1. AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  android:versionCode="1" android:versionName="1.0"
  package="skripsi.akakom.peranghelikopter.game">

  <uses-sdk android:minSdkVersion="8" />

  <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
  <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
  <uses-permission android:name="android.permission.WAKE_LOCK" />
  <uses-permission android:name="android.permission.CAMERA" />
  <uses-permission android:name="android.permission.INTERNET" />
  <uses-permission android:name="android.permission.READ_PHONE_STATE" />
  <uses-permission android:name="android.permission.VIBRATE" />

  <supports-screens android:largeScreens="true"
        android:normalScreens="true" android:smallScreens="true"
        android:anyDensity="true"></supports-screens>

  <application android:icon="@drawable/icon"
android:label="@string/app_name" android:debuggable="true">

        <activity
android:name="skripsi.akakom.peranghelikopter.game.GameSplashActivity">

              <intent-filter>
                    <action android:name="android.intent.action.MAIN" />
                    <category
android:name="android.intent.category.LAUNCHER" />
              </intent-filter>

        </activity>

        <activity
android:name="skripsi.akakom.peranghelikopter.game.GameActivity"
android:label="@string/app_name">

        </activity>
        <activity
android:name="skripsi.akakom.peranghelikopter.angket.AngketActivity"
android:label="@string/app_name">

        </activity>

        <activity
android:name="skripsi.akakom.peranghelikopter.game.OptionsActivity"
android:label="@string/app_name"

  android:theme="@android:style/Theme.Black.NoTitleBar.Fullscreen"
              android:screenOrientation="portrait" />

        <activity
android:name="skripsi.akakom.peranghelikopter.game.HelpActivity"
android:label="@string/app_name"
```

```
  android:theme="@android:style/Theme.Black.NoTitleBar.Fullscreen"
              android:screenOrientation="portrait" />
        <activity
android:name="skripsi.akakom.peranghelikopter.game.MenuActivity"
android:label="@string/app_name"

  android:theme="@android:style/Theme.Black.NoTitleBar.Fullscreen"
              android:screenOrientation="portrait">
        </activity>




  </application>


</manifest>
```

2. GameSplashActivity

```
package skripsi.akakom.peranghelikopter.game;

import org.anddev.andengine.engine.Engine;
import org.anddev.andengine.engine.camera.Camera;
import org.anddev.andengine.engine.options.EngineOptions;
import
org.anddev.andengine.engine.options.EngineOptions.ScreenOrientation;
import
org.anddev.andengine.engine.options.resolutionpolicy.RatioResolutionPolicy
;
import org.anddev.andengine.entity.scene.Scene;
import org.anddev.andengine.entity.scene.background.ColorBackground;
import org.anddev.andengine.entity.shape.IShape;
import org.anddev.andengine.entity.shape.modifier.AlphaModifier;
import org.anddev.andengine.entity.shape.modifier.DelayModifier;
import org.anddev.andengine.entity.shape.modifier.IShapeModifier;
import org.anddev.andengine.entity.shape.modifier.SequenceModifier;
import
org.anddev.andengine.entity.shape.modifier.IShapeModifier.IShapeModifierLi
stener;
import org.anddev.andengine.entity.sprite.Sprite;
import org.anddev.andengine.opengl.texture.Texture;
import org.anddev.andengine.opengl.texture.TextureOptions;
import org.anddev.andengine.opengl.texture.region.TextureRegion;
import org.anddev.andengine.opengl.texture.region.TextureRegionFactory;
import org.anddev.andengine.opengl.texture.source.ITextureSource;
import org.anddev.andengine.ui.activity.BaseSplashActivity;

import android.app.Activity;
import android.content.Intent;

public class GameSplashActivity extends BaseSplashActivity {

        private static final int CAMERA_WIDTH = 480;
        private static final int CAMERA_HEIGHT = 800;
```

```
        private Camera camera;
        private Texture texture;
        private TextureRegion pictureRegion;

        @Override
        public Engine onLoadEngine() {
               this.camera = new Camera(0, 0, CAMERA_WIDTH,
CAMERA_HEIGHT);
               final EngineOptions engineOptions = new EngineOptions(true,
                              ScreenOrientation.PORTRAIT, new
RatioResolutionPolicy(
                                             CAMERA_WIDTH, CAMERA_HEIGHT),
this.camera);
               return new Engine(engineOptions);
        }

        @Override
        public void onLoadResources() {
                texture = new Texture(1024, 1024,
TextureOptions.BILINEAR);
                TextureRegionFactory.setAssetBasePath("img/");
                pictureRegion =
TextureRegionFactory.createFromAsset(this.texture, this, "splash.png", 0,
0);
                mEngine.getTextureManager().loadTexture(this.texture);
        }

        @Override
        public Scene onLoadScene() {

                final Scene scene = new Scene(1);
                scene.setBackground(new ColorBackground(0, 0, 0));

                final int x = (CAMERA_WIDTH -
this.pictureRegion.getWidth()) / 2;
                final int y = (CAMERA_HEIGHT -
this.pictureRegion.getHeight()) / 2;

                final Sprite picture = new Sprite(x, y,
this.pictureRegion);
                picture.setAlpha(0);
                scene.getTopLayer().addEntity(picture);

                picture.addShapeModifier(new SequenceModifier(
                          new IShapeModifierListener() {

                                        @Override
                                        public void
onModifierFinished(IShapeModifier pShapeModifier, IShape pShape) {


  texture.clearTextureSources();

TextureRegionFactory.createFromAsset(texture, GameSplashActivity.this,
"splashq.png", 0, 0);
```

```
                                                picture.addShapeModifier(new
SequenceModifier(
                                                new IShapeModifierListener()
{
                                                                        @Override
                                                                        public void
onModifierFinished(
   IShapeModifier pShapeModifier,
   IShape pShape) {

   Intent menu = new Intent(GameSplashActivity.this, MenuActivity.class);
   startActivity(menu);
   finish();
                                                                        }
                                                                    },
                                                new DelayModifier(1),
                                                new AlphaModifier(0.5f, 0,
1),
                                                new DelayModifier(2),
                                                new AlphaModifier(0.5f, 1,
0),
                                                                    new
DelayModifier(0.5f)
                                        ));
                                          }
                                    },
                                    new DelayModifier(1),
                              new AlphaModifier(0.5f, 0, 1),
                              new DelayModifier(2),
                              new AlphaModifier(0.5f, 1, 0)
                ));

                return scene;
        }

        @Override
        public void onLoadComplete() {}

        @Override
        protected ScreenOrientation getScreenOrientation() {
              return null;
        }

        @Override
        protected ITextureSource onGetSplashTextureSource() {
              return null;
        }

        @Override
        protected float getSplashDuration() {
              return 0;
```

```
        }

        @Override
        protected Class<? extends Activity> getFollowUpActivity() {
              return null;
        }
}
```

3. GameActivity.java

```
package skripsi.akakom.peranghelikopter.game;

import static android.view.ViewGroup.LayoutParams.FILL_PARENT;
import static android.view.ViewGroup.LayoutParams.WRAP_CONTENT;

import java.io.IOException;
import java.util.ArrayList;


import org.anddev.andengine.audio.music.Music;
import org.anddev.andengine.audio.music.MusicFactory;
import org.anddev.andengine.audio.sound.Sound;
import org.anddev.andengine.audio.sound.SoundFactory;
import org.anddev.andengine.collision.BaseCollisionChecker;
import org.anddev.andengine.engine.Engine;
import org.anddev.andengine.engine.camera.Camera;
import
org.anddev.andengine.engine.camera.hud.controls.AnalogOnScreenControl;
import
org.anddev.andengine.engine.camera.hud.controls.AnalogOnScreenControl.IAna
logOnScreenControlListener;
import
org.anddev.andengine.engine.camera.hud.controls.BaseOnScreenControl;
import org.anddev.andengine.engine.handler.IUpdateHandler;
import org.anddev.andengine.engine.handler.timer.ITimerCallback;
import org.anddev.andengine.engine.handler.timer.TimerHandler;
import org.anddev.andengine.engine.options.EngineOptions;
import
org.anddev.andengine.engine.options.EngineOptions.ScreenOrientation;
import
org.anddev.andengine.engine.options.resolutionpolicy.RatioResolutionPolicy
;
import org.anddev.andengine.entity.primitive.Rectangle;
import org.anddev.andengine.entity.scene.Scene;
import org.anddev.andengine.entity.shape.IShape;
import org.anddev.andengine.entity.shape.modifier.ColorModifier;
import org.anddev.andengine.entity.shape.modifier.DelayModifier;
import org.anddev.andengine.entity.shape.modifier.IShapeModifier;
import
org.anddev.andengine.entity.shape.modifier.IShapeModifier.IShapeModifierLi
stener;
import org.anddev.andengine.entity.shape.modifier.AlphaModifier;
import org.anddev.andengine.entity.shape.modifier.PathModifier;
import org.anddev.andengine.entity.shape.modifier.RotationByModifier;
import org.anddev.andengine.entity.shape.modifier.ScaleModifier;
import org.anddev.andengine.entity.shape.modifier.SequenceModifier;
```

```
import org.anddev.andengine.entity.shape.modifier.ease.EaseSineInOut;
import org.anddev.andengine.entity.sprite.Sprite;
import org.anddev.andengine.entity.text.Text;
import org.anddev.andengine.entity.util.FPSLogger;
import org.anddev.andengine.opengl.font.Font;
import org.anddev.andengine.opengl.font.FontFactory;
import org.anddev.andengine.opengl.texture.BuildableTexture;
import org.anddev.andengine.opengl.texture.Texture;
import org.anddev.andengine.opengl.texture.TextureOptions;
import
org.anddev.andengine.opengl.texture.builder.BlackPawnTextureBuilder;
import
org.anddev.andengine.opengl.texture.builder.ITextureBuilder.TextureSourceP
ackingException;
import org.anddev.andengine.opengl.texture.region.TextureRegion;
import org.anddev.andengine.opengl.texture.region.TextureRegionFactory;
import org.anddev.andengine.opengl.texture.region.TiledTextureRegion;
import org.anddev.andengine.opengl.view.RenderSurfaceView;
import org.anddev.andengine.sensor.accelerometer.AccelerometerData;
import org.anddev.andengine.sensor.accelerometer.IAccelerometerListener;
import org.anddev.andengine.ui.activity.BaseGameActivity;
import org.anddev.andengine.util.Debug;
import org.anddev.andengine.util.HorizontalAlign;
import org.anddev.andengine.util.Path;

import skripsi.akakom.peranghelikopter.game.R;
import skripsi.akakom.peranghelikopter.game.background.ScrollBackground;
import skripsi.akakom.peranghelikopter.game.effects.Explosion;
import skripsi.akakom.peranghelikopter.game.enemies.Boss;
import skripsi.akakom.peranghelikopter.game.enemies.EnemyBullet;
import skripsi.akakom.peranghelikopter.game.enemies.EnemyShip;
import skripsi.akakom.peranghelikopter.game.hud.HudScore;
import skripsi.akakom.peranghelikopter.game.level.Level;
import skripsi.akakom.peranghelikopter.game.options.Options;
import skripsi.akakom.peranghelikopter.game.player.Bullet;
import skripsi.akakom.peranghelikopter.game.player.PlayerShip;

import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.Typeface;
import android.net.Uri;
import android.os.Vibrator;
import android.view.Gravity;
import android.view.KeyEvent;
import android.widget.EditText;
import android.widget.FrameLayout;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.RelativeLayout.LayoutParams;
import android.widget.TextView;

public class GameActivity extends BaseGameActivity implements
IAccelerometerListener {
```

```
// CONSTANTS
public static final int CAMERA_WIDTH = 480;
public static final int CAMERA_HEIGHT = 800;
public static final int ANIMATION_FRAMELENGTH = 1;
public static final int ANIMATION_FRAMELENGTH1 = 60;
// CAMERA
private Camera camera;

// TEXTURES AND FONTS
private BuildableTexture mBuildableTexture;
private static TiledTextureRegion mShipTextureRegion;
private static TiledTextureRegion mEnemyTextureRegion;
public static TextureRegion mBulletTextureRegion;
public static TextureRegion mEnemyBulletTextureRegion;
private static TiledTextureRegion mExplosionTextureRegion;
private TiledTextureRegion mBossTextureRegion;
private TextureRegion mLaserTextureRegion;
private Texture font_texture;
private Texture gameOver_font_texture;
private Font font;
private Font gameOver_font;
private Texture mOnScreenControlTexture;
  private TextureRegion mOnScreenControlBaseTextureRegion;
  private TextureRegion mOnScreenControlKnobTextureRegion;
private Texture mAutoScrollBackgroundTexture;
private TextureRegion mScrollLayer;

// SOUNDS
public static Sound shotSound;
private static Sound explosionSound;
private static Sound gameOverSound;

private Music music;

// OBJECTS
private PlayerShip ship;
private Boss boss;
public static ArrayList<EnemyShip> enemies;
public static ArrayList<EnemyShip> enemiesToReuse;
public static ArrayList<Explosion> explosions;
public static ArrayList<Explosion> explosionsToReuse;
public static ArrayList<Bullet> bullets;
public static ArrayList<Bullet> bulletsToReuse;
public static ArrayList<EnemyBullet> enemyBullets;
public static ArrayList<EnemyBullet> enemyBulletsToReuse;

// PATHS
public static final Path[] paths = new Path[4];

// VARIABLES
public static final HudScore score = new HudScore();
public static boolean isGameOver;
public static boolean isGameReady;
public static boolean isBossFight;

// SHARED PREFERENCES
public static Options options;
```

```
   // SCORE - TEXTVIEW
   private static TextView scoreView;

   // Hit point
   private Rectangle healthbar;
   private Rectangle okvir;

   // LEVEL
   private Level level;

   // backgroud
   private ScrollBackground ScrollBackground;

   // FADE
   private Rectangle fade;

   @Override
   protected void onSetContentView() {
         final RelativeLayout relativeLayout = new RelativeLayout(this);
         final FrameLayout.LayoutParams relativeLayoutLayoutParams = new
FrameLayout.LayoutParams(FILL_PARENT, FILL_PARENT);

         scoreView = new TextView(getApplicationContext());
         Typeface font =
Typeface.createFromAsset(getAssets(),"font/pf_tempesta_five.ttf");
         scoreView.setTypeface(font);

         scoreView.setPadding(30, 30, 0, 0);
         scoreView.setTextColor(Color.BLACK);
         scoreView.setGravity(Gravity.CENTER);

         this.mRenderSurfaceView = new RenderSurfaceView(this,
this.mEngine);
         this.mRenderSurfaceView.applyRenderer();

         final LayoutParams surfaceViewLayoutParams = new
RelativeLayout.LayoutParams(super.createSurfaceViewLayoutParams());
         surfaceViewLayoutParams.addRule(RelativeLayout.CENTER_IN_PARENT);
         relativeLayout.addView(this.mRenderSurfaceView,
surfaceViewLayoutParams);
         relativeLayout.addView(scoreView,
this.createAdViewLayoutParams());

         this.setContentView(relativeLayout, relativeLayoutLayoutParams);
   }

   private LayoutParams createAdViewLayoutParams() {
         final LayoutParams adViewLayoutParams = new
LayoutParams(WRAP_CONTENT, WRAP_CONTENT);
         adViewLayoutParams.addRule(RelativeLayout.ALIGN_PARENT_TOP);
         adViewLayoutParams.addRule(RelativeLayout.ALIGN_LEFT);
         return adViewLayoutParams;
   }

   @Override
   public Engine onLoadEngine() {
```

```
        options = new Options(GameActivity.this);
        this.camera = new Camera(0, 0, CAMERA_WIDTH, CAMERA_HEIGHT);
        final EngineOptions engineOptions = new EngineOptions(true,
                        ScreenOrientation.PORTRAIT, new
RatioResolutionPolicy(
                                    CAMERA_WIDTH, CAMERA_HEIGHT),
this.camera);

        if (options.getSoundEffects() == true)
              engineOptions.setNeedsSound(true);

        if (options.getMusic() == true)
              engineOptions.setNeedsMusic(true);

        return new Engine(engineOptions);
  }

  @Override
  public void onLoadResources() {

        this.mBuildableTexture = new BuildableTexture(512, 512,
TextureOptions.DEFAULT);

        TextureRegionFactory.setAssetBasePath("img/");

        mLaserTextureRegion =
TextureRegionFactory.createFromAsset(this.mBuildableTexture, this,
"laser.png");

        GameActivity.mShipTextureRegion =
TextureRegionFactory.createTiledFromAsset(this.mBuildableTexture, this,
"player.png", 3, 1);
        GameActivity.mEnemyTextureRegion =
TextureRegionFactory.createTiledFromAsset(this.mBuildableTexture, this,
"enemy2.png", 3, 1);
        GameActivity.mBulletTextureRegion =
TextureRegionFactory.createFromAsset(this.mBuildableTexture, this,
"meci.png");
        GameActivity.mEnemyBulletTextureRegion =
TextureRegionFactory.createFromAsset(this.mBuildableTexture, this,
"meci_neprijatelji.png");
        GameActivity.mExplosionTextureRegion =
TextureRegionFactory.createTiledFromAsset(this.mBuildableTexture, this,
"explosion2.png", 4, 2);

        if (GameActivity.options.getControlls().equals("3")) {
              this.mOnScreenControlTexture = new Texture(256, 128,
TextureOptions.BILINEAR);
          this.mOnScreenControlBaseTextureRegion =
TextureRegionFactory.createFromAsset(this.mOnScreenControlTexture, this,
"onscreen_control_base.png", 0, 0);
          this.mOnScreenControlKnobTextureRegion =
TextureRegionFactory.createFromAsset(this.mOnScreenControlTexture, this,
"onscreen_control_knob.png", 128, 0);
```

```
this.mEngine.getTextureManager().loadTextures(this.mOnScreenControlTexture
);
        }

        mBossTextureRegion =
TextureRegionFactory.createTiledFromAsset(this.mBuildableTexture, this,
"boss.png", 2, 2);

        try {
              this.mBuildableTexture.build(new
BlackPawnTextureBuilder());
        } catch (final TextureSourcePackingException e) {
              Debug.e(e);
        }

        this.mAutoScrollBackgroundTexture = new Texture(512, 1024,
TextureOptions.DEFAULT);
        this.mScrollLayer =
TextureRegionFactory.createFromAsset(this.mAutoScrollBackgroundTexture,
this,"background0.png", 0, 0);

        FontFactory.setAssetBasePath("font/");
        font_texture = new Texture(256, 256, TextureOptions.BILINEAR);
        font = FontFactory.createFromAsset(font_texture,
this,"pf_tempesta_five.ttf", 60, true, Color.RED);

        gameOver_font_texture = new Texture(256, 256,
TextureOptions.BILINEAR);
        gameOver_font =
FontFactory.createFromAsset(gameOver_font_texture,this,
"pf_tempesta_five.ttf", 60, true, Color.BLACK);

        this.mEngine.getTextureManager().loadTextures(mBuildableTexture,
font_texture,
                    gameOver_font_texture, mAutoScrollBackgroundTexture);
        this.mEngine.getFontManager().loadFont(font);
        this.mEngine.getFontManager().loadFont(gameOver_font);

        if (options.getSoundEffects() == true) {
              SoundFactory.setAssetBasePath("snd/");
              try {
                    GameActivity.shotSound =
SoundFactory.createSoundFromAsset(
                                this.mEngine.getSoundManager(), this,
"shot.ogg");
                    GameActivity.explosionSound = SoundFactory

  .createSoundFromAsset(this.mEngine.getSoundManager(),
                                          this, "explosion.ogg");
                    GameActivity.gameOverSound =
SoundFactory.createSoundFromAsset(
                                this.mEngine.getSoundManager(), this,
"gameover.ogg");
              } catch (final IOException e) {
                    Debug.e("Error", e);
              }
```

```
        }

        if (options.getMusic() == true) {
              MusicFactory.setAssetBasePath("snd/");
              try {
                    this.music = MusicFactory.createMusicFromAsset(
                                this.mEngine.getMusicManager(), this,
"music.ogg");
                    this.music.setLooping(true);
              } catch (final IOException e) {
                    Debug.e("Error", e);
              }
        }

        if (GameActivity.options.getControlls().equals("1"))
              this.enableAccelerometerSensor(this);

  }

  @Override
  public Scene onLoadScene() {
        this.mEngine.registerUpdateHandler(new FPSLogger());

        mEngine.stop();
        gameInit();

        loadScene();

        return level.getScene();
  }

  public void showScore(){
        runOnUiThread(new Runnable() {

              @Override
              public void run() {

                    AlertDialog.Builder alert = new
AlertDialog.Builder(GameActivity.this);

                    alert.setTitle("LEVEL CLEARED");
                    alert.setIcon(R.drawable.trophy);
                    alert.setMessage("Score kamu:");

                    final TextView lblScore = new
TextView(GameActivity.this);
                    lblScore.setTextColor(Color.rgb(20,164,255));
                    lblScore.setTextSize(32f);
                    Typeface font =
Typeface.createFromAsset(getAssets(),"font/pf_tempesta_five.ttf");
                    lblScore.setTypeface(font);
                    lblScore.setPadding(0, 0, 0, 20);
                    lblScore.setGravity(Gravity.CENTER_HORIZONTAL);


  lblScore.setText(String.valueOf(GameActivity.score.getScore()));
                    alert.setView(lblScore);
```

```
                    alert.setPositiveButton("OK", new
DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int
whichButton) {
                          mEngine.start();
                    }
                    });
                    mEngine.stop();
                    alert.show();
              }
        });
  };

  private void goToNextLevel(){
        fade.addShapeModifier(new SequenceModifier(new
IShapeModifierListener() {

              @Override
              public void onModifierFinished(IShapeModifier
pShapeModifier, IShape pShape) {

                    showScore();

                    mEngine.clearUpdateHandlers();
                    level.getScene().clearUpdateHandlers();
                    level.getScene().clearChildScene();
                    level.getScene().clearTouchAreas();

                    level.next();
                    bulletsToReuse.clear();
                    enemiesToReuse.clear();
                    enemyBulletsToReuse.clear();
                    explosionsToReuse.clear();

                    loadScene();

                    fade.addShapeModifier(new SequenceModifier(new
IShapeModifierListener() {
                          @Override
                          public void onModifierFinished(IShapeModifier
pShapeModifier, IShape pShape) {
                                gameGetReady();
                          }
                    },new DelayModifier(1), new AlphaModifier(0.5f, 1,
0)));
              }
        },new AlphaModifier(0.5f, 0, 1)));
  }

  public void loadScene(){
        isGameOver = false;
        isGameReady = false;
        isBossFight = false;

        if(level.getLevel()!=0){
              mAutoScrollBackgroundTexture.clearTextureSources();
```

```
TextureRegionFactory.createFromAsset(mAutoScrollBackgroundTexture,this,
"background"+level.getLevel()+".png", 0, 0);
        }

        level.getScene().setBackground(ScrollBackground);
        level.getScene().getTopLayer().addEntity(okvir);
        level.getScene().getTopLayer().addEntity(healthbar);
        level.getScene().getTopLayer().addEntity(fade);

        level.getScene().registerUpdateHandler(new TimerHandler(5-
level.getLevel(), new ITimerCallback() {
              @Override
              public void onTimePassed(TimerHandler pTimerHandler) {
                    pTimerHandler.reset();

                    if (isGameReady == true){
                          if(score.getScore()>=100){
                                if(boss.isVisible()==false &&
enemies.size()==0 && isBossFight==false){
                                      isBossFight=true;
                                      gameWarning();
                                }
                          }else{

  if(level.getScene().getWaves().getCurrentWave()==null){
                                      if(enemies.size()==0){
                                            gameLevelCleared();
                                      }
                                }else{
                                      for(int
i=0;i<level.getScene().getWaves().getCurrentWave().size();i++){

  if(level.getScene().getWaves().getCurrentWave().getType(i).equals("ship
")){
                                                      for(int
j=0;j<level.getScene().getWaves().getCurrentWave().getCount(i);j++){

  createEnemyShip(paths[j]);
                                                      }
                                            }
                                      }
                                      level.getScene().getWaves().next();
                                }
                          }
                    }
              }
        }));

        level.getScene().registerUpdateHandler(new IUpdateHandler() {
              @Override
              public void reset() {}

              @Override
              public void onUpdate(float pSecondsElapsed) {
                    checkCollusions();
              }
```

```
        });

        ship.setPosition(CAMERA_WIDTH / 2 -
mShipTextureRegion.getTileWidth() / 2, CAMERA_HEIGHT + 300);
        ship.addToScene(level.getScene());

        if(options.getAutoFire()==true){
              level.getScene().registerUpdateHandler(new
TimerHandler(0.8f, new ITimerCallback() {
                    @Override
                    public void onTimePassed(final TimerHandler
pTimerHandler) {
                          pTimerHandler.reset();
                          if (isGameReady == true)
                                ship.fire();
                    }
              }));
        }

        if (GameActivity.options.getControlls().equals("3")) {

              final AnalogOnScreenControl analogOnScreenControl = new
AnalogOnScreenControl(CAMERA_WIDTH-
this.mOnScreenControlBaseTextureRegion.getWidth(), CAMERA_HEIGHT -
this.mOnScreenControlBaseTextureRegion.getHeight()-30, camera,
this.mOnScreenControlBaseTextureRegion,
this.mOnScreenControlKnobTextureRegion, 0.1f, 200, new
IAnalogOnScreenControlListener() {
              @Override
              public void onControlChange(final BaseOnScreenControl
pBaseOnScreenControl, final float pValueX, final float pValueY) {
                      ship.setVelocity(pValueX * 150, pValueY * 150);
              }

              @Override
              public void onControlClick(final AnalogOnScreenControl
pAnalogOnScreenControl) {}
            });

            analogOnScreenControl.getControlBase().setAlpha(0.5f);
            analogOnScreenControl.getControlBase().setScaleCenter(0,
128);
            analogOnScreenControl.getControlBase().setScale(0.75f);
            analogOnScreenControl.getControlKnob().setScale(0.75f);
            analogOnScreenControl.refreshControlKnobPosition();

            level.getScene().setChildScene(analogOnScreenControl);
        }
  }

  @Override
  public void onLoadComplete() {
        runOnUiThread(new Runnable() {
              @Override
              public void run() {
                    scoreView.setText("SCORE: 0");
              }
```

```
        });

        if (options.getMusic() == true) {
              music.play();
        }
        mEngine.start();

        fade.addShapeModifier(new SequenceModifier(new
IShapeModifierListener() {
              @Override
              public void onModifierFinished(IShapeModifier
pShapeModifier, IShape pShape) {
                    gameGetReady();
              }
        },new DelayModifier(1), new AlphaModifier(0.5f, 1, 0)));


  }

  @Override
  public void onAccelerometerChanged(AccelerometerData accelerometer) {
        if (isGameReady == true)
              ship.move(accelerometer);
        else
              ship.setVelocity(0);
  }

  @Override
  public boolean onKeyDown(final int pKeyCode, final KeyEvent pEvent) {
        if (isGameReady == true) {

              if (GameActivity.options.getControlls().equals("2")) {
                    if (pKeyCode == KeyEvent.KEYCODE_DPAD_LEFT
                                && pEvent.getAction() ==
KeyEvent.ACTION_DOWN) {
                          ship.setVelocity(-100, 0);
                          return true;
                    } else if (pKeyCode == KeyEvent.KEYCODE_DPAD_RIGHT
                                && pEvent.getAction() ==
KeyEvent.ACTION_DOWN) {
                          ship.setVelocity(100, 0);
                          return true;
                    } else if (pKeyCode == KeyEvent.KEYCODE_DPAD_UP
                                && pEvent.getAction() ==
KeyEvent.ACTION_DOWN) {
                          ship.setVelocity(0, -100);
                          return true;
                    } else if (pKeyCode == KeyEvent.KEYCODE_DPAD_DOWN
                                && pEvent.getAction() ==
KeyEvent.ACTION_DOWN) {
                          ship.setVelocity(0, 100);
                          return true;
                    }
              }

              if(options.getAutoFire()==false)
                    if (pKeyCode == KeyEvent.KEYCODE_DPAD_CENTER
```

```
                                && pEvent.getAction() ==
KeyEvent.ACTION_DOWN) {
                            ship.fire();
                            return true;
                        }
                }

                if (pKeyCode == KeyEvent.KEYCODE_BACK
                            && pEvent.getAction() == KeyEvent.ACTION_DOWN) {
                      showExitDialog();
                      return true;
                }

                return super.onKeyDown(pKeyCode, pEvent);
          }

          public void gameInit() {

                score.reset();

                enemies = new ArrayList<EnemyShip>();
                enemiesToReuse = new ArrayList<EnemyShip>();

                bullets = new ArrayList<Bullet>();
                bulletsToReuse = new ArrayList<Bullet>();

                enemyBullets = new ArrayList<EnemyBullet>();
                enemyBulletsToReuse = new ArrayList<EnemyBullet>();

                explosions = new ArrayList<Explosion>();
                explosionsToReuse = new ArrayList<Explosion>();

                paths[0] = new Path(7).to(50, -60).to(50, 150).to(400,
150).to(300, 250).to(400, 500).to(150, 400).to(400, -60);
                paths[1] = new Path(7).to(100, -60).to(250, 100).to(100,
400).to(300, 300).to(400, 400).to(250, 100).to(300, -60);
                paths[2] = new Path(7).to(300, -60).to(400, 120).to(300,
100).to(200, 400).to(200, 320).to(150, 300).to(200, -60);
                paths[3] = new Path(7).to(400, -60).to(200, 50).to(50,
300).to(300, 400).to(150, 400).to(300, 200).to(100, -60);

                level = new Level(this);
                ScrollBackground = new ScrollBackground(new Sprite(0,
CAMERA_HEIGHT - this.mScrollLayer.getHeight(),this.mScrollLayer), 100);

                fade = new Rectangle(0, 0, 480, 800);
                fade.setColor(0, 0, 0);
                fade.setAlpha(1);

                boss = new Boss(200, -200, mBossTextureRegion,
mLaserTextureRegion, getEngine());

                okvir = new Rectangle(310, 30, 140, 18);
                okvir.setColor(0.1f, 0.1f, 0.1f);

                healthbar = new Rectangle(314, 34, 132, 10);
                healthbar.setColor(1, 0, 0);
```

```
        ship = new PlayerShip(CAMERA_WIDTH / 2 -
mShipTextureRegion.getTileWidth() / 2, CAMERA_HEIGHT + 300,
mShipTextureRegion, getEngine());
   }

   public void gameGetReady() {
         ship.addShapeModifier(new SequenceModifier(
                     new IShapeModifierListener() {
                           @Override
                           public void onModifierFinished(
                                       IShapeModifier pShapeModifier,
final IShape enemy) {
                                 runOnUpdateThread(new Runnable() {
                                       public void run() {

                                             final Text textCenter = new
Text(
                                                         CAMERA_WIDTH / 2
- 150,
                                                         CAMERA_HEIGHT / 2
- 60, gameOver_font,
                                                         "GET\nREADY",
HorizontalAlign.CENTER);

   mEngine.getScene().getTopLayer()

   .addEntity(textCenter);
                                             textCenter

   .addShapeModifier(new SequenceModifier(
                                                                     new
IShapeModifierListener() {


   @Override

   public void onModifierFinished(

         IShapeModifier pShapeModifier,

         IShape pShape) {

   GameActivity.isGameReady = true;

   }
                                                                     },
new ScaleModifier(1, 0, 1),
                                                                     new
DelayModifier(1),
                                                                     new
ScaleModifier(1, 1, 0)));

                                       }
                                 });
                           }
                     }, new PathModifier(2, new Path(2).to(
```

```
                                    CAMERA_WIDTH / 2 -
mShipTextureRegion.getTileWidth()
                                                / 2, CAMERA_HEIGHT + 100).to(
                                    CAMERA_WIDTH / 2 -
mShipTextureRegion.getTileWidth()
                                                / 2, CAMERA_HEIGHT - 160),
EaseSineInOut
                                    .getInstance())));
   }

   public void gameLevelCleared() {
         final Text textCenter = new Text(
                     CAMERA_WIDTH / 2 - 215,
                     CAMERA_HEIGHT / 2 - 60, gameOver_font,
                     "LEVEL\nCLEARED", HorizontalAlign.CENTER);
         mEngine.getScene().getTopLayer()
                     .addEntity(textCenter);
         textCenter
                     .addShapeModifier(new SequenceModifier(
                                 new IShapeModifierListener() {
                                       @Override
                                       public void onModifierFinished(
                                                   IShapeModifier
pShapeModifier,
                                                   IShape pShape) {
                                             GameActivity.isGameReady =
false;
                                             ship.addShapeModifier(new
SequenceModifier(
                                                         new
IShapeModifierListener() {
                                                               @Override
                                                               public void
onModifierFinished(

   IShapeModifier pShapeModifier, final IShape enemy) {

   runOnUpdateThread(new Runnable() {

   public void run() {

   //finish();

   goToNextLevel();

   }
                                                                     });
                                                               }
                                                         }, new
PathModifier(2, new Path(2).to(ship.getX(),ship.getY()).to(

   ship.getX(), -100), EaseSineInOut

   .getInstance())));
                                       }
                                 }, new ScaleModifier(1, 0, 1),
                                 new DelayModifier(1),
```

```
                              new ScaleModifier(1, 1, 0)));
  }

  public void gameWarning() {

        final Text textCenter = new Text(
                    CAMERA_WIDTH / 2 - 195,
                    CAMERA_HEIGHT / 2 - 60, font,
                    "WARNING", HorizontalAlign.CENTER);
        mEngine.getScene().getTopLayer()
                    .addEntity(textCenter);

        textCenter
                    .addShapeModifier(new SequenceModifier(
                              new IShapeModifierListener() {

                                    @Override
                                    public void onModifierFinished(
                                                IShapeModifier
pShapeModifier,
                                                IShape pShape) {
                                          //GameActivity.isBossFight =
true;

                                          boss.addToScene();
                                          boss.fight();
                                    }
                              }, new ScaleModifier(1, 0, 1),
                              new DelayModifier(1),
                              new RotationByModifier(1.5f, 360),
                              new DelayModifier(1),
                              new ScaleModifier(1, 1, 0)));
  }

  public void createEnemyShip(Path path) {

        if (enemiesToReuse.isEmpty()) {
              final EnemyShip enemy = new EnemyShip(CAMERA_WIDTH / 2
                          - mShipTextureRegion.getTileWidth() / 2, -50,
                          mEnemyTextureRegion, getEngine());
              enemy.addToScene();
              enemy.folow(path, 10);
        } else {
              final EnemyShip enemy = EnemyShip.reuse();
              enemy.addToScene();
              enemy.folow(path, 10);
        }
  }

  public void checkCollusions() {

        // COLLUSION BETWEEN SHIP AND ENEMY BULLETS
        if (enemyBullets.size() > 0) {
              for (int i = 0; i < enemyBullets.size(); i++) {
                    final EnemyBullet bullet = enemyBullets.get(i);
                    if
(BaseCollisionChecker.checkAxisAlignedRectangleCollision(
```

```
                                    bullet.getX(), bullet.getY(),
                                    bullet.getX() + bullet.getWidth(),
bullet.getY()
                                                + bullet.getHeight(),
ship.getX(), ship.getY(),
                                    ship.getX() + ship.getWidth(),
                                    ship.getY() + ship.getHeight()) == true)
{
                              runOnUpdateThread(new Runnable() {
                                    public void run() {

                                          bullet.removeFromScene();
                                          ship.hit();


                                          // hit point
                                          Debug.d("HEALTH >
"+ship.getHealth());
                                          if(ship.getHealth()>=0){

   healthbar.setWidth((ship.getHealth()+1)*35);
                                          }else{

   healthbar.setWidth((ship.getHealth()+1)*35);
                                                healthbar.setVisible(false);
                                                okvir.setVisible(false);
                                          }

                                          if
(GameActivity.options.getVibration() == true)
                                                vibrate();
                                          if (ship.getHealth() < 0) {
                                                if
(GameActivity.options.getSoundEffects() == true)
                                                      explosionSound.play();
                                                makeExplosion(ship.getX(),
ship.getY());
                                                showGameOver();
                                          }
                                    }
                              });
                        }
                  }
            }

            if(boss!=null && boss.getLaser()!=null){
                  if(boss.getLaser().isVisible()==true)
                  {
                        if
((BaseCollisionChecker.checkAxisAlignedRectangleCollision(
                                    boss.getLaser().getX(),
boss.getLaser().getY(),
                                    boss.getLaser().getX() +
boss.getLaser().getWidth(), boss.getLaser().getY()
                                                +
boss.getLaser().getHeight(), ship.getX(), ship.getY(),
                                    ship.getX() + ship.getWidth(),
```

```
                                    ship.getY() + ship.getHeight()) == true
&& ship.isKilled()==false) ||

   (BaseCollisionChecker.checkAxisAlignedRectangleCollision(
                                                boss.getLaser2().getX(),
boss.getLaser2().getY(),
                                                boss.getLaser2().getX() +
boss.getLaser2().getWidth(), boss.getLaser2().getY()
                                                                        +
boss.getLaser2().getHeight(), ship.getX(), ship.getY(),
                                                ship.getX() +
ship.getWidth(),
                                                ship.getY() +
ship.getHeight()) == true && ship.isKilled()==false)
                        ) {
                              runOnUpdateThread(new Runnable() {
                                    public void run() {
                                          ship.kill();

                                          if
(GameActivity.options.getVibration() == true)
                                                vibrate();

                                          if
(GameActivity.options.getSoundEffects() == true)
                                                      explosionSound.play();

                                          makeExplosion(ship.getX(),
ship.getY());
                                          showGameOver();
                                    }
                              });
                        }
                  }
            }

            // COLLUSION BETWEEN ENEMY AND SHIP BULLETS
            if (bullets.size() > 0) {

                        for (int i = 0; i < bullets.size(); i++) {
                              final Bullet bullet = bullets.get(i);

                              if (enemies.size() > 0) {
                                    for (int j = 0; j < enemies.size(); j++)
{
                                          final EnemyShip enemy =
enemies.get(j);
                                          if (BaseCollisionChecker

   .checkAxisAlignedRectangleCollision(

   bullet.getX(), bullet.getY(),

   bullet.getX() + bullet.getWidth(),

   bullet.getY() + bullet.getHeight(),
```

```
   enemy.getX(), enemy.getY(),
   enemy.getX() + enemy.getWidth(),
   enemy.getY() + enemy.getHeight()) == true
                                                      && !enemy.isKilled()) {
                                                runOnUpdateThread(new
Runnable() {
                                                      public void run() {
   score.addPoints();
                                                            refreshScore();
   bullet.removeFromScene();
                                                            if
(GameActivity.options.getSoundEffects() == true)
   explosionSound.play();
   makeExplosion(enemy.getX(), enemy.getY());
   enemy.removeFromScene();
                                                      }
                                                });
                                          }
                                    }
                              }
                              if (BaseCollisionChecker
   .checkAxisAlignedRectangleCollision(
                                                      bullet.getX(),
bullet.getY(),
                                                      bullet.getX() +
bullet.getWidth(),
                                                      bullet.getY() +
bullet.getHeight(),
                                                      boss.getX(),
boss.getY(), boss.getX()
                                                            +
boss.getWidth(), boss.getY()
                                                            +
boss.getHeight()) == true
                                          && !boss.isKilled()) {
                                    runOnUpdateThread(new Runnable() {
                                          public void run() {
                                                score.addPoints();
                                                refreshScore();
                                                bullet.removeFromScene();
                                                if
(GameActivity.options.getSoundEffects() == true)
                                                      explosionSound.play();
                                                makeExplosion(boss.getX()+15,
boss.getY()+15);
```

```
                                          boss.addShapeModifier(new
SequenceModifier(new ColorModifier(0.3f, 1, 1, 1, 0, 1, 0), new
ColorModifier(0.3f, 1, 1, 0, 1, 0, 1)));
                                    }
                              });
                        }
                  }
            }
      }

      public void refreshScore(){
            runOnUiThread(new Runnable() {
                  @Override
                  public void run() {
                        GameActivity.scoreView.setText("SCORE:
"+score.getScore());
                  }
            });
      }

      public void makeExplosion(float posx, float posy) {
            if (explosionsToReuse.isEmpty())
                  new Explosion(posx, posy, mExplosionTextureRegion,
getEngine());
            else
                  Explosion.reuse(posx, posy);
      }

      public void vibrate() {
            // script yg digunakan untuk menjalankan fungsi vibrate
            // <uses-permission android:name="android.permission.VIBRATE"/>
            final Vibrator vibrator = (Vibrator)
getSystemService(Context.VIBRATOR_SERVICE);
            vibrator.vibrate(500);
      }

      public void showGameOver() {
            ship.addShapeModifier(new SequenceModifier(
                        new IShapeModifierListener() {
                              @Override
                              public void onModifierFinished(
                                          IShapeModifier pShapeModifier,
IShape pShape) {
                                    runOnUpdateThread(new Runnable() {
                                          public void run() {
                                                isGameOver = true;

                                                if
(GameActivity.options.getSoundEffects() == true)

      GameActivity.gameOverSound.play();

                                                ship.removeFromScene();
                                                ship.setPosition(-20,
ship.getAccelerationY()); // IZBACI GA SA VIDA SCENE
```

```
                                                final Text textCenter = new
Text(
                                                            CAMERA_WIDTH / 2
- 120,
                                                            CAMERA_HEIGHT / 2
- 60, gameOver_font,
                                                            "GAME\nOVER",
HorizontalAlign.CENTER);
   mEngine.getScene().getTopLayer()
   .addEntity(textCenter);
                                                textCenter
   .addShapeModifier(new SequenceModifier(
                                                                        new
IShapeModifierListener() {

   @Override
   public void onModifierFinished(
         IShapeModifier pShapeModifier,
         IShape pShape) {
   ship.clearShapeModifiers();
   if (options.getMusic() == true) {
         music.stop();
   }

   mEngine.stop();

   final Highscore highscore = new Highscore(getApplicationContext());

   if(highscore.inHighscore(score.getScore()))
   {
   runOnUiThread(new Runnable() {

         @Override
         public void run() {
```

```
            AlertDialog.Builder alert = new
AlertDialog.Builder(GameActivity.this);


            alert.setTitle("HIGHSCORE");

            alert.setIcon(R.drawable.trophy);

            alert.setMessage("Score : "+score.getScore()+"\nMasukkan
namamu:");


            final LinearLayout layout = new
LinearLayout(GameActivity.this);

            final EditText input = new EditText(GameActivity.this);

            layout.setLayoutParams(new
LayoutParams(LayoutParams.FILL_PARENT,LayoutParams.FILL_PARENT));

            input.setLayoutParams(new
LayoutParams(LayoutParams.FILL_PARENT,LayoutParams.FILL_PARENT));

            layout.setPadding(20, 0, 20, 0);

            layout.addView(input);



            alert.setView(layout);

            alert.setPositiveButton("Ok", new
DialogInterface.OnClickListener() {

                  public void onClick(DialogInterface dialog, int
whichButton) {

                    String value = input.getText().toString();



                        highscore.addScore(value,score.getScore());

                        String url =
"https://docs.google.com/forms/d/1i5gfrqlHPrKUOw45p7wUY-HRo-
3WQ29Rb7CXYBOJY8M/viewform";

                        Intent i = new Intent(Intent.ACTION_VIEW);

                        i.setData(Uri.parse(url));

                        startActivity(i);

                        finish();

                    }
```

```
                  });

            alert.show();

      }

  });

  }else{

      finish();

  }


  }
                                                                        },
new ScaleModifier(1, 0, 1),
                                                                        new
DelayModifier(2),
                                                                        new
ScaleModifier(1, 1, 0),
                                                                        new
DelayModifier(1)));
                                    }
                              });
                        }
                  }, new ScaleModifier(0.3f, 1, 0)));

      //ship.getShadow().addShapeModifier(new SequenceModifier(new
ScaleModifier(0.3f, 1, 0)));
  }

  public void showExitDialog() {
      if (options.getMusic() == true) {
            music.pause();
      }
      mEngine.stop();
      AlertDialog.Builder builder = new AlertDialog.Builder(this);
      builder.setMessage("Yakin pengen keluar?")
                  .setCancelable(false)
                  .setTitle("EXIT")
                  .setPositiveButton("Yes",
                              new DialogInterface.OnClickListener() {
                                    public void onClick(DialogInterface
dialog, int id) {
                                          if (options.getMusic() ==
true) {
                                                music.stop();
                                          }
                                          finish();
                                    }
                              })
                  .setNegativeButton("No", new
DialogInterface.OnClickListener() {
```

```
                              public void onClick(DialogInterface dialog, int
id) {
                                    dialog.cancel();
                                    if (options.getMusic() == true) {
                                          music.play();
                                    }
                                    mEngine.start();
                              }
                        });
          AlertDialog alert = builder.create();
          alert.setIcon(R.drawable.icon);
          alert.show();
    }
}
```

4. HelpActivity.java

```
package skripsi.akakom.peranghelikopter.game;

import skripsi.akakom.peranghelikopter.game.R;

import android.app.Activity;
import android.os.Bundle;

public class HelpActivity extends Activity {

      @Override
      public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.help);

          //TextView help = (TextView) findViewById(R.id.txtHelp);
        //Typeface font =
Typeface.createFromAsset(getAssets(),"font/pf_tempesta_five.ttf");

            //help.setTypeface(font);
      }
}
```

5. Highscore.java

```
package skripsi.akakom.peranghelikopter.game;

import android.content.Context;
import android.content.SharedPreferences;

public class Highscore {

  private SharedPreferences preferences;
  private String names[];
  private long score[];

  public Highscore(Context context) {
        preferences = context.getSharedPreferences("Highscore", 0);
        names = new String[10];
```

```
        score = new long[10];
        for (int x = 0; x < 10; x++) {
              names[x] = preferences.getString("name" + x, "-");
              score[x] = preferences.getLong("score" + x, 0);
        }
  }

  public String getName(int x) {
        // Mendapatkan nama urutan ke x dalam daftar highscore
        return names[x];
  }

  public long getScore(int x) {
        // Mendapatkan nilai score untuk daftar list Highscore
        return score[x];
  }

  public boolean inHighscore(long score) {
        //Mengetes apak masuk dalam daftar highscore atau tidak
        int position;
        for (position = 0; position < 10 && this.score[position] >=
score; position++)
              ;
        if (position == 10)
              return false;
        return true;
  }

  public boolean addScore(String name, long score) {
        // Menambahkan nama dan score kedalam list highscore
        int position;
        for (position = 0; position < 10 && this.score[position] > score;
position++);

        if (position == 10)
              return false;
        for (int x = 9; x > position; x--) {
              names[x] = names[x - 1];
              this.score[x] = this.score[x - 1];
        }

        this.names[position] = new String(name);
        this.score[position] = score;
        SharedPreferences.Editor editor = preferences.edit();
        for (int x = 0; x < 10; x++) {
              editor.putString("name" + x, this.names[x]);
              editor.putLong("score" + x, this.score[x]);
        }
        editor.commit();
        return true;
  }
}
```

6. MenuActivity.java

```
package skripsi.akakom.peranghelikopter.game;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Typeface;
import android.net.Uri;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.TextView;

public class MenuActivity extends Activity {

	@Override
	public void onCreate(Bundle savedInstanceState) {
	    super.onCreate(savedInstanceState);
	    setContentView(R.layout.main_new);

	    Typeface font =
Typeface.createFromAsset(getAssets(),"font/pf_tempesta_five.ttf");

	      TextView txtView = (TextView) findViewById(R.id.copyright);
	      txtView.setTypeface(font);

	      TextView btnHelp = (TextView) findViewById(R.id.btnHelp);
	      btnHelp.setTypeface(font);

	        btnHelp.setOnClickListener(new OnClickListener() {

	              @Override
	              public void onClick(View v) {
	                    Intent help = new Intent(MenuActivity.this,
HelpActivity.class);
	                    startActivity(help);
	              }
	        });

	      TextView btnExit = (TextView) findViewById(R.id.btnExit);
	    btnExit.setTypeface(font);
	    btnExit.setOnClickListener(new OnClickListener() {

	              @Override
	              public void onClick(View v) {
	                    showExitDialog();
	              }
	        });

	    TextView btnAbout = (TextView) findViewById(R.id.btnAbout);
	    btnAbout.setTypeface(font);
	    btnAbout.setOnClickListener(new OnClickListener() {

	              @Override
```

```
                    public void onClick(View v) {
                          AlertDialog.Builder alertbox = new
AlertDialog.Builder(MenuActivity.this);
                          alertbox.setTitle("ABOUT");
                          alertbox.setIcon(R.drawable.icon);
                          alertbox.setCancelable(false);
                    alertbox.setMessage("Skripsi Game Perang
Helikopter\n" +
                                                "Dibuat Oleh Muchammad
Ali Nur Secha\n " +
                                                "NIM : 105410354
Jurusan Teknik Informatika Akakom\n" +
                                                "Tahun 2013");

                    alertbox.setPositiveButton("Close",
                                          new
DialogInterface.OnClickListener() {
                                                public void
onClick(DialogInterface dialog, int id) {
                                                }
                                          });
                    alertbox.show();
                    }
              });

          TextView btnStartGame = (TextView)
findViewById(R.id.btnStartGame);
          btnStartGame.setTypeface(font);
          btnStartGame.setOnClickListener(new OnClickListener() {

                    @Override
                    public void onClick(View v) {
                          Intent game = new Intent(MenuActivity.this,
GameActivity.class);
                          startActivity(game);
                    }
              });

          TextView btnOptions = (TextView) findViewById(R.id.btnOptions);
          btnOptions.setTypeface(font);
          btnOptions.setOnClickListener(new OnClickListener() {

                    @Override
                    public void onClick(View v) {
                          Intent game = new Intent(MenuActivity.this,
OptionsActivity.class);
                          startActivity(game);
                    }
              });

          TextView btnAngket = (TextView) findViewById(R.id.btnAngket);
          btnAngket.setTypeface(font);
          btnAngket.setOnClickListener(new OnClickListener() {

                    @Override
                    public void onClick(View v) {
```

```
                        String url =
"https://docs.google.com/forms/d/1i5gfrqlHPrKUOw45p7wUY-HRo-
3WQ29Rb7CXYBOJY8M/viewform";
                        Intent i = new Intent(Intent.ACTION_VIEW);
                        i.setData(Uri.parse(url));
                        startActivity(i);
                  }
            });

          TextView btnHighscore = (TextView)
findViewById(R.id.btnHighscore);
          btnHighscore.setTypeface(font);
          btnHighscore.setOnClickListener(new OnClickListener() {

                  @Override
                  public void onClick(View v) {
                        AlertDialog.Builder alertbox = new
AlertDialog.Builder(MenuActivity.this);
                        alertbox.setTitle("HIGHSCORE");
                        alertbox.setIcon(R.drawable.trophy);
                        alertbox.setCancelable(false);

                        Highscore score = new
Highscore(MenuActivity.this);
                        StringBuilder str = new StringBuilder();

                        for(int i=0;i<10;i++){
                              if(i!=9)
                                    str.append(i+1 +"  .
"+score.getName(i)+"  ("+score.getScore(i)+" points)\n");
                              else
                                    str.append(i+1 +".
"+score.getName(i)+"  ("+score.getScore(i)+" points)\n");
                        }

                  alertbox.setMessage(str.toString());

                  alertbox.setPositiveButton("Close",
                                    new
DialogInterface.OnClickListener() {
                                          public void
onClick(DialogInterface dialog, int id) {
                                          }
                                    });

                  alertbox.show();

                  }
            });

      }


      @Override
        public boolean onKeyDown(final int pKeyCode, final KeyEvent
pEvent) {
```

```
                if (pKeyCode == KeyEvent.KEYCODE_BACK
                            && pEvent.getAction() == KeyEvent.ACTION_DOWN)
    {
                      showExitDialog();
                      return true;
                }

                return super.onKeyDown(pKeyCode, pEvent);
          }

         public void showExitDialog() {
           AlertDialog.Builder builder = new
    AlertDialog.Builder(MenuActivity.this);
                builder.setMessage("Yakin pengen keluar?")
                            .setCancelable(false)
                            .setTitle("EXIT")
                            .setPositiveButton("Yes",
                                        new
    DialogInterface.OnClickListener() {
                                                public void
    onClick(DialogInterface dialog, int id) {

      MenuActivity.this.finish();
                                                }
                                        })
                            .setNegativeButton("No", new
    DialogInterface.OnClickListener() {
                                    public void onClick(DialogInterface
    dialog, int id) {
                                          dialog.cancel();
                                    }
                            });

                AlertDialog alert = builder.create();
                alert.setIcon(R.drawable.icon);
                alert.show();
         }
    }
```

7. OptionsActivity.java

```
package skripsi.akakom.peranghelikopter.game;

import skripsi.akakom.peranghelikopter.game.R;

import android.os.Bundle;
import android.preference.PreferenceActivity;

public class OptionsActivity extends PreferenceActivity {

  @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
       addPreferencesFromResource(R.layout.options);
```

```
    }

}
```

8. ScrollBackground.java

```
package skripsi.akakom.peranghelikopter.game.background;

import javax.microedition.khronos.opengles.GL10;

import org.anddev.andengine.engine.camera.Camera;
import org.anddev.andengine.entity.scene.background.ColorBackground;
import org.anddev.andengine.entity.shape.Shape;

public class ScrollBackground extends ColorBackground {

        private final Shape mShape;
        private final float mScrollChangePerSecond;
        protected float mScrollValue;

        public ScrollBackground(final Shape pShape, final float
pScrollChangePerSecond) {
              super(0, 0, 0);
              this.mScrollChangePerSecond = pScrollChangePerSecond;
              mShape = pShape;
        }

        public void setScrollValue(final float pScrollValue) {
              this.mScrollValue = pScrollValue;
        }

        @Override
        public void onUpdate(final float pSecondsElapsed) {
              super.onUpdate(pSecondsElapsed);

              this.mScrollValue += this.mScrollChangePerSecond *
pSecondsElapsed;
        }

        @Override
        public void onDraw(final GL10 pGL, final Camera pCamera) {
              super.onDraw(pGL, pCamera);

              pGL.glPushMatrix();
              {
                    final float cameraHeight = pCamera.getHeight();
                    final float shapeHeightScaled = 800;
                    float baseOffset = mScrollValue % shapeHeightScaled;

                    while(baseOffset > 0) {
                          baseOffset -= shapeHeightScaled;
                    }
                    pGL.glTranslatef(0, baseOffset, 0);

                    float currentMaxY = baseOffset;
```

```
                        do {
                              this.mShape.onDraw(pGL, pCamera);
                              pGL.glTranslatef(0, shapeHeightScaled, 0);
                              currentMaxY += shapeHeightScaled;
                        } while(currentMaxY < cameraHeight);
                  }
                  pGL.glPopMatrix();
            }
      }
```

9. Explosion.java

```
package skripsi.akakom.peranghelikopter.game.effects;
import org.anddev.andengine.engine.Engine;
import org.anddev.andengine.entity.sprite.AnimatedSprite;
import org.anddev.andengine.opengl.texture.region.TiledTextureRegion;

import skripsi.akakom.peranghelikopter.game.GameActivity;

public class Explosion extends AnimatedSprite {

   private final Engine mEngine;

   public Explosion(float pX, float pY, TiledTextureRegion
pTiledTextureRegion, final Engine engine) {
         super(pX, pY, pTiledTextureRegion);
         mEngine = engine;
         GameActivity.explosions.add(this);
         addToScene();
         explode();
   }

   public static Explosion reuse(float posx, float posy) {
         final Explosion explosion =
GameActivity.explosionsToReuse.get(0);
         GameActivity.explosions.add(explosion);
         GameActivity.explosionsToReuse.remove(explosion);
         explosion.setPosition(posx, posy);
         explosion.explode();
         explosion.setVisible(true);
         return explosion;
   }

   public void addToScene() {
         mEngine.getScene().getTopLayer().addEntity(this);
   }

   public void explode() {
         this.animate(GameActivity.ANIMATION_FRAMELENGTH1, false,
                     new IAnimationListener() {
                           @Override
                           public void onAnimationEnd(final AnimatedSprite
explosion) {
                                 explosion.setVisible(false);

   GameActivity.explosionsToReuse.add((Explosion) explosion);
```

```
      GameActivity.explosions.remove(explosion);
                              }
                  }
            );
      }

   }
```

10. Boss.java

```
   package skripsi.akakom.peranghelikopter.game.enemies;
   import org.anddev.andengine.engine.Engine;
   import org.anddev.andengine.engine.handler.timer.ITimerCallback;
   import org.anddev.andengine.engine.handler.timer.TimerHandler;
   import org.anddev.andengine.entity.shape.IShape;
   import org.anddev.andengine.entity.shape.modifier.IShapeModifier;
   import
   org.anddev.andengine.entity.shape.modifier.IShapeModifier.IShapeModifierLi
   stener;
   import org.anddev.andengine.entity.shape.modifier.LoopModifier;
   import org.anddev.andengine.entity.shape.modifier.PathModifier;
   import
   org.anddev.andengine.entity.shape.modifier.PathModifier.IPathModifierListe
   ner;
   import org.anddev.andengine.entity.shape.modifier.ScaleModifier;
   import org.anddev.andengine.entity.shape.modifier.SequenceModifier;
   import org.anddev.andengine.entity.shape.modifier.ease.EaseSineInOut;
   import org.anddev.andengine.entity.sprite.AnimatedSprite;
   import org.anddev.andengine.opengl.texture.region.TextureRegion;
   import org.anddev.andengine.opengl.texture.region.TiledTextureRegion;
   import org.anddev.andengine.util.Path;

   import skripsi.akakom.peranghelikopter.game.GameActivity;

   public class Boss extends AnimatedSprite implements IEnemy {

      private final Engine mEngine;
      private boolean killed;
      private TextureRegion laserRegion;
      private Laser laser;
      private Laser laser2;
      private boolean i;
      private boolean uleteo;

      public Boss(float pX, float pY, TiledTextureRegion pTiledTextureRegion,
   TextureRegion laserRegion, final Engine engine) {
            super(pX, pY, pTiledTextureRegion);
            this.laserRegion = laserRegion;
            mEngine = engine;
            killed=false;
            uleteo=false;
            this.setVisible(false);
      }

      public boolean isKilled(){
```

```
        return killed;
    }

    public void fight(){
        this.setVisible(true);
        this.animate(GameActivity.ANIMATION_FRAMELENGTH1);
    }

    public Laser getLaser(){
        return laser;
    }

    public Laser getLaser2(){
        return laser2;
    }

    @Override
    public void addToScene() {
        this.laser = new Laser(this.getX()+this.getWidth()/2,
this.getY()+this.getHeight(), laserRegion, mEngine);
        this.laser2 = new Laser(this.getX()+this.getWidth()/2,
laser.getY()+laser.getHeight(), laserRegion, mEngine);
        laser.addToScene();
        laser2.addToScene();
        laser.setVisible(false);
        laser2.setVisible(false);
        i=true;

        mEngine.getScene().registerUpdateHandler(new TimerHandler(4f,
                    new ITimerCallback() {
              @Override
              public void onTimePassed(final TimerHandler pTimerHandler)
{
                    pTimerHandler.reset();
                    if(uleteo==true){
                          if(i==true){
                                laser.setVisible(true);
                                laser2.setVisible(true);
                                i=false;
                          }else{
                                laser.setVisible(false);
                                laser2.setVisible(false);
                                i=true;
                          }
                    }
              }
        }));


        mEngine.getScene().getBottomLayer().addEntity(this);
        Path start = new Path(2).to(200, -200).to(200, 50);

        this.addShapeModifier(new PathModifier(2f, start,
                    null, new IPathModifierListener() {
                          @Override
                          public void onWaypointPassed(
                                      final PathModifier pPathModifier,
```

```
                                        final IShape pShape, final int
pWaypointIndex) {

                                  if (pWaypointIndex == 1) {
                                        uleteo=true;
                                        mEngine.runOnUpdateThread(new
Runnable() {
                                              public void run() {
                                                    //Path p = new
Path(7).to(200, 100).to(250, 100).to(100, 400)
                                                    //.to(300, 300).to(400,
400).to(250, 100).to(200, 200);

                                                    Path p = new
Path(11).to(200, 50).to(50, 150).to(150, 250)
                                                    .to(250, 250).to(350,
150).to(200, 50).to(350, 150).to(250, 250).to(150, 250).to(50,
150).to(200, 50);

                                                    ((Boss)
pShape).addShapeModifier(new LoopModifier(new PathModifier(15, p,
                                                                null, new
IPathModifierListener() {
                                                          @Override
                                                          public void
onWaypointPassed(
                                                                final
PathModifier pPathModifier,
                                                                final
IShape pShape, final int pWaypointIndex) {

                                                                ((Boss)
pShape).fire();
                                                          }
                                                    },
EaseSineInOut.getInstance())));
                                              }
                                        });
                                  }

                            }
                      }, EaseSineInOut.getInstance()));
   }

   @Override
   public void removeFromScene() {
         killed=true;
         this.clearShapeModifiers();
         this.addShapeModifier(new SequenceModifier(
                     new IShapeModifierListener() {
                           @Override
                           public void onModifierFinished(
                                 IShapeModifier pShapeModifier,
final IShape boss) {
                                 mEngine.runOnUpdateThread(new Runnable()
{
                                       public void run() {
```

```
      mEngine.getScene().getBottomLayer().removeEntity(((Boss) boss));
                                        }
                                  });
                            }
                      }, new ScaleModifier(0.3f, 1, 0)));
      }

      @Override
      public void fire() {

      }

      @Override
      protected void onManagedUpdate(final float pSecondsElapsed) {
            laser.setPosition(this.getX()+this.getWidth()/2-5,
this.getY()+this.getHeight()-15);
            laser2.setPosition(laser.getX(), laser.getY()+laser.getHeight()-
4);
            super.onManagedUpdate(pSecondsElapsed);
      }

}
```

11. EnemyBullet.java

```
package skripsi.akakom.peranghelikopter.game.enemies;


import org.anddev.andengine.engine.Engine;
import org.anddev.andengine.entity.sprite.Sprite;
import org.anddev.andengine.opengl.texture.region.TextureRegion;

import skripsi.akakom.peranghelikopter.game.GameActivity;

public class EnemyBullet extends Sprite {

      private final Engine mEngine;

      public EnemyBullet(float pX, float pY,
                  TextureRegion pTiledTextureRegion, Engine engine) {
            super(pX, pY, pTiledTextureRegion);
            mEngine = engine;
            this.setVelocity(0, 300);
            GameActivity.enemyBullets.add(this);
      }

      public static EnemyBullet reuse(float posx, float posy) {
            final EnemyBullet bullet =
GameActivity.enemyBulletsToReuse.get(0);
            GameActivity.enemyBullets.add(bullet);
            GameActivity.enemyBulletsToReuse.remove(bullet);
            bullet.setPosition(posx, posy);
            return bullet;
      }
```

```
    public void addToScene() {
          mEngine.getScene().getBottomLayer().addEntity(this);
    }

    public void removeFromScene() {
          mEngine.getScene().getBottomLayer().removeEntity(this);
          GameActivity.enemyBulletsToReuse.add(0, this);
          GameActivity.enemyBullets.remove(this);
    }

    @Override
    protected void onManagedUpdate(final float pSecondsElapsed) {
          final EnemyBullet bullet = this;
          if (this.getY() > GameActivity.CAMERA_HEIGHT) {
                mEngine.runOnUpdateThread(new Runnable() {
                      public void run() {
                            bullet.removeFromScene();
                      }
                });
          }

          super.onManagedUpdate(pSecondsElapsed);
    }

}
```

12. EnemyShip.java

```
package skripsi.akakom.peranghelikopter.game.enemies;
import org.anddev.andengine.engine.Engine;
import org.anddev.andengine.entity.shape.IShape;
import org.anddev.andengine.entity.shape.modifier.IShapeModifier;
import
org.anddev.andengine.entity.shape.modifier.IShapeModifier.IShapeModifierLi
stener;
import org.anddev.andengine.entity.shape.modifier.LoopModifier;
import org.anddev.andengine.entity.shape.modifier.PathModifier;
import
org.anddev.andengine.entity.shape.modifier.PathModifier.IPathModifierListe
ner;
import org.anddev.andengine.entity.shape.modifier.ScaleModifier;
import org.anddev.andengine.entity.shape.modifier.SequenceModifier;
import org.anddev.andengine.entity.shape.modifier.ease.EaseSineInOut;
import org.anddev.andengine.entity.sprite.AnimatedSprite;
import org.anddev.andengine.opengl.texture.region.TiledTextureRegion;
import org.anddev.andengine.util.Path;

import skripsi.akakom.peranghelikopter.game.GameActivity;

public class EnemyShip extends AnimatedSprite implements IEnemy {

    private final Engine mEngine;
    private boolean killed;

    public EnemyShip(float pX, float pY, TiledTextureRegion
pTiledTextureRegion, final Engine engine) {
```

```
        super(pX, pY, pTiledTextureRegion);
        mEngine = engine;
        killed=false;
        GameActivity.enemies.add(this);
        this.animate(GameActivity.ANIMATION_FRAMELENGTH);
  }

  public boolean isKilled(){
        return killed;
  }

  public static EnemyShip reuse() {
        final EnemyShip enemy = GameActivity.enemiesToReuse.get(0);
        enemy.killed = false;
        GameActivity.enemies.add(enemy);
        GameActivity.enemiesToReuse.remove(enemy);
        return enemy;
  }

  public void addToScene() {
        mEngine.getScene().getBottomLayer().addEntity(this);
  }

  public void removeFromScene() {
        killed=true;
        this.clearShapeModifiers();
        this.addShapeModifier(new SequenceModifier(
                    new IShapeModifierListener() {
                          @Override
                          public void onModifierFinished(
                                      IShapeModifier pShapeModifier,
final IShape enemy) {
                                mEngine.runOnUpdateThread(new Runnable()
{
                                      public void run() {

  mEngine.getScene().getBottomLayer().removeEntity(((EnemyShip) enemy));

  GameActivity.enemiesToReuse.add(0,(EnemyShip) enemy);

  GameActivity.enemies.remove((EnemyShip) enemy);
                                                  ((EnemyShip)
enemy).setScale(1);
                                      }
                                });
                          }
                    }, new ScaleModifier(0.3f, 1, 0)));
  }

  public void folow(Path path, float speed) {
        this.addShapeModifier(new LoopModifier(new PathModifier(speed,
path,
                    null, new IPathModifierListener() {
                          @Override
                          public void onWaypointPassed(
                                      final PathModifier pPathModifier,
```

```
                                        final IShape pShape, final int
pWaypointIndex) {
                                    if (pWaypointIndex == 6) {
                                        mEngine.runOnUpdateThread(new
Runnable() {
                                            public void run() {
                                                ((EnemyShip)
pShape).removeFromScene();
                                            }
                                        });
                                    }

                                    if (pWaypointIndex != 0 && pWaypointIndex
!= 6)
                                        ((EnemyShip) pShape).fire();
                                }
                            }, EaseSineInOut.getInstance())));
    }

    public void fire() {
        if(GameActivity.options.getSoundEffects()==true)
            GameActivity.shotSound.play();
        if (GameActivity.enemyBulletsToReuse.isEmpty()) {
            final EnemyBullet metak = new EnemyBullet(this.getX() + 10,
                    this.getY() + 30,
GameActivity.mEnemyBulletTextureRegion,
                    mEngine);
            metak.addToScene();
        } else {
            final EnemyBullet metak = EnemyBullet.reuse(this.getX() +
10,
                    this.getY() + 30);
            metak.addToScene();
        }
    }

}
```

13. IEnemy.java

```
package skripsi.akakom.peranghelikopter.game.enemies;

public interface IEnemy {

    public abstract void addToScene();

    public abstract void removeFromScene();

    public abstract void fire();

}
```

14. Laser.java

```
package skripsi.akakom.peranghelikopter.game.enemies;
import org.anddev.andengine.engine.Engine;
import org.anddev.andengine.entity.shape.modifier.AlphaModifier;
import org.anddev.andengine.entity.shape.modifier.LoopModifier;
import org.anddev.andengine.entity.shape.modifier.SequenceModifier;
import org.anddev.andengine.entity.sprite.Sprite;
import org.anddev.andengine.opengl.texture.region.TextureRegion;

public class Laser extends Sprite {

   private final Engine mEngine;

   public Laser(float pX, float pY,
               TextureRegion pTiledTextureRegion, Engine engine) {
         super(pX, pY, pTiledTextureRegion);
         mEngine = engine;

   }

   public void addToScene() {
         mEngine.getScene().getBottomLayer().addEntity(this);
         addShapeModifier(new LoopModifier(new SequenceModifier(new
AlphaModifier(0.5f, 1, 0.5f), new AlphaModifier(0.5f, 0.5f, 1))));
   }

   public void removeFromScene() {
         mEngine.getScene().getBottomLayer().removeEntity(this);
   }

   @Override
   protected void onManagedUpdate(final float pSecondsElapsed) {

         super.onManagedUpdate(pSecondsElapsed);
   }

}
```

15. Mine.java

```
package skripsi.akakom.peranghelikopter.game.enemies;
import org.anddev.andengine.engine.Engine;
import org.anddev.andengine.entity.shape.IShape;
import org.anddev.andengine.entity.shape.modifier.IShapeModifier;
import
org.anddev.andengine.entity.shape.modifier.IShapeModifier.IShapeModifierLi
stener;
import org.anddev.andengine.entity.shape.modifier.ScaleModifier;
import org.anddev.andengine.entity.shape.modifier.SequenceModifier;
import org.anddev.andengine.entity.sprite.AnimatedSprite;
import org.anddev.andengine.opengl.texture.region.TiledTextureRegion;

public class Mine extends AnimatedSprite implements IEnemy {
```

```
   private final Engine mEngine;
   private boolean killed;

   public Mine(float pX, float pY, TiledTextureRegion pTiledTextureRegion,
final Engine engine) {
         super(pX, pY, pTiledTextureRegion);
         mEngine = engine;
         killed=false;
         this.animate(60);
         this.setVelocityY(100);
   }

   public boolean isKilled(){
         return killed;
   }

   public void addToScene() {
         mEngine.getScene().getBottomLayer().addEntity(this);
   }

   public void removeFromScene() {
         killed=true;
         this.clearShapeModifiers();
         this.addShapeModifier(new SequenceModifier(
                     new IShapeModifierListener() {
                           @Override
                           public void onModifierFinished(
                                       IShapeModifier pShapeModifier,
final IShape enemy) {
                                 mEngine.runOnUpdateThread(new Runnable()
{
                                       public void run() {

   mEngine.getScene().getBottomLayer().removeEntity(((Mine) enemy));
                                       }
                                 });
                           }
                     }, new ScaleModifier(0.3f, 1, 0)));
   }

   @Override
   public void fire() {
         // TODO Auto-generated method stub
   }

}
```

16. HudScore.java

```
package skripsi.akakom.peranghelikopter.game.hud;

public class HudScore {

   private int score;

   public HudScore(){
```

```
		score = 0;
	}

	public void reset(){
		score = 0;
	}

	public int getScore() {
		return score;
	}

	public void addPoints() {
		this.score +=50 ;
	}

}
```

17. Level.java

```
package skripsi.akakom.peranghelikopter.game.level;
import java.util.ArrayList;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.anddev.andengine.entity.scene.background.ColorBackground;
import org.anddev.andengine.ui.activity.BaseGameActivity;
import org.xml.sax.InputSource;
import org.xml.sax.XMLReader;

import skripsi.akakom.peranghelikopter.game.R;

public class Level {

	private BaseGameActivity activity;
	private ArrayList<LevelScene> scenes;
	private int currLevel;

	public Level(BaseGameActivity e){
		activity = e;
		scenes = new ArrayList<LevelScene>();
		currLevel = 0;
		initScenes();
	}

	public void initScenes(){
		for(int i=0;i<4;i++){
			LevelScene s = new LevelScene(3);
			s.setBackground(new ColorBackground(0, 0, 0));
			// LOAD LEVEL
			try {
				SAXParserFactory spf =
SAXParserFactory.newInstance();
				SAXParser sp = spf.newSAXParser();
				XMLReader xr = sp.getXMLReader();

				LevelHandler myXMLHandler = new LevelHandler();
				xr.setContentHandler(myXMLHandler);
```

```
                        switch(i){
                        case 0:
                              xr.parse(new
InputSource(activity.getResources().openRawResource(R.raw.level1)));
                              break;
                        case 1:
                              xr.parse(new
InputSource(activity.getResources().openRawResource(R.raw.level2)));
                              break;
                        case 2:
                              xr.parse(new
InputSource(activity.getResources().openRawResource(R.raw.level3)));
                              break;
                        case 3:
                              xr.parse(new
InputSource(activity.getResources().openRawResource(R.raw.level4)));
                              break;
                        }
                        s.setWaves(LevelHandler.wavesList);
                  } catch (Exception e) {
                        System.out.println("XML Pasing Excpetion = " + e);
                  }
                  scenes.add(s);
            }
      }

      public int getLevel(){
            return currLevel;
      }

      public LevelScene getScene(){
            return scenes.get(currLevel);
      }

      public void next(){
            currLevel++;
            activity.getEngine().setScene(scenes.get(currLevel));
      }
}
```

18. LevelHandler.java

```
package skripsi.akakom.peranghelikopter.game.level;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class LevelHandler extends DefaultHandler {

      Boolean currentElement = false;
      String currentValue = null;
      public static WaveList wavesList = null;
      Wave temp;

      public static WaveList getSitesList() {
            return wavesList;
```

```
    }

    public static void setSitesList(WaveList sitesList) {
          LevelHandler.wavesList = sitesList;
    }

    @Override
    public void startElement(String uri, String localName, String qName,
                Attributes attributes) throws SAXException {

          currentElement = true;

          if (localName.equals("perang_helikopter"))
          {
                wavesList = new WaveList();
          } else if (localName.equals("wave")) {
                temp = new Wave();
          } else if (localName.equals("enemy")) {
                String attr = attributes.getValue("type");
                temp.setType(attr);
          }

    }

    @Override
    public void endElement(String uri, String localName, String qName)
                throws SAXException {

          currentElement = false;

          if (localName.equalsIgnoreCase("enemy"))
                temp.setCount(Integer.parseInt(currentValue));
          if (localName.equalsIgnoreCase("wave"))
                wavesList.addWave(temp);

    }

    @Override
    public void characters(char[] ch, int start, int length)
                throws SAXException {

          if (currentElement) {
                currentValue = new String(ch, start, length);
                currentElement = false;
          }

    }

}
```

19. LevelScene.java

```
package skripsi.akakom.peranghelikopter.game.level;

import org.anddev.andengine.entity.scene.Scene;
```

```
public class LevelScene extends Scene {

   private WaveList waves;

   public LevelScene(int pLayerCount) {
         super(pLayerCount);
         initScene();
   }

   public void setWaves(WaveList w){
         waves=w;
   }

   public WaveList getWaves(){
         return waves;
   }

   public void initScene(){

   }

}
```

20. Wave.java

```
package skripsi.akakom.peranghelikopter.game.level;
import java.util.ArrayList;
public class Wave {

   private ArrayList<String> types = new ArrayList<String>();
   private ArrayList<Integer> counts = new ArrayList<Integer>();

   public int size(){
         return types.size();
   }

   public int getCount(int i) {
         return counts.get(i);
   }

   public void setCount(int c) {
         this.counts.add(c);
   }

   public String getType(int i) {
         return types.get(i);
   }

   public void setType(String t) {
         this.types.add(t);
   }

   public String display(){
         String temp = "";
         for(int i=0;i<types.size();i++){
```

```
                temp += (i+1)+". "+types.get(i)+" "+counts.get(i)+"\n";
          }
          return temp;
    }

}
```

21. WaveList.java

```
package skripsi.akakom.peranghelikopter.game.level;
import java.util.ArrayList;
public class WaveList {

    private ArrayList<Wave> waves = new ArrayList<Wave>();
    private int i=0;

    public ArrayList<Wave> getWaves() {
          return waves;
    }

    public void addWave(Wave w) {
          this.waves.add(w);
    }

    public Wave getCurrentWave(){
          if(i==waves.size())
                return null;
          else{
                return waves.get(i);
          }
    }

    public void next(){
          i++;
    }

}
```

22. Options.java

```
package skripsi.akakom.peranghelikopter.game.options;
import android.app.Activity;
import android.content.SharedPreferences;
import android.preference.PreferenceManager;

public class Options {

    SharedPreferences options;

    public Options(Activity activity){
          options =
PreferenceManager.getDefaultSharedPreferences(activity);
    }

    public boolean getSoundEffects(){
```

```
        return options.getBoolean("chkSound",true);
    }

    public boolean getMusic(){
        return options.getBoolean("chkMusic",true);
    }

    public boolean getVibration(){
        return options.getBoolean("chkVibration",true);
    }

    public boolean getAutoFire(){
        return options.getBoolean("chkAutoFire",true);
    }

    public String getUsername(){
        return options.getString("edUsername",null);
    }

    public String getControlls(){
        return options.getString("lstControls","3");
    }

    public String getDifficulty(){
        return options.getString("lstDifficulty","1");
    }

    public String getResolution(){
        return options.getString("lstResolution","1");
    }

}
```

23. Bullet.java

```
package skripsi.akakom.peranghelikopter.game.player;
import org.anddev.andengine.engine.Engine;
import org.anddev.andengine.entity.sprite.Sprite;
import org.anddev.andengine.opengl.texture.region.TextureRegion;

import skripsi.akakom.peranghelikopter.game.GameActivity;

public class Bullet extends Sprite {

    private final Engine mEngine;

    public Bullet(float pX, float pY, TextureRegion pTiledTextureRegion,
Engine engine) {
        super(pX, pY, pTiledTextureRegion);
        mEngine = engine;
        this.setVelocity(0, -300);
        GameActivity.bullets.add(this);
    }
    public static Bullet reuse(float posx, float posy) {
        final Bullet bullet = GameActivity.bulletsToReuse.get(0);
        GameActivity.bullets.add(bullet);
```

```
            GameActivity.bulletsToReuse.remove(bullet);
            bullet.setVisible(true);
            bullet.setPosition(posx, posy);
            bullet.setVelocity(0, -300);
            return bullet;
      }

      public void addToScene() {
            mEngine.getScene().getBottomLayer().addEntity(this);
      }

      public void removeFromScene() {
            this.setVelocity(0,0);
            this.setVisible(false);
            GameActivity.bulletsToReuse.add(this);
            GameActivity.bullets.remove(this);
      }

      @Override
      protected void onManagedUpdate(final float pSecondsElapsed) {
            final Bullet bullet = this;
            if(bullet.isVisible())
                  if (this.getY() < -this.getHeight()) {
                        mEngine.runOnUpdateThread(new Runnable() {
                              public void run() {
                                          bullet.removeFromScene();
                              }
                        });
                  }

            super.onManagedUpdate(pSecondsElapsed);
      }
   }
```

24. PlayerShip.java

```
   package skripsi.akakom.peranghelikopter.game.player;
   import org.anddev.andengine.engine.Engine;
   import org.anddev.andengine.entity.scene.Scene;
   import org.anddev.andengine.entity.shape.modifier.ColorModifier;
   import org.anddev.andengine.entity.shape.modifier.SequenceModifier;
   import org.anddev.andengine.entity.sprite.AnimatedSprite;
   import org.anddev.andengine.opengl.texture.region.TiledTextureRegion;
   import org.anddev.andengine.sensor.accelerometer.AccelerometerData;

   import skripsi.akakom.peranghelikopter.game.GameActivity;

   public class PlayerShip extends AnimatedSprite {

      private final Engine mEngine;
      //private final PlayerShipShadow shadow;
      private int health = 3;
      private boolean killed=false;
      private int numBullets;

      public boolean isKilled(){
```

```
		return killed;
	}

	public void kill(){
		killed=true;
	}

	public PlayerShip(float pX, float pY,
			TiledTextureRegion pTiledTextureRegion, Engine engine) {
		super(pX, pY, pTiledTextureRegion);
		mEngine = engine;
		this.animate(GameActivity.ANIMATION_FRAMELENGTH);
		numBullets = 3;
		//shadow = new PlayerShipShadow(this);
	}

	//public PlayerShipShadow getShadow(){
		//return shadow;
	//}

	public void addToScene(Scene scene) {
		//scene.getBottomLayer().addEntity(shadow);
		scene.getBottomLayer().addEntity(this);
	}

	public void removeFromScene() {
		//mEngine.getScene().getBottomLayer().removeEntity(shadow);
		mEngine.getScene().getBottomLayer().removeEntity(this);
	}

	public void hit() {
		health--;
		this.addShapeModifier(new SequenceModifier(new
ColorModifier(0.3f, 1, 1, 1, 0, 1, 0), new ColorModifier(0.3f, 1, 1, 0, 1,
0, 1)));
	}

	public int getHealth() {
		return health;
	}

	public void fire() {

		if (GameActivity.bullets.size() < numBullets &&
GameActivity.isGameOver == false) {
			//Debug.d("MECI="+GameActivity.bullets.size()+"
UNISTENO="+GameActivity.bulletsToReuse.size());
			if(GameActivity.options.getSoundEffects()==true)
				GameActivity.shotSound.play();

			if (GameActivity.bulletsToReuse.size() == 0) {
				final Bullet bullet = new Bullet(this.getX()+10,
this.getY(), GameActivity.mBulletTextureRegion, mEngine);
				bullet.addToScene();
				//Debug.d("NEW");
			} else {
				Bullet.reuse(this.getX()+10, this.getY());
```

```
                        //Debug.d("REUSE");
                }
        }
  }

  public void move(AccelerometerData accelerometer) {
        if (GameActivity.isGameOver == false) {
                if (this.getX() > 20 && this.getX() <
mEngine.getCamera().getWidth()-20 - this.getWidth()) {
                        this.setVelocityX(-accelerometer.getX() * 50);
                } else {
                        this.setVelocityX(0);

                        if (this.getX() <= 20 && accelerometer.getX() < 0)
                                this.setVelocityX(-accelerometer.getX() * 50);

                        if (this.getX() >= mEngine.getCamera().getWidth()-20
- this.getWidth()
                                        && accelerometer.getX() > 0)
                                this.setVelocityX(-accelerometer.getX() * 50);
                }

                if (this.getY() > mEngine.getCamera().getHeight()-200 &&
this.getY() < mEngine.getCamera().getHeight()-20 - this.getHeight()) {
                        this.setVelocityY(accelerometer.getY() * 50);
                } else {
                        this.setVelocityY(0);

                        if (this.getY() <= mEngine.getCamera().getHeight()-
200 && accelerometer.getY() > 0)
                                this.setVelocityY(accelerometer.getY() * 50);

                        if (this.getY() >= mEngine.getCamera().getHeight()-20
- this.getHeight()
                                        && accelerometer.getY() < 0)
                                this.setVelocityY(accelerometer.getY() * 50);
                }
        }
  }

}
```

25. PlayerShipShadow.java

```
package skripsi.akakom.peranghelikopter.game.player;

import org.anddev.andengine.entity.sprite.AnimatedSprite;

import skripsi.akakom.peranghelikopter.game.GameActivity;

public class PlayerShipShadow1 extends AnimatedSprite {

  private final PlayerShip s;

  public PlayerShipShadow1(PlayerShip ship) {
        super(ship.getX(), ship.getY(), ship.getTextureRegion());
```

```
        this.animate(GameActivity.ANIMATION_FRAMELENGTH);
        this.setColor(0, 0, 0);
        this.setAlpha(0.4f);
        s=ship;
  }

  @Override
  protected void onManagedUpdate(final float pSecondsElapsed) {

        if(s.getX()+s.getWidth()/2>= GameActivity.CAMERA_WIDTH/2)
              this.setPosition(s.getX()-((s.getX()-
GameActivity.CAMERA_WIDTH/2)/2), s.getY()+10);
        else
              this.setPosition(s.getX()+((GameActivity.CAMERA_WIDTH/2-
s.getX())/2), s.getY()+10);

        super.onManagedUpdate(pSecondsElapsed);
  }

}
```