  - No painel da esquerda, selecione Android
  - Clique em Browse para localizar o diretório do SDK Instalado
  - Clique em Apply e depois OK.

Primeiro projeto

- Agora que o nosso ambiente está montado, vamos criar o nosso primeiro projeto:
  - Selecione File > New > Other
  - Selecione Android > Android Project > Next
  - Preencha os campos de acordo com suas informações:
    - Project Name – Nome do seu projeto
    - Build Target – Versão do Android
    - Application Name – Nome da aplicação
    - Package Name – Nome do pacote
    - Create Activity – Nome da Activity
    - Min SDK Version – Versão do SDK
  - Clique em FINISH

Primeiro projeto

- O nosso primeiro projeto foi criado.
- Na classe criada deve ter um código parecido com esse:

```
public class MiniCursoAndEngine extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Primeiro projeto

- Feito isso, vamos começar a desenvolver com a biblioteca.
  - Clique com o botão direito sobre o seu projeto
  - Selecione New > Other
  - Selecione General > Folder
  - Em Folder Name digite: lib
  - A pasta lib será criada em seu projeto
  - Copie a biblioteca Andengine.jar que você baixou para o diretório lib
  - Adicione a biblioteca ao build path

Primeiro projeto

- Abra a classe do seu projeto
- Modifique-a para que ela herde de **BaseGameActivity**
- Adicione os imports necessários
- Adicione os métodos não implementados

Primeiro projeto

- Sua classe deve ficar parecida com essa

```
public class MiniCursoAndEngine extends BaseGameActivity {
	@Override
	public Engine onLoadEngine() {
		// TODO Auto-generated method stub
		return null;
	}
	@Override
	public void onLoadResources() {
	// TODO Auto-generated method stub
	}
	@Override
	public Scene onLoadScene() {
		// TODO Auto-generated method stub
		return null;
	}
	@Override
	public void onLoadComplete() {
		// TODO Auto-generated method stub
	}
}
```

Entendendo os métodos da classe

- onLoadEngine: Carrega o motor do jogo. São definidos nesse método se o jogo será fullscreen, qual será a resolução, a orientação da tela (portrait ou landscape), e etc...

- onLoadResources: este método é utilizado para carregar os resources do game. As imagens, fontes e áudios do game são carregados neste método.

- onLoadScene: método onde a cena será construída e carregada. É neste método que é estabelecido a posição e o estado inicial de todos os objetos do game.

- public void onLoadComplete(): Método chamado quando o load do game é completado.

Criando uma cena

- Vamos criar a nossa primeira cena do jogo.Para isso, precisamos de três elementos fundamentais para o jogo
  - Camera
  - Engine
  - SCene

Criando uma cena

- Camera
  - como tudo é baseado numa cena do jogo, precisamos de uma camera.

```
private Camera camera;
```

```
this.camera = new Camera(0, 0, CAMERA_WIDTH, CAMERA_HEIGHT);
```

- Parâmetros(posiçãoX, posiçãoY, largura, altura)

Criando uma cena

- Engine
  - Definimos a camera que será utilizada na cena

```
Engine(new EngineOptions(
true, ScreenOrientation.LANDSCAPE,
new RatioResolutionPolicy(CAMERA_WIDTH, CAMERA_HEIGHT), this.camera));
```

Parâmetros(Opções da Engine

(FullScreen,Posição, politica de resolução,camera))

Criando uma cena

- Scene
  - Criação da cena

```
private Scene cena;
cena = new Scene();
```

- Parâmetros – Pode ser com ou sem parâmetros.
- Se quiser passar parâmetros, será um inteiro, representando a quantidade de layer existente na cena.

Criando uma cena

Exercício - Conhecido os elementos principais do jogo,vamos criar a nossa primeira cena.

Criando uma cena

```
public class MiniCursoAndEngine extends BaseGameActivity {
    private final int CAMERA_WIDTH = 720;
    private final int CAMERA_HEIGHT = 480;
    private Camera camera;
    private Scene cena;

    public Engine onLoadEngine() {
        this.camera = new Camera(0, 0, CAMERA_WIDTH, CAMERA_HEIGHT);
        return new Engine(new EngineOptions(true, ScreenOrientation.LANDSCAPE,
                new RatioResolutionPolicy(CAMERA_WIDTH, CAMERA_HEIGHT), this.camera));
    }
    @Override
    public void onLoadResources() {
    }
    @Override
    public Scene onLoadScene() {
        cena = new Scene();
        cena.setBackground(new ColorBackground(225,225,225));
        return cena;
    }
    @Override
    public void onLoadComplete() {
    }
}
```

A função attachChild()

- A função attachChild() é uma das funções principais de um jogo, pois, é com ela que podemos colocar elementos na cena.

```
this.cena.attachChild(pEntity);
```

- Ela recebe por parâmetro imagens, textos, etc.

A função attachChild()

- Exercício – Desenhar o tabuleiro de um jogo da velha.

- Dica – Utilize linhas

```
Line line1 = new Line(Xinicial,Yinicial,Xfinal,Yfinal);
```

- Após criada as linhas, adicione na cena com a função attachChild().
- *você pode utilizar o método setColor(R,G,B) para colorir as linhas.

```
private void montarTabuleiro(){

    Line line1 = new Line((CAMERA_WIDTH/2)/2+40, 85, (CAMERA_WIDTH/2)/2+40, CAMERA_HEIGHT-16, 6);
    Line line2 = new Line(CAMERA_WIDTH-(CAMERA_WIDTH/2)/2-40, 85, CAMERA_WIDTH-(CAMERA_WIDTH/2)/2-40, CAMERA_HEIGHT-16, 6);
    Line line3 = new Line(0, 70, CAMERA_WIDTH, 70, 2);
    Line line4 = new Line(16, CAMERA_HEIGHT/2-30, CAMERA_WIDTH -16 ,CAMERA_HEIGHT/2-30,6);
    Line line5 = new Line(16, CAMERA_HEIGHT-130, CAMERA_WIDTH-16 ,CAMERA_HEIGHT-130,6);

    line1.setColor(100, 69, 0);
    line2.setColor(100, 69, 0);
    line3.setColor(148, 0, 150);
    line4.setColor(100, 69, 0);
    line5.setColor(100, 69, 0);

    this.cena.attachChild(line1);
    this.cena.attachChild(line2);
    this.cena.attachChild(line3);
    this.cena.attachChild(line4);
    this.cena.attachChild(line5);
}
```

Textures/TextureRegion

- Para adicionar imagens, texto, etc. na cena, precisamos de Textures.
- Elas são necessárias pois carregam as imagens

```
Texture textura = new Texture(
        256, 128, TextureOptions.BILINEAR_PREMULTIPLYALPHA);
```

- Parâmetros – (quantidade de pixelX, quantidade de pixelY, tipo da Textura)
- *Atenção especial para as quantidades de pixels, pois elas precisam ser potências de dois:
  - **(2,4,8,16,32,64,128,256...)**

Textures/TextureRegion

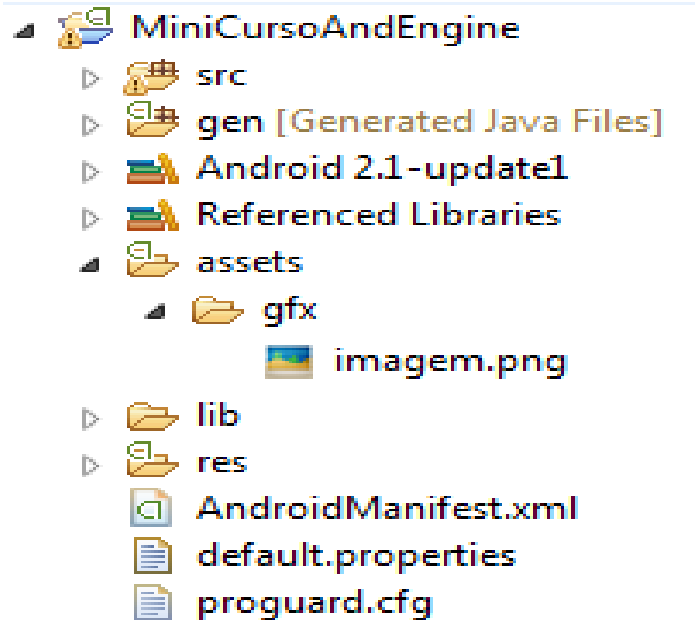
- TextureRegion É um objeto que representa uma imagem na textura.
- Ele é o "apontador" da imagem na textura que iremos carregar.

```
TextureRegion texturaJogador0 = TextureRegionFactory.createFromAsset(
        textura, this, "imagem.png", 0, 0);
```

- Parâmetros – (textura ,Contexto ”que no caso é a activity” , nome do arquivo,posiçãoX, posiçãoY)

Textures/TextureRegion

- Por boa prática de programação, o diretório utilizado para guardar as imagens é o gfx, que por sua vez deve estar dentro do diretório assets.



- Para facilitar, podemos utilizar a função abaixo para setar o caminho das imagens.

```
TextureRegionFactory.setAssetBasePath("gfx/");
```

Textures/TextureRegion

- Criada as texturas, devemos carregá-las para poder utilizá-las.
- O carregamento das texturas ocorre no método loadTextures()

```
public void onLoadResources() {
    TextureRegionFactory.setAssetBasePath("gfx/");
    Texture textura = new Texture(
            256, 128, TextureOptions.BILINEAR_PREMULTIPLYALPHA);
    TextureRegion texturaJogadorO = TextureRegionFactory.createFromAsset(
            textura, this, "imagem.png", 0, 0);
    TextureRegion texturaJogadorX = TextureRegionFactory.createFromAsset(
            textura, this, "imagem2.png", 128, 0);

    this.mEngine.getTextureManager().loadTextures(texturaJogadores,texturaFonte);

}
```

Textures/TextureRegion

- Carregada as texturas, podemos enfim criar imagens para adicionar no nosso jogo

```
Sprite bola = new Sprite(66,96, this.texturaJogadorO);
Sprite xis = new Sprite(320,96, this.texturaJogadorX);
this.cena.attachChild(bola);
this.cena.attachChild(xis);
```

Parâmetros – (posX,posY, TextureRegion)

*Lembrando que devem ser criadas no método onLoadScene()

Textures/TextureRegion

- Exercício – Criar texturas para representar uma bola e um xis de um jogo da velha e adicionar nas 9(nove) posições do tabuleiro de forma aleatória.

- Ex: Casa 1 – Bola
- Ex: Casa 2 - Xis

Fontes

- Para carregar uma fonte, também necessitamos de uma textura

```
public void onLoadResources() {

    this.texturaFonte = new Texture(256, 128, TextureOptions.NEAREST);
    this.fonte = new Font(texturaFonte, Typeface.create(
            Typeface.DEFAULT, 0), 32, true, Color.BLACK);
    this.mEngine.getTextureManager().loadTexture(texturaFonte);
    this.mEngine.getFontManager().loadFont(this.fonte);

    this.mEngine.getTextureManager().loadTextures(texturaFonte);

}
```

- Parâmetros – (Texture,tipo, antiAlias, cor)

Fontes

- Carregada as texturas das fontes, podemos utilizá-las.

```
textoJogador1 = new Text(
        10, 15, this.fonte, "Player 1: " +String.valueOf(pontosJogador1), HorizontalAlign.CENTER);
textoJogador2 = new Text(
        550, 15, this.fonte, "Player 2: " +String.valueOf(pontosJogador2), HorizontalAlign.CENTER);
```

- Parâmetros – (posX,posY,fonte, conteúdo(String),orientação)

Fontes

- Exercício – Escreva no topo da tela:
  - Jogador 1 = 0, Jogador 2 = 0

  Esse texto irá servir para os pontos

- *Lembrando que o conteúdo de um texto é uma String, ou seja, os pontos que são inteiros devem ser transformados em String. Para issol,utilize a função:
  - String.valueOf();

Eventos na tela

- Vamos agora tratar os eventos na tela

- Funciona da seguinte forma: quando o jogador clicar em um determinado canto da tela, acontece alguma ação.

Eventos na tela

- Defina sua classe para implementar a interface: IOSceneTouchListener

```
public class MiniCursoAndEngine extends BaseGameActivity implements IOnSceneTouchListener{
```

Eventos na tela

- O método onSceneTouchEvent é criado

```
public boolean onSceneTouchEvent(Scene pScene, TouchEvent pSceneTouchEvent) {
    return false;
}
```

Eventos na tela

- A função que identifica se houve ou não evento na tela é: isActionDown();

```
pSceneTouchEvent.isActionDown();
```

- Se houve evento na tela, então recupere a posição X e Y do evento com a função:

```
pSceneTouchEvent.getX(), pSceneTouchEvent.getY()
```

Eventos na tela

- Registre na cena o IOSceneTouchListener

```
this.cena.setOnSceneTouchListener(this);
```

Eventos na tela

- Exercício – Faça com que a bola/xis apareça na tela apenas com eventos.

Eventos na tela

- Resposta do exercício anterior

```
public boolean onSceneTouchEvent(Scene pScene, TouchEvent pSceneTouchEvent) {
    if(pSceneTouchEvent.isActionDown()) {
        Sprite bola = new Sprite(
                pSceneTouchEvent.getX(), pSceneTouchEvent.getY(), this.texturaJogador0);
        this.cena.attachChild(bola);
        return true;
    }

    return false;
}
```

Eventos em botões

- Com o IOSceneTouchListener identificamos que houve evento em qualquer parte da cena.
- Porém, dessa forma, qualquer parte da cena vai funcionar como um único evento.
- Por exemplo: Se no seu jogo tiver um botão, então você necessita de um evento específico para aquela área do botão.
- Então, para isso, criaremos um evento específico para cada botão do jogo, um evento de área.

Eventos em botões

- Para isso, criamos o evento do botão após a sua criação(Texturas), implementando o método:
  - onAreaTouch();
  - Parâmetros – (evento da cena, posX, posY)

```
botao = new Sprite(250,15,texturaBotaoNovoJogo){
    @Override
    public boolean onAreaTouched(
            final TouchEvent pSceneTouchEvent, final float pTouchAreaLocalX, final float pTouchAreaLocalY)
```

Eventos em botões

- Depois, utilizamos a função:

```
(pSceneTouchEvent.getAction()
```

- Que vai identificar qual o tipo do evento aconteceu.

Eventos em botões

- Os dois eventos mais comuns são:

```
TouchEvent.ACTION_DOWN:
```

  – Identifica um evento caso for um clique

```
TouchEvent.ACTION_UP:
```

  – Identifica um evento caso não for um clique

Eventos em botões

- Registre na cena o evento específico desse botão:

```
this.cena.registerTouchArea(botao);
```

Eventos em botões

- Exercício – Crie um botão Novo Jogo.
- Faça com que os elementos bola/xis, só possam aparecer na tela após o botão novo jogo ser acionado

Eventos em botões

- Reposta do exercício anterior

```
botao = new Sprite(250,15,texturaBotaoNovoJogo){
    @Override
    public boolean onAreaTouched(
            final TouchEvent pSceneTouchEvent, final float pTouchAreaLocalX, final float pTouchAreaLocalY) {
        switch(pSceneTouchEvent.getAction()) {
        case TouchEvent.ACTION_DOWN:
            this.setScale(1f);
            podeJogar = true;
            break;
        case TouchEvent.ACTION_UP:
            this.setScale(0.8f);
        default:
            this.setScale(0.8f);
            break;
        }
        return true;
    }
};
this.cena.attachChild(botao);
this.cena.registerTouchArea(botao);
```

Efeitos Sonoros

- Para carregar efeitos sonoros, precisamos pedir permissão para a engine.

```
public Engine onLoadEngine() {
    this.camera = new Camera(0, 0, CAMERA_WIDTH, CAMERA_HEIGHT);
    return new Engine(new EngineOptions(
            true, ScreenOrientation.LANDSCAPE,
            new RatioResolutionPolicy(CAMERA_WIDTH, CAMERA_HEIGHT), this.camera).setNeedsSound(true));
    }
```

Efeitos Sonoros

- Por boa prática de programação, os arquivos de audio devem estar na pasta mfx que por sua vez deve ser criado na pasta Assets.
- Para facilitar, podemos utilizar a função abaixo para setar o caminho dos efeitos.

```
SoundFactory.setAssetBasePath("mfx/");
```

Efeitos Sonoros

- Criaremos uma variável do tipo Sound

```
private Sound efeito;
```

Efeitos Sonoros

- Agora podemos carregar os efeitos sonoros da nossa aplicação

```
private void carregarEfeito(){
    SoundFactory.setAssetBasePath("mfx/");
    try {
        this.efeito = SoundFactory.createSoundFromAsset(this.mEngine.getSoundManager(), this, "efeito.mp3");
    } catch (final IOException e) {
        Debug.e(e);
    }
}
```

- *AndEngine trabalha com todos os tipos de mídia

Efeitos Sonoros

- Com os arquivos devidamente carregados, podemos colocá-los para tocar utilizando a função play()

```
this.efeito.play();
```

Efeitos Sonoros

- Exercício – Carregue efeitos sonoros para que cada jogador ao realizar uma jogada, toque o efeito definido.

- * Cada jogador deve ter um efeito diferente.

Musicas

- Para carregar musicas, precisamos pedir permissão para a engine.

```
public Engine onLoadEngine() {
    this.camera = new Camera(0, 0, CAMERA_WIDTH, CAMERA_HEIGHT);
    return new Engine(new EngineOptions(
            true, ScreenOrientation.LANDSCAPE,
            new RatioResolutionPolicy(CAMERA_WIDTH, CAMERA_HEIGHT), this.camera).setNeedsSound(true).setNeedsMusic(true));
}
```

Musicas

- Por boa prática de programação, os arquivos de musica também devem estar na pasta mfx que por sua vez deve ser criado na pasta Assets.
- Para facilitar, podemos utilizar a função abaixo para setar o caminho dos efeitos.

```
SoundFactory.setAssetBasePath("mfx/");
```

Efeitos Sonoros

- Criaremos uma variável do tipo Music

```
private Music musica;
```

Musicas

- Agora podemos carregar as músicas da nossa aplicação

```
private void carregarMusica(){
    MusicFactory.setAssetBasePath("mfx/");
    try {
        this.musica = MusicFactory.createMusicFromAsset(
                this.mEngine.getMusicManager(), this, "musica.mp3");
        this.musica.setLooping(true);
    } catch (final IOException e) {
        Debug.e(e);
    }
}
```

- *AndEngine trabalha com todos os tipos de mídia

Músicas

- Com os arquivos devidamente carregados, podemos colocá-los para tocar utilizando a função play()

```
this.musica.play();
```

Músicas

- Carregue uma música que irá tocar enquanto o jogo estiver rodando.

- Toda a parte gráfica foi desenvolvida, restando apenas a parte lógica do nosso jogo.
- Agora é com vocês !!!

BOA SORTE

Desenvolvendo jogos com AndEngine

OBRIGADO

Diego Lopes Marques

http://sites.google.com/site/lmsdiego

@diegolms