AntiAircraft - Android tutorial

1 Introduction

The goal of this tutorial is to show how to realize Android application from a very pragmatic point of view.
For this purpose, I will assume that the reader is familiar with Java terminology and programming techniques and knows how to use Eclipse (at least the basics).
The tutorial has been modelled in six incremental steps that will lead the reader from few elements to the whole application.
In each step every created or modified class will be presented so that the reader can compare his own code and don't get lost.
Moreover this tutorial comes with a package containing the code of the six steps and the graphic/sound material described. This will help the reader during development.

Since before this tutorial I didn't know how to develop Android application, I want to thank my friend JuanMa Cuevas Caminero for his precious advices, examples and materials.

The rest of the document is organized as follows:

- **Summary**
  The table of contents of the document itself.
- **Get ready**
  How to download Android SDK, how to set up Android tools and emulators.
- **AntiAircraft**
  A step-by-step incremental tutorial about how to create a non-trivial Android application.
- **Further improvement**
  Some opportunities to deepen Android programming techniques while improving the game.
- **Conclusions**
  A summary of the concepts and techniques we have learnt.
- **Appendix**
  The code of the project after the last step is provided divided into classes for all the needs of the reader.

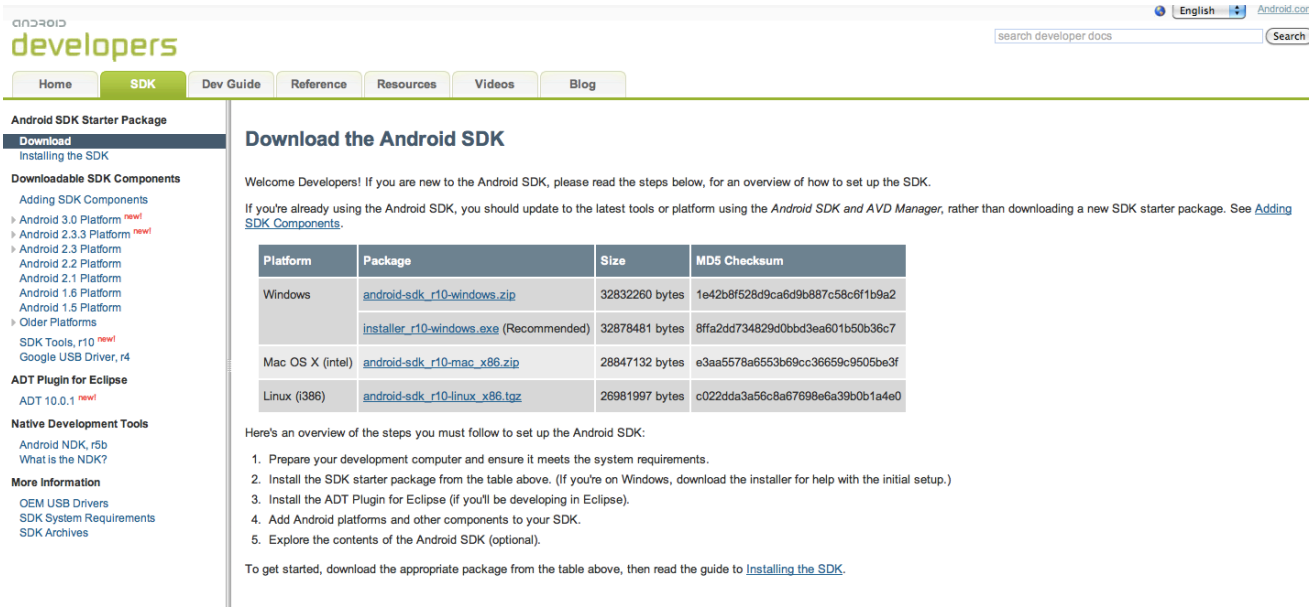
2 Summary

3 Get ready

Android SDK

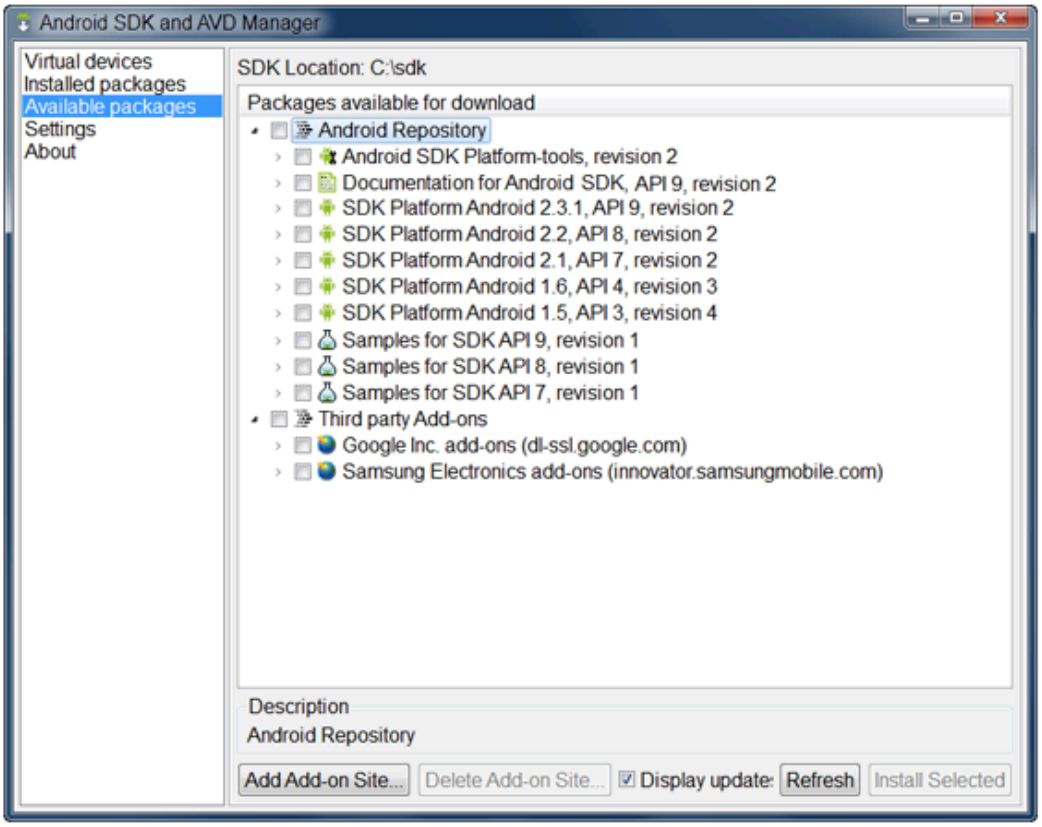
1) Go to http://developer.android.com/

2) Go to SDK tab and download SDK



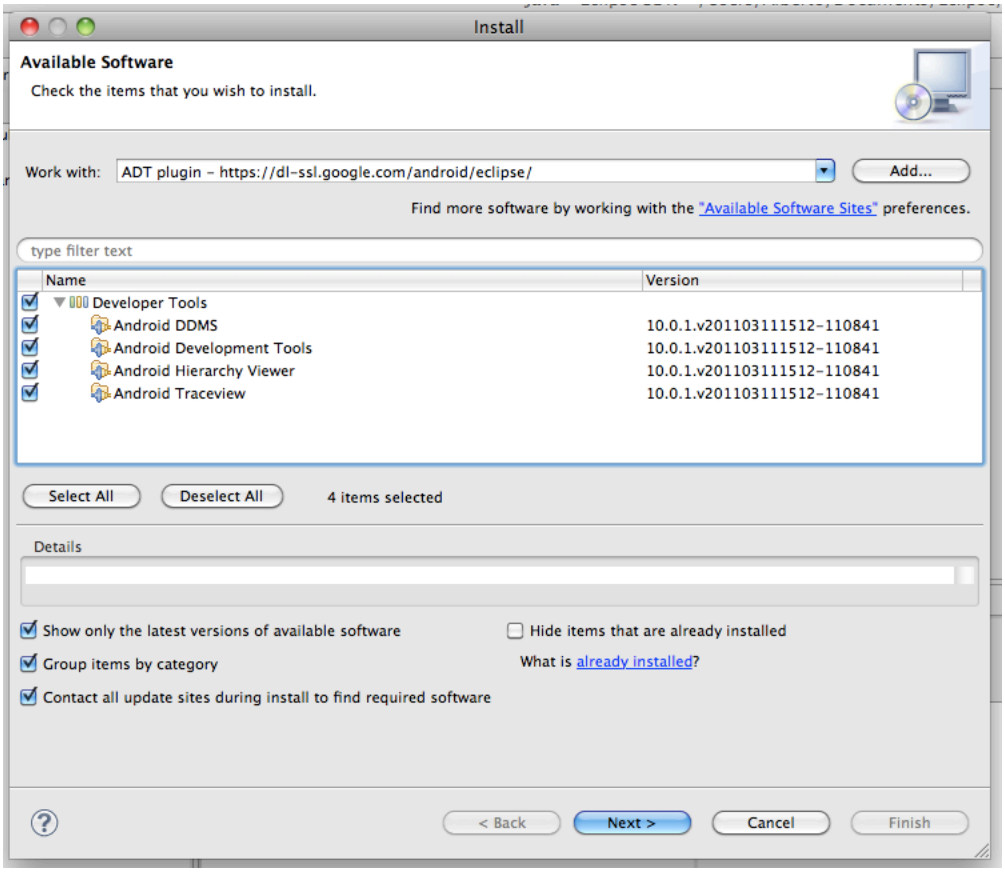
3) Unpack SDK
4) Open the folder called *tools* and execute *android*
5) Download platform
   Usually it's a good practice to download the most updated version but Android **3.0 (Honeycomb)** is a tablet-oriented release so choose Android **2.3 (Gingerbread)**. During 2011 it will probably be released Android 4, codename **Ice Cream Sandwich**, a combination of Gingerbread and Honeycomb, therefore it will be a good idea to work on

that platform.



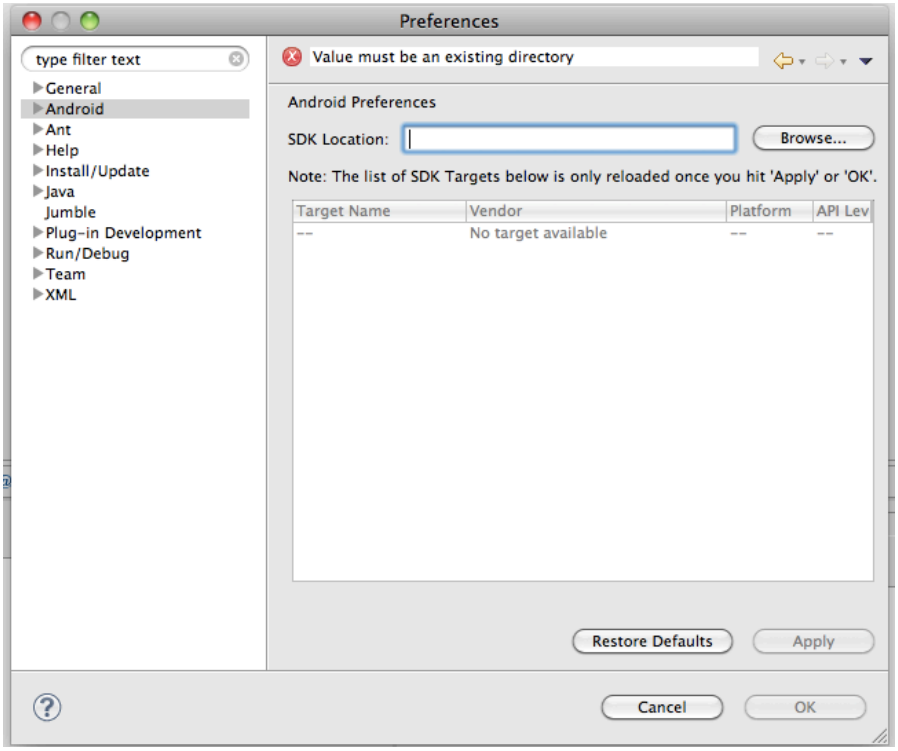
ADT Plugin

1) Start Eclipse, then select **Help** > **Install New Software...**.
2) Click **Add**, in the top-right corner.
3) In the Add Repository dialog that appears, enter "ADT Plugin" for the *Name* and the following URL for the *Location*: https://dl-ssl.google.com/android/eclipse/
   If you have trouble acquiring the plugin, try using "http" in the Location URL, instead of "https" (https is preferred for security reasons).
4) Click **OK**.
5) In the Available Software dialog, select the checkbox next to Developer Tools and click **Next**.
6) In the next window, you'll see a list of the tools to be downloaded. Click **Next**.
7) Read and accept the license agreements, then click **Finish**.
8) When the installation completes, restart Eclipse.

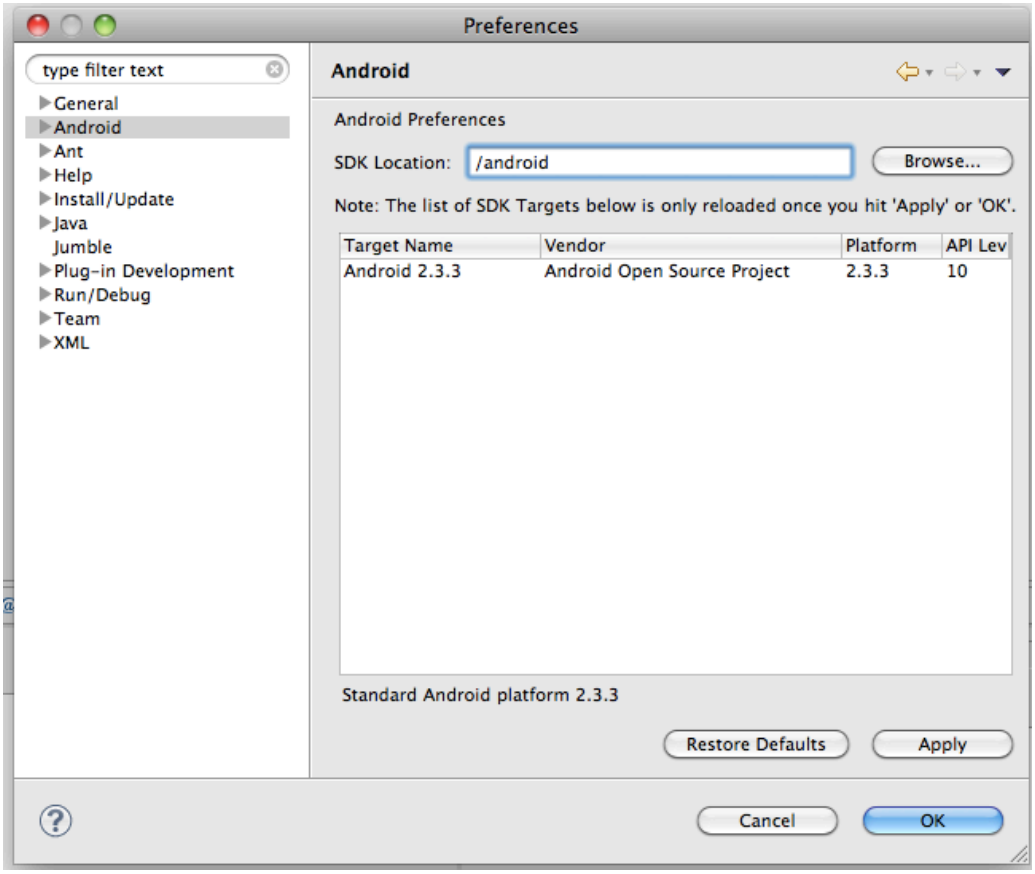


Configure the pluging

1) Open Eclipse preferences.
2) Open Android tab.



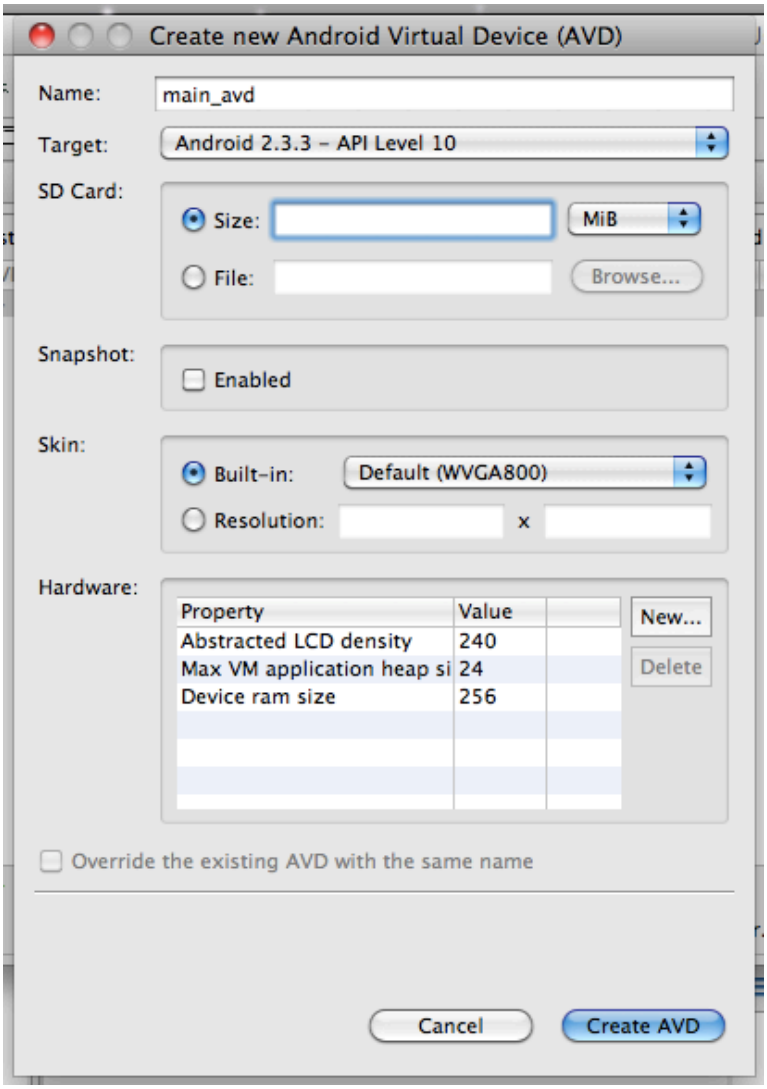
3) Set SDK location by browsing your file system



Create an AVD

In this tutorial, you will run your application in the Android Emulator. Before you can launch the emulator, you must create an Android Virtual Device (AVD). An AVD defines the system image and device settings used by the emulator.
To create an AVD:

1) In Eclipse, choose Window > Android SDK and AVD Manager.
2) Select Virtual Devices in the left panel.
3) Click New.
4) The Create New AVD dialog appears.
5) Type the name of the AVD
6) Choose a target that is the platform you want to run on the emulator.
7) Click Create AVD.


Create new Android Virtual Device (AVD)
Name:
main_avd
Target:
Android 2.3.3 - API Level 10
SD Card:
Size:
MiB
File:
Browse...
Snapshot:
Enabled
Skin:
Built-in:
Default (WVGA800)
Resolution:
x
Hardware:
Property
Value
Abstracted LCD density
240
Max VM application heap si
24
Device ram size
256
New...
Delete
Override the existing AVD with the same name
Cancel
Create AVD

4 AntiAircraft

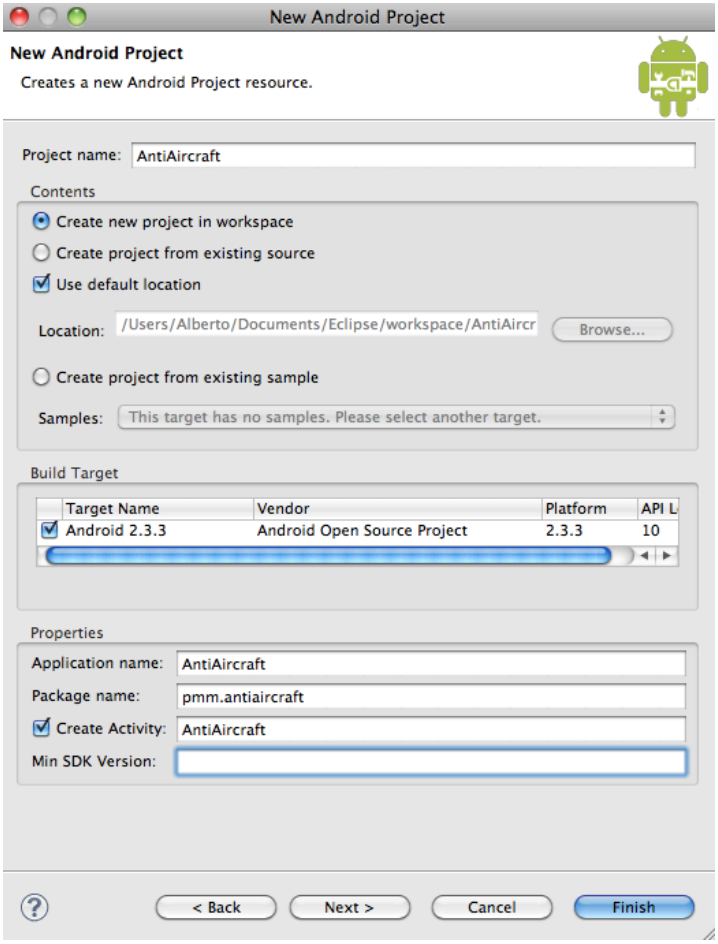
AntiAircraft is an Android application where the user has to fire down alien ships with a tank. The user can rotate and shoot touching the screen. The shot power depends from how the duration of the press and the trajectory follows physics rules.

Step 1 – Game basics

In this first step we want to set up the game view and draw the tank.
We need the following components:

- **GameActivity**
  Manages the actions related to the Activity that is the class that responsible for the window actions (creation, pause, etc.).
  For further details on Activities check
  http://developer.android.com/guide/topics/fundamentals/activities.html.
- **GameView**
  Manages the View that is the basic user interface element.
- **GameThread**
  The class in charge for managing the game flow. Invokes redraw operations.
- **Tank**
  The class responsible for the tank element.

First of all create a new Android project and fill in the required fields.



We can now start implementing the four components.
Let's start from the class will be first launched when application starts: GameActivity.

1) In src folder go inside your package and create a new class called GameActivity.
2) This class has to extend Activity and implement the method onStart() that is invoked when the activity becomes visible to the user.
3) We will define the View in a xml file called *main*. For the moment let's assume it exists and use it.
4) This class has to register an handle to the thread GameThread

```
package pmm.antiaircraft1;

import android.app.Activity;

public class GameActivity extends Activity{

	private GameThread mGameThread;

	public void onStart() {
		super.onStart();
		setContentView(R.layout.main);
//		mGameThread = something
	}
}
```

5) We want also our application to be shown in full screen so let's add these two lines:

```
requestWindowFeature(Window.FEATURE_NO_TITLE);
getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
```

For the moment this is all we can do so let's pass to define another class: GameView.

1) In src folder go inside your package and create a new class called GameView.
2) This class will extend SurfaceView and implement SurfaceHolder.callback so that we can draw on it from our thread.
3) The GameView constructor has to register the holder and add a callback to it (for the same reason explained above) and create a thread.
4) When creating the thread we will pass the holder and the context.
5) We have register the thread and to create a method to return it (to the main activity that will use it in case of pause of restart for example).
6) Three methods have to be implemented as required by SurfaceHolder.Callback:
    a. surfaceCreated
       starts the thread
    b. surfaceChanged
       we won't use this
    c. surfaceDestroyed
       we won't use this

Finally, the GameView class should look like this.
It contains some inexistent methods, related to GameThread, that we will create in few minutes.

```
public class GameView extends SurfaceView implements SurfaceHolder.Callback {
	private GameThread thread;
```

```
	public GameView(Context context, AttributeSet attrs) {
		super(context, attrs);
		SurfaceHolder holder = getHolder();
		holder.addCallback(this);
		thread = new GameThread(holder, context,null);
	}

	public GameThread getThread() {
		return thread;
	}

	public void surfaceCreated(SurfaceHolder holder) {
		thread.setRunning(true);
		thread.start();
	}

	@Override
	public void surfaceChanged(SurfaceHolder holder, int format, int width,
			int height) {}

	@Override
	public void surfaceDestroyed(SurfaceHolder holder) {}
}
```

Before proceeding with the GameThread we have to complete the GameActivity where we have to implement the commented part: adding reference to the thread.
To do this we first need to define the view *main*.

1) Create a new XML file inside res/layout and call it main.
2) Define a frame layout.
3) Add the GameView with a "fill_parent" layout_width and layout_height to make it as big as possible.
4) Add a id reference to the GameView so that we can retrieve it univocally.

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent" android:layout_height="fill_parent">
    <pmm.antiaircraft1.GameView
      android:id="@+id/game"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"/>
</FrameLayout>
```

Now we can get the thread using the getThread method in the GameView.

1) In GameActivity substitute the commented code with the following one.

```
mGameThread = ((GameView) findViewById(R.id.game)).getThread();
```

2) Now as a last operation of onStart() let's create the graphics invoking createGraphics(), a method we will define in GameThread soon.

The final code for GameActivity should be like this:

```
package pmm.antiaircraft1;
```

```
import android.app.Activity;
import android.view.Window;
import android.view.WindowManager;

public class GameActivity extends Activity{

	private GameThread mGameThread;

	public void onStart() {
		super.onStart();
		requestWindowFeature(Window.FEATURE_NO_TITLE);
		getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
		setContentView(R.layout.main);
		mGameThread = ((GameView) findViewById(R.id.game)).getThread();
		mGameThread.createGraphics();
	}
}
```

Although the code has some errors for callings to inexistent methods, we have a basic idea of the preliminary operations: a (Game)View with FrameLayout is created, it creates and register the game thread and when it is ready it starts the thread itself. The view is found by GameActivity from its ID and from the method getThread() of the view the GameThread is registered. GameActivity then invokes the creation of Graphics to the GameThread.
We have now to define the code for GameThread.

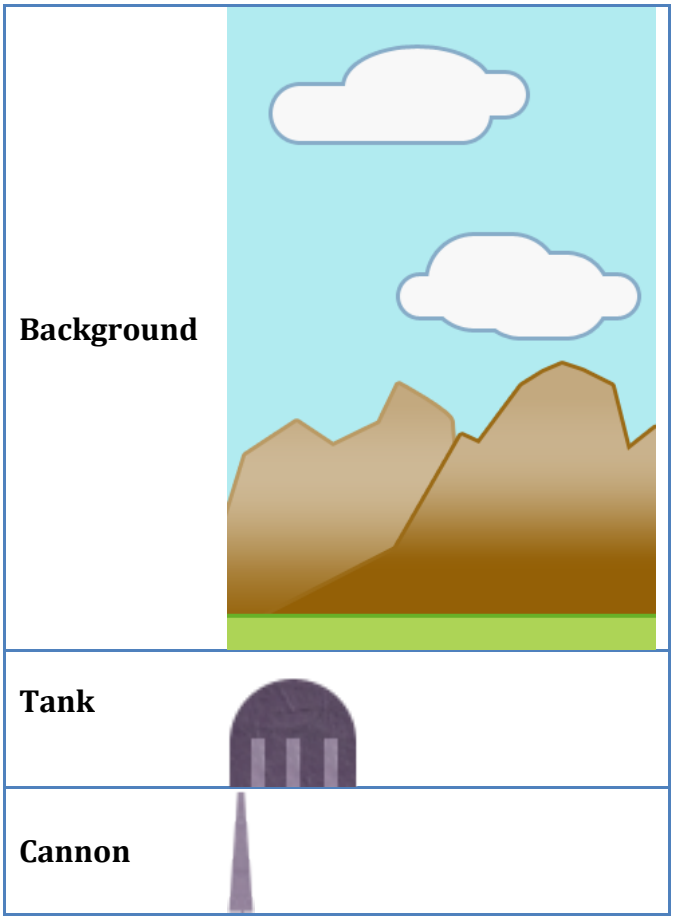
1) In src folder go inside your package and create a new class called GameThread.
2) GameThread is a thread which means it has to extend the class Thread
3) Besides constructor it contains 4 methods:
    a. Run
       responsible for the execution of the game and the repainting operations
    b. setRunning
       sets a flag that indicates if the thread is running or not
    c. draw
       draws the elements on the canvas
    d. createGraphics
       creates the elements of the game

The code for this class is provided below. Take a look, we discuss some parts after the code itself.

```
package pmm.antiaircraft1;

import android.content.Context;
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.os.Handler;
import android.util.DisplayMetrics;
import android.view.SurfaceHolder;
```

```
public class GameThread extends Thread {
	private SurfaceHolder mSurfaceHolder;
	public DisplayMetrics metrics = new DisplayMetrics();
	private static Bitmap mBackgroundImage;
	public static Bitmap tankImg;
	public static Bitmap cannonImg;
	public static boolean mRun = false;
	private Tank tank;
	public static Resources res;

	public GameThread(SurfaceHolder surfaceHolder, Context context, Handler handler) {
		mSurfaceHolder = surfaceHolder;
		Resources res = context.getResources();
		mBackgroundImage = BitmapFactory.decodeResource(res,R.drawable.background);
		tankImg = BitmapFactory.decodeResource(res,R.drawable.tank);
		cannonImg = BitmapFactory.decodeResource(res,R.drawable.cannon);
	}

	@Override
	public void run() {
		while (mRun) {
			Canvas c = null;
			try {
				c = mSurfaceHolder.lockCanvas(null);
				synchronized (mSurfaceHolder) {
					draw(c);
				}
			} finally {
				if (c != null) {
					mSurfaceHolder.unlockCanvasAndPost(c);
				}
			}
		}
	}

	public void setRunning(boolean b) {
		mRun = b;
	}

	private void draw(Canvas canvas) {
		canvas.drawBitmap(mBackgroundImage, 0, 0, null);
		tank.draw(canvas);
	}

	public void createGraphics() {
		mBackgroundImage=Bitmap.createBitmap(mBackgroundImage);
		tank = new Tank();
	}
}
```

In the constructor we getResources (rsc folder) and then invoke decodeResource. The parameters of this method are the graphic resources we will use for our application.
For the moment we will use the images summarized in the following table.



I created this png images (png is the recommended encoding for Android images) using Photoshop. The reader can also use other applications or download similar assets from the web. The cannon is separated from the tank because we want to rotate it.
Since the application will be published for devices with 320x480 monitor resolution, then the background has these dimensions. Usually it is a good practice to adapt the image to the monitor but for this tutorial we won't consider these kind of techniques.
It is important to notice that the canvas is obtained from the surfaceHolder.
Finally notice the steps required to draw an element:

1) Get the context
2) Get the resources
3) Decode the image
4) Create the bitmap
5) Draw the bitmap on the canvas

For the background we did all the previous operations but for the tank we delegate this duty to the Tank class.
Let's define it.

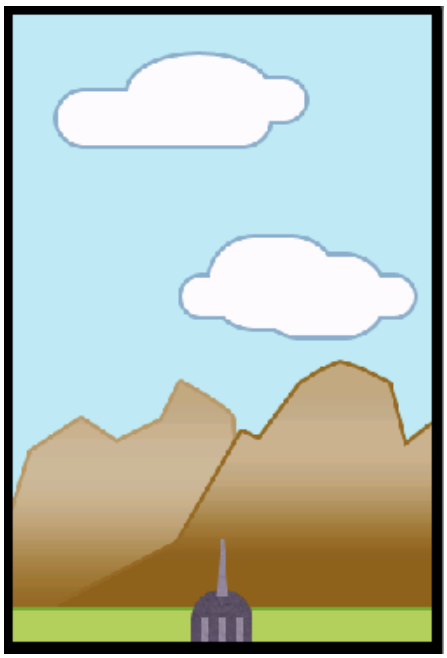
1) In src folder go inside your package and create a new class called Tank.
2) This class has some dimensions of used images set as constant because they will be used in position calculus.
3) The constructor creates bitmap and register those in GameThread
4) The method *draw* (yes the one we invoked in GameThread) is in charge of drawing operations

5) The draw method uses Matrix for place the cannon because it will be very useful in the next code iteration.

```
package pmm.antiaircraft1;

import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Matrix;

public class Tank {
	private static final int TANK_HEIGHT = 50;
	private static final int TANK_WIDTH = 50;
	private static final int TANK_TOP = 480 - TANK_HEIGHT;
	private static final int CANNON_WIDTH = 10;

	public Tank(){
		GameThread.tankImg=Bitmap.createBitmap(GameThread.tankImg);
		GameThread.cannonImg=Bitmap.createBitmap(GameThread.cannonImg);
	}

	public void draw(Canvas c) {
		c.drawBitmap(GameThread.tankImg, (320-TANK_WIDTH)/2, TANK_TOP, null);
		Matrix m = new Matrix();
		m.postTranslate((320-CANNON_WIDTH)/2, TANK_TOP - 30);
		c.drawBitmap(GameThread.cannonImg, m, null);
	}
}
```

Note the calculus will place the tank at the bottom-center of our 320x480 screen.

Finally, if we Run our application as Android Application we will see something like this.



This concludes the first step: we have our running application that… doesn't do anything!

Step 2 – Cannon movements

In this step we'll realize the cannon movements so we need a class to manage user interaction: InputObject.

1) In src folder go inside your package (for this step the package has been renamed to pmm.antiaircraft2) and create a new class called InputObject.
2) Create a method useEvent for MotionEvent that has to register the type of action, the time it happened and its coordinates.
3) The constructor registers an ArrayBlockingQueue of InputObject. We'll see later when it is called.
4) A method adds the InputObject to the ArrayBlockingQueue of InputObject.

This class is not difficult but requires an understanding of the Android touch events. The reader can get information at http://developer.android.com/guide/topics/ui/ui-events.html.
The final class should look like this.

```
package pmm.antiaircraft2;
import java.util.concurrent.ArrayBlockingQueue;
import android.view.MotionEvent;

public class InputObject {
	public static final byte EVENT_TYPE_KEY = 1;
	public static final byte EVENT_TYPE_TOUCH = 2;
	public static final int ACTION_KEY_DOWN = 1;
	public static final int ACTION_KEY_UP = 2;
	public static final int ACTION_TOUCH_DOWN = 3;
	public static final int ACTION_TOUCH_MOVE = 4;
	public static final int ACTION_TOUCH_UP = 5;
	public ArrayBlockingQueue<InputObject> pool;
	public byte eventType;
	public long time;
	public int action;
	public int keyCode;
	public int x;
	public int y;

	public InputObject(ArrayBlockingQueue<InputObject> pool) {
		this.pool = pool;
	}

	public void useEvent(MotionEvent event) {
		eventType = EVENT_TYPE_TOUCH;
		int a = event.getAction();
		switch (a) {
		case MotionEvent.ACTION_DOWN:
			action = ACTION_TOUCH_DOWN;
			break;
		case MotionEvent.ACTION_MOVE:
			action = ACTION_TOUCH_MOVE;
			break;
		case MotionEvent.ACTION_UP:
			action = ACTION_TOUCH_UP;
			break;
		default:
			action = 0;
```

```
            }
            time = event.getEventTime();
            x = (int) event.getX() ;
            y = (int) event.getY();
      }

      public void useEventHistory(MotionEvent event, int historyItem) {
            eventType = EVENT_TYPE_TOUCH;
            action = ACTION_TOUCH_MOVE;
            time = event.getHistoricalEventTime(historyItem);
            x = (int) event.getHistoricalX(historyItem);
            y = (int) event.getHistoricalY(historyItem);
      }

      public void returnToPool() {
            pool.add(this);
      }
}
```

Now let's edit our GameActivity class:

1) Add a method for creating the InputObject queue.
2) Call this method as the last onStart action.
3) Override onTouchEvent method to manage event queue and related action.

The 2 method are the following.

```
      private void createInputObjectPool() {
            inputObjectPool = new ArrayBlockingQueue<InputObject>(INPUT_QUEUE_SIZE);
            for (int i = 0; i < INPUT_QUEUE_SIZE; i++) {
                  inputObjectPool.add(new InputObject(inputObjectPool));
            }
      }

      @Override
      public boolean onTouchEvent(MotionEvent event) {
            try {
                  int hist = event.getHistorySize();
                  for (int i = 0; i < hist; i++) {
                        InputObject input = inputObjectPool.take();
                        input.useEventHistory(event, i);
                        mGameThread.feedInput(input);
                  }
                  InputObject input = inputObjectPool.take();
                  input.useEvent(event);
                  mGameThread.feedInput(input);
            } catch (InterruptedException e) {
            }
            try {
                  Thread.sleep(16);
            } catch (InterruptedException e) {
            }
            return true;
      }
```

The class should now look like this:

```
package pmm.antiaircraft2;

import java.util.concurrent.ArrayBlockingQueue;

import android.app.Activity;
import android.view.MotionEvent;
import android.view.Window;
import android.view.WindowManager;

public class GameActivity extends Activity{

	private GameThread mGameThread;
	public static final int INPUT_QUEUE_SIZE = 30;
	public ArrayBlockingQueue<InputObject> inputObjectPool;

	public void onStart() {
		super.onStart();
		requestWindowFeature(Window.FEATURE_NO_TITLE);
	  getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
		setContentView(R.layout.main);
		mGameThread = ((GameView) findViewById(R.id.game)).getThread();
		mGameThread.createGraphics();
		createInputObjectPool();
	}

	private void createInputObjectPool() {
		inputObjectPool = new ArrayBlockingQueue<InputObject>(INPUT_QUEUE_SIZE);
		for (int i = 0; i < INPUT_QUEUE_SIZE; i++) {
			inputObjectPool.add(new InputObject(inputObjectPool));
		}
	}

	@Override
	public boolean onTouchEvent(MotionEvent event) {
		try {
			int hist = event.getHistorySize();
			for (int i = 0; i < hist; i++) {
				InputObject input = inputObjectPool.take();
				input.useEventHistory(event, i);
				mGameThread.feedInput(input);
			}
			InputObject input = inputObjectPool.take();
			input.useEvent(event);
			mGameThread.feedInput(input);
		} catch (InterruptedException e) {
		}
		try {
			Thread.sleep(16);
		} catch (InterruptedException e) {
		}
		return true;
	}
}
```

As we have seen some modification must be performed on GameThread:

1) New method to feedInput.
2) New method to processInput.
3) New method to processMotionEvent, for the moment it takes care only of cannon movement but it contains 3 different behaviours because this will be necessary for firing operation.

The methods are defined as follows.

```
	public void feedInput(InputObject input) {
		synchronized(inputQueueMutex) {
			try {
				inputQueue.put(input);
			} catch (InterruptedException e) {
			}
		}
	}

	private void processInput() {
		synchronized(inputQueueMutex) {
			ArrayBlockingQueue<InputObject> inputQueue = this.inputQueue;
			while (!inputQueue.isEmpty()) {
				try {
					InputObject input = inputQueue.take();
					if (input.eventType == InputObject.EVENT_TYPE_TOUCH) {
						processMotionEvent(input);
					}
					input.returnToPool();
				} catch (InterruptedException e) {
				}
			}
		}
	}

	private void processMotionEvent(InputObject input) {
		if( input.action==InputObject.ACTION_TOUCH_DOWN){

			tank.setTarget(input.x, input.y);

		}	else if( input.action==InputObject.ACTION_TOUCH_MOVE) {
			tank.setTarget(input.x, input.y);
		}

		else if( input.action==InputObject.ACTION_TOUCH_UP){
			tank.setTarget(input.x, input.y);
		}
	}
```

4) In run() invoke processInput just before the draw invocation.

Now the class should look like follows:

```
package pmm.antiaircraft2;

import java.util.concurrent.ArrayBlockingQueue;
```

```
import android.content.Context;
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.os.Handler;
import android.util.DisplayMetrics;
import android.view.SurfaceHolder;

public class GameThread extends Thread {
	private SurfaceHolder mSurfaceHolder;
	public DisplayMetrics metrics = new DisplayMetrics();
	private static Bitmap mBackgroundImage;
	public static Bitmap tankImg;
	public static Bitmap cannonImg;
	public static boolean mRun = false;
	private Tank tank;
	public static Resources res;
	private Object inputQueueMutex = new Object();
	private ArrayBlockingQueue<InputObject> inputQueue = new
ArrayBlockingQueue<InputObject>(30);

	public GameThread(SurfaceHolder surfaceHolder, Context context, Handler handler) {
		mSurfaceHolder = surfaceHolder;
		Resources res = context.getResources();
		mBackgroundImage = BitmapFactory.decodeResource(res,R.drawable.background);
		tankImg = BitmapFactory.decodeResource(res,R.drawable.tank);
		cannonImg = BitmapFactory.decodeResource(res,R.drawable.cannon);
	}

	@Override
	public void run() {
		while (mRun) {
			Canvas c = null;
			try {
				c = mSurfaceHolder.lockCanvas(null);
				synchronized (mSurfaceHolder) {
					processInput();
					draw(c);
				}
			} finally {
				if (c != null) {
					mSurfaceHolder.unlockCanvasAndPost(c);
				}
			}
		}
	}

	public void setRunning(boolean b) {
		mRun = b;
	}

	private void draw(Canvas canvas) {
		canvas.drawBitmap(mBackgroundImage, 0, 0, null);
		tank.draw(canvas);
	}
```

```
	public void createGraphics() {
		mBackgroundImage=Bitmap.createBitmap(mBackgroundImage);
		tank = new Tank();
	}

	public void feedInput(InputObject input) {
		synchronized(inputQueueMutex) {
			try {
				inputQueue.put(input);
			} catch (InterruptedException e) {
			}
		}
	}

	private void processInput() {
		synchronized(inputQueueMutex) {
			ArrayBlockingQueue<InputObject> inputQueue = this.inputQueue;
			while (!inputQueue.isEmpty()) {
				try {
					InputObject input = inputQueue.take();
					if (input.eventType == InputObject.EVENT_TYPE_TOUCH) {
						processMotionEvent(input);
					}
					input.returnToPool();
				} catch (InterruptedException e) {
				}
			}
		}
	}

	private void processMotionEvent(InputObject input) {
		if( input.action==InputObject.ACTION_TOUCH_DOWN){
			tank.setTarget(input.x, input.y);

		}	else if( input.action==InputObject.ACTION_TOUCH_MOVE) {
			tank.setTarget(input.x, input.y);
		}

		else if( input.action==InputObject.ACTION_TOUCH_UP){
			tank.setTarget(input.x, input.y);
		}
	}
}
```

Last but not less important we have to define the motion of the cannon itself in Tank class.

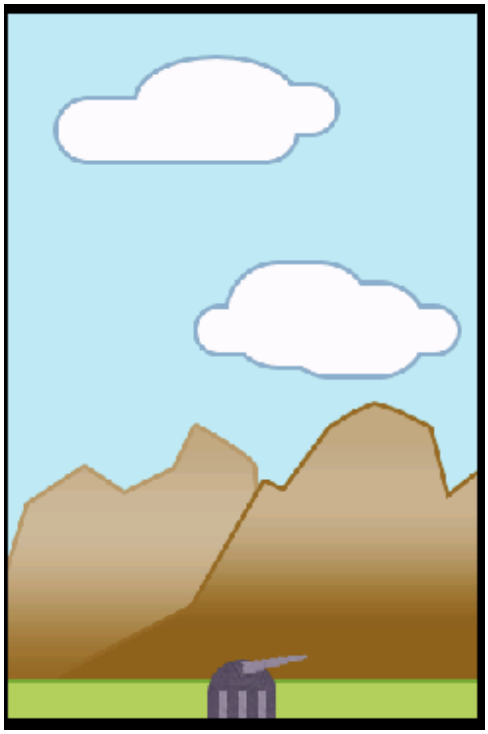
1) Create setTarget method. Use arctang to calculate the angle but pay attention to division by 0 and similar cases.
2) Rotate the cannon using this value before drawing it.

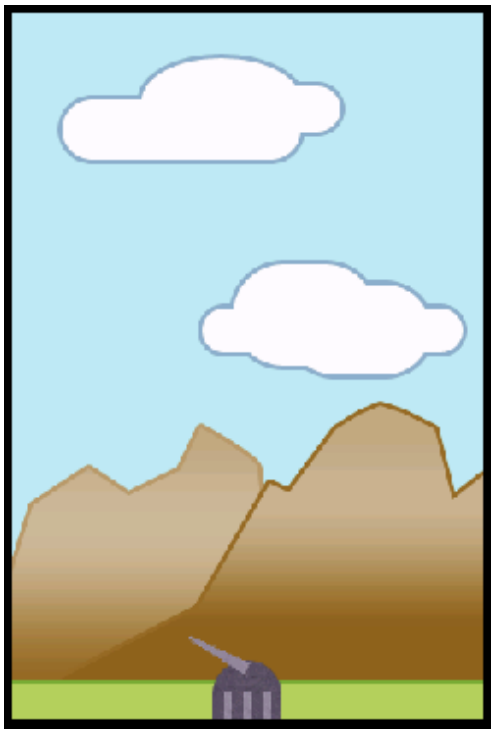
It's not difficult but some attention is required in order to avoid mathematical errors. The use of debug may come in handy.
This is the final version of the class:

```
package pmm.antiaircraft2;
```

```
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Matrix;

public class Tank {
	private static final int TANK_HEIGHT = 50;
	private static final int TANK_WIDTH = 50;
	private static final int TANK_TOP = 480 - TANK_HEIGHT;
	private static final int CANNON_WIDTH = 10;
	private float angle;

	public Tank(){
		GameThread.tankImg=Bitmap.createBitmap(GameThread.tankImg);
		GameThread.cannonImg=Bitmap.createBitmap(GameThread.cannonImg);
	}

	public void draw(Canvas c) {
		c.drawBitmap(GameThread.tankImg, (320-TANK_WIDTH)/2, TANK_TOP, null);
		Matrix m = new Matrix();
		m.postTranslate((320-CANNON_WIDTH)/2, TANK_TOP - 30);
		m.postRotate((float) ((-angle)*180 /Math.PI),160,446);
		c.drawBitmap(GameThread.cannonImg, m, null);
	}

	public void setTarget(int x, int y) {
		if (x==160)
			angle=0;
		else{
			if(y>=446){
				angle=(float)Math.PI/2;
				if(x>160)
					angle-=2*angle;
			}
			else
				angle = (float) Math.atan((float)(x-160)/(float)(y-446));
		}
	}
}
```

We can now run the application and interact with it moving the cannon.





Step 3 – Shoot!

Now we can move the cannon but the tank is not ready yet: it's time to shoot!
As usual, we want to create a new class to model the bullet:

1) In src folder go inside your package (for this step the package has been renamed to pmm.antiaircraft3) and create a new class called Bullet.
2) This class defines some constants to identify the status of the bullet and its size.
3) This class has the variables needed to identify the position of the bullet, its velocity and the direction.
4) There is a method to update all this variables in relation of time passed.
5) We want to model also the explosion of the bullet after X milliseconds.
6) There is a method draw that takes care of drawing a circle representing the bullet itself.
7) We can calculate the new position with the following formula

```
posX=(float) (posX0 + iniSpeedX*t2);
posY=(float) (posY0 - (iniSpeedY*t2 - (9.8 *Math.pow(t2, 2)/2)));
```

The final Bullet class should be similar to the one below.

```
package pmm.antiaircraft3;

import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;

public class Bullet {
	private static final int STATUS_FLYING = 0;
	private static final int STATUS_BOOM = 1;
	private static final int STATUS_OVER = 2;
```

```
	private static final int BULLET_RADIUS  = 4;

	private static final long TIME_EXPLODING  = 100;

	private long time;

	private int status;
	private int power;
	private float posX0;
	private float posY0;
	private float posX;
	private float posY;
	private float iniSpeedX;
	private float iniSpeedY;

	private Paint paint;

	private long explodingTimer;

	public Bullet(int power,float angle,int x0,int y0){
		time=0;
		paint = new Paint();
		this.power = power;
		posX0 = x0;
		posY0 = y0;
		iniSpeedY=(float) (Math.sin(angle)*power*1.5) ;
		iniSpeedX=(float) (Math.cos(angle)*power*1.5);
	}

	public void draw(Canvas c) {
		paint.setColor(Color.RED);
		if (status == STATUS_FLYING)
			paint.setColor(Color.BLACK);
		c.drawCircle(posX, posY, BULLET_RADIUS, paint);
	}

	public void update(long elapsedTime) {
		switch (status){
		case STATUS_FLYING:
			time+=elapsedTime;
			double t2=(double)(time)/100;
			posX=(float) (posX0 + iniSpeedX*t2);
			posY=(float) (posY0 - (iniSpeedY*t2 - (9.8 *Math.pow(t2, 2)/2)));

			if (posY>=480) {
				status=STATUS_BOOM;
				posY = 480;
				explodingTimer = 0;
			}
			break;
		case STATUS_BOOM:
			explodingTimer+=elapsedTime;
			if (explodingTimer>TIME_EXPLODING)
				status=STATUS_OVER;
			break;
		}
	}
```

```
	public void setImpact(){
		status = STATUS_BOOM;
		explodingTimer=0;
	}

	public float getPosX() {
		return posX;
	}

	public float getPosY() {
		return posY;
	}

	public boolean isFlying(){
		return (status == STATUS_FLYING);
	}

	public boolean isExploding() {
		return (status==STATUS_BOOM);
	}

	public boolean isOver() {
		return (status==STATUS_OVER);
	}

}
```

Now we have to update the GameThread in order to insert the callings to fire methods and physics (that will be in Tank)

1) Update processMotionEvent as follows

```
private void processMotionEvent(InputObject input) {
	if( input.action==InputObject.ACTION_TOUCH_DOWN){
		tank.pressFire();
		tank.setTarget(input.x, input.y);
	}
	if( input.action==InputObject.ACTION_TOUCH_UP){
		tank.setTarget(input.x, input.y);
		tank.releaseFire();
	}
	if( input.action==InputObject.ACTION_TOUCH_MOVE)
		tank.setTarget(input.x, input.y);
}
```

2) The class constructor should now create a list of bullets.
3) The method run has to calculate the time passed and invoke updatePhysics method after processed the input.
4) The updatePhysics method calls methods to update the status of our objects if they are related to time as the bullet and the tank. Why also the tank? Because as we will see it will have methods to calculate the initial velocity of the bullet based on the duration of the finger tap on device screen.
5) The draw method has to be updated in order to draw all the bullets in the list.

The final code for this class is this:

```
package pmm.antiaircraft3;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.concurrent.ArrayBlockingQueue;

import android.content.Context;
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.os.Handler;
import android.util.DisplayMetrics;
import android.view.SurfaceHolder;

public class GameThread extends Thread {
	private SurfaceHolder mSurfaceHolder;
	public DisplayMetrics metrics = new DisplayMetrics();
	private static Bitmap mBackgroundImage;
	public static Bitmap tankImg;
	public static Bitmap cannonImg;
	public static boolean mRun = false;
	private Tank tank;
	public static Resources res;
	private Object inputQueueMutex = new Object();
	private ArrayBlockingQueue<InputObject> inputQueue = new
ArrayBlockingQueue<InputObject>(30);
	private static ArrayList<Bullet> bullets;
	private long lastUpdateTime;

	public GameThread(SurfaceHolder surfaceHolder, Context context, Handler handler) {
		mSurfaceHolder = surfaceHolder;
		Resources res = context.getResources();
		mBackgroundImage = BitmapFactory.decodeResource(res,R.drawable.background);
		tankImg = BitmapFactory.decodeResource(res,R.drawable.tank);
		cannonImg = BitmapFactory.decodeResource(res,R.drawable.cannon);
		bullets = new ArrayList<Bullet>();
	}

	@Override
	public void run() {
		while (mRun) {
			Canvas c = null;
			try {
				c = mSurfaceHolder.lockCanvas(null);
				synchronized (mSurfaceHolder) {
					long currentTime = System.currentTimeMillis();
					long delta = (long) (currentTime - lastUpdateTime);
					lastUpdateTime = currentTime;
					processInput();
					updatePhysics(delta);
					draw(c);
				}
			} finally {
				if (c != null) {
```

```
                        mSurfaceHolder.unlockCanvasAndPost(c);
                    }
                }
            }
        }

        public void setRunning(boolean b) {
            mRun = b;
        }

        private void draw(Canvas canvas) {
            canvas.drawBitmap(mBackgroundImage, 0, 0, null);
            for(Iterator<Bullet> it = bullets.iterator();it.hasNext();){
                Bullet b = (Bullet) it.next();
                if (b!=null)
                    b.draw(canvas);
            }
            tank.draw(canvas);
        }

        public void createGraphics() {
            mBackgroundImage=Bitmap.createBitmap(mBackgroundImage);
            tank = new Tank();
        }

        private void updatePhysics(long timer) {
            timer=(long) (timer);
            tank.update(timer);
            Bullet b ;
            int i=0;
            while (!bullets.isEmpty() && bullets.size()>i){
                b = bullets.get(i);
                if (b.isOver())
                    bullets.remove(i);
                else {
                    b.update(timer);
                    i++;
                }
            }
        }

        public void feedInput(InputObject input) {
            synchronized(inputQueueMutex) {
                try {
                    inputQueue.put(input);
                } catch (InterruptedException e) {
                }
            }
        }

        private void processInput() {
            synchronized(inputQueueMutex) {
                ArrayBlockingQueue<InputObject> inputQueue = this.inputQueue;
                while (!inputQueue.isEmpty()) {
                    try {
                        InputObject input = inputQueue.take();
                        if (input.eventType == InputObject.EVENT_TYPE_TOUCH) {
```

```
                                        processMotionEvent(input);
                                }
                                input.returnToPool();
                        } catch (InterruptedException e) {
                        }
                }
        }
    }

    private void processMotionEvent(InputObject input) {
        if( input.action==InputObject.ACTION_TOUCH_DOWN){
            tank.pressFire();
            tank.setTarget(input.x, input.y);
        }
        if( input.action==InputObject.ACTION_TOUCH_UP){
            tank.setTarget(input.x, input.y);
            tank.releaseFire();
        }
        if( input.action==InputObject.ACTION_TOUCH_MOVE)
            tank.setTarget(input.x, input.y);
    }

    public static void shootBullet(float angle, int power,int x, int y) {
        bullets.add(new Bullet(power, angle, x, y));
    }
}
```

Note we draw the bullet before the tank so the bullet is behind it.
Finally we have to create the firing methods in Tank.

1) Create methods pressFire and releaseFire used to calculate the power between the two events.
2) Create update method to calculate the power at any call. Set a maximum boundary to avoid too fast shots.
3) Add variables to track the coordinates of the cannon end so that we can make the bullet start from there. These variables have to be updated every time the cannon moves.

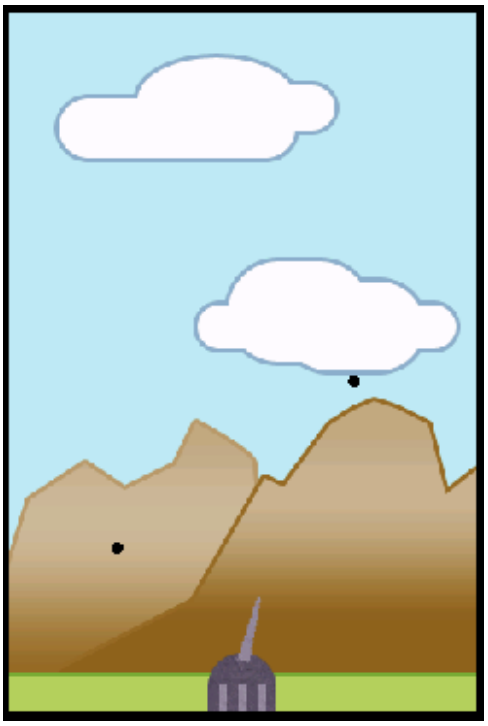
The Tank class should now look like this.

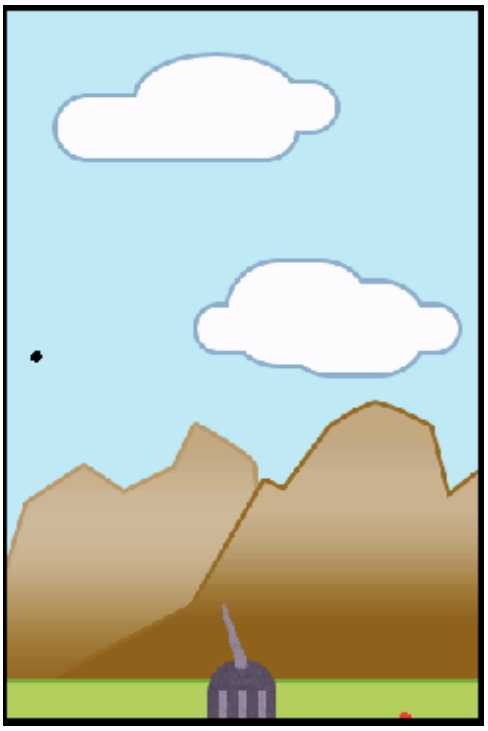
```
package pmm.antiaircraft3;

import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Matrix;

public class Tank {
    private static final int TANK_HEIGHT = 50;
    private static final int TANK_WIDTH = 50;
    private static final int TANK_TOP = 480 - TANK_HEIGHT;
    private static final int CANNON_WIDTH = 10;
    private static final int GUNBARREL_LENGTH = 46;
    private static final int STATUS_IDLE=0;
    private static final int STATUS_POWERING=1;
    private static final long POWERING_TIMER_LIMIT = 1200;
    private float angle;
```

```
	public static int cannonEndX;
	public static int cannonEndY;
	private int status;
	private long poweringTimer;
	private int power;
	private int lastBulletPower;

	public Tank(){
		GameThread.tankImg=Bitmap.createBitmap(GameThread.tankImg);
		GameThread.cannonImg=Bitmap.createBitmap(GameThread.cannonImg);
	}

	public void draw(Canvas c) {
		c.drawBitmap(GameThread.tankImg, (320-TANK_WIDTH)/2, TANK_TOP, null);
		Matrix m = new Matrix();
		m.postTranslate((320-CANNON_WIDTH)/2, TANK_TOP - 30);
		m.postRotate((float) ((-angle)*180 /Math.PI),160,446);
		c.drawBitmap(GameThread.cannonImg, m, null);
		cannonEndX = (int) (160 + (Math.cos(angle) * GUNBARREL_LENGTH));
		cannonEndY = (int) (446 - (Math.sin(angle) * GUNBARREL_LENGTH));
	}

	public void setTarget(int x, int y) {
		if (x==160)
			angle=0;
		else{
			if(y>=446){
				angle=(float)Math.PI/2;
				if(x>160)
					angle-=2*angle;
			}
			else
				angle = (float) Math.atan((float)(x-160)/(float)(y-446));
		}
	}

	public void update(long elapsedTime){
		if (status == STATUS_POWERING){
			poweringTimer=poweringTimer+elapsedTime;
			if (poweringTimer>POWERING_TIMER_LIMIT)
				poweringTimer=POWERING_TIMER_LIMIT;
			power = (int) (((float)poweringTimer / POWERING_TIMER_LIMIT) *100);
		}
	}

	public void pressFire( ) {
		status = STATUS_POWERING;
		power = 0;
		poweringTimer = 0;
	}

	public void releaseFire() {
		GameThread.shootBullet(angle+(float)Math.PI/2,40+power*60/100,160,446);
		lastBulletPower = power;
		status = STATUS_IDLE;
		power = 0;
	}
```

```
}
```

At the end of this step we have a tank with moving cannon that shoots big black bullets according to physics rules!





Step 4 – Aliens!

The tank is ready to fight! Now we need enemies. Directly from science fiction: UFO!

Fist of all we need to draw our flying saucers using Photoshop or another graphic application. I did the following image in 5 minutes; the reader can use it or make a better one.

As always we have to create a specific class, let's call it Ufo.

1) In src folder go inside your package (for this step the package has been renamed to pmm.antiaircraft4) and create a new class called Ufo.
2) The constructor has to define the initial position of the ufo and its speed, and these values have to be randomly generated.
3) The method update updates all the values related to the ufo movement, if its status is "flying", however for the moment this is the only possible status.
4) There has to be a method to draw the ufo.

Below there is the Ufo class.

```
package pmm.antiaircraft4;

import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Matrix;

public class Ufo {
	private static final int STATUS_FLYING = 0;
	private static final int STATUS_OVER = 1;
	private static final float TIME_FLYING = 5;

	private float lowerPosition;
	private long time;
	private int status;
	private float posX;
	private float posY;
	private double iniSpeedX;
	private double iniSpeedY;
	private float posX0;
	private float drawX;
	private float drawY;
	private int angle;
	private int direction;
	private static float acceleration;

	public Ufo(){
		time = 0;
		angle = 0;
		status = STATUS_FLYING;
		direction = (Math.random()<0.5?1:-1);
		posX0 = (float) (160 + (320*-direction*Math.random()));

		lowerPosition = 480;

		acceleration = (float) (lowerPosition/Math.pow(TIME_FLYING/2,2));

		iniSpeedX = (320 / TIME_FLYING);
		iniSpeedY = (TIME_FLYING * acceleration / 2) - 1 / TIME_FLYING;

		GameThread.ufoImg = Bitmap.createBitmap(GameThread.ufoImg);
	}

	public void draw(Canvas c) {
		Matrix m = new Matrix();
		m.postTranslate(drawX, drawY);

		if (status == STATUS_FLYING)
			c.drawBitmap(GameThread.ufoImg, m, null);
	}

	public void update(long elapsedTime) {
		double t2;
		switch (status){
			case STATUS_FLYING:
				time+=elapsedTime;
				t2=(double)(time)/1000;
				posX = (float) (posX0 +  iniSpeedX*t2*direction);
```

```
                        posY = (float) (iniSpeedY * t2 - (acceleration/2 *
Math.pow(t2,2) ));
                        angle = (direction>0?180:0) + (int) (Math.atan((iniSpeedY-
acceleration*t2)/iniSpeedX*direction)*180/Math.PI);
                        if (posY<0)
                              status=STATUS_OVER;
                  break;
            }

            drawX = posX - 30;
            drawY = posY - 20;
      }

      public boolean isOver() {
            return (status==STATUS_OVER);
      }

      public boolean isFlying(){
            return (status == STATUS_FLYING);
      }
}
```

We now have to update the GameThread in a way it is now able to generate Ufo instances and invoke the related methods.

1) There is a new image to decode in the constructor.
2) Create a new list of Ufo.
3) Add the code for drawing the Ufo in the draw method.
4) In updatePhysics add a section where a new Ufo instance is created every X seconds and invoke the ufo update method for each ufo in the list.

Now the GameThread should look like this:

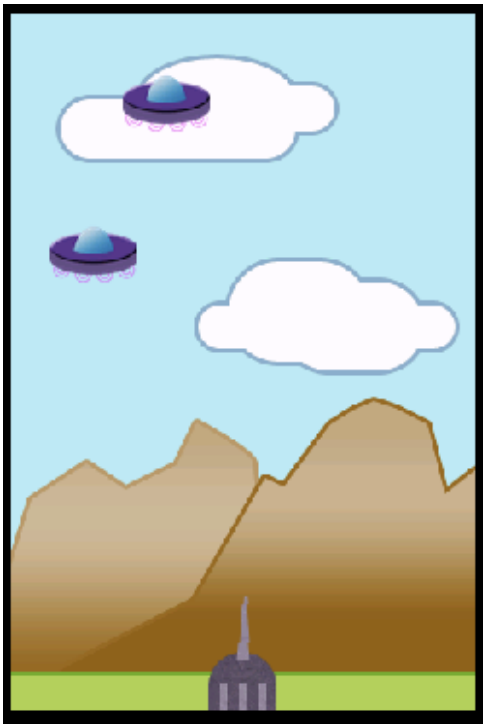
```
package pmm.antiaircraft4;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.concurrent.ArrayBlockingQueue;

import android.content.Context;
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.os.Handler;
import android.util.DisplayMetrics;
import android.view.SurfaceHolder;

public class GameThread extends Thread {
      private SurfaceHolder mSurfaceHolder;
      public DisplayMetrics metrics = new DisplayMetrics();
      private static Bitmap mBackgroundImage;
      public static Bitmap tankImg;
      public static Bitmap cannonImg;
      public static Bitmap ufoImg;
      public static boolean mRun = false;
```

```
	private Tank tank;
	public static Resources res;
	private Object inputQueueMutex = new Object();
	private ArrayBlockingQueue<InputObject> inputQueue = new
ArrayBlockingQueue<InputObject>(30);
	private static ArrayList<Bullet> bullets;
	private static ArrayList<Ufo> ufos;
	private long lastUpdateTime;
	private long newUfoTimer;

	public GameThread(SurfaceHolder surfaceHolder, Context context, Handler handler) {
		mSurfaceHolder = surfaceHolder;
		Resources res = context.getResources();
		mBackgroundImage = BitmapFactory.decodeResource(res,R.drawable.background);
		tankImg = BitmapFactory.decodeResource(res,R.drawable.tank);
		cannonImg = BitmapFactory.decodeResource(res,R.drawable.cannon);
		ufoImg= BitmapFactory.decodeResource(res,R.drawable.ufo);
		bullets = new ArrayList<Bullet>();
		ufos = new ArrayList<Ufo>();
	}

	@Override
	public void run() {
		while (mRun) {
			Canvas c = null;
			try {
				c = mSurfaceHolder.lockCanvas(null);
				synchronized (mSurfaceHolder) {
					long currentTime = System.currentTimeMillis();
					long delta = (long) (currentTime - lastUpdateTime);
					lastUpdateTime = currentTime;
					processInput();
					updatePhysics(delta);
					draw(c);
				}
			} finally {
				if (c != null) {
					mSurfaceHolder.unlockCanvasAndPost(c);
				}
			}
		}
	}

	public void setRunning(boolean b) {
		mRun = b;
	}

	private void draw(Canvas canvas) {
		canvas.drawBitmap(mBackgroundImage, 0, 0, null);
		for(Iterator<Bullet> it = bullets.iterator();it.hasNext();){
			Bullet b = (Bullet) it.next();
			if (b!=null)
				b.draw(canvas);
		}

		for(Iterator<Ufo> it = ufos.iterator();it.hasNext();){
			Ufo a = (Ufo) it.next();
```

```
                if (a!=null)
                        a.draw(canvas);
        }

        tank.draw(canvas);
    }

    public void createGraphics() {
        mBackgroundImage=Bitmap.createBitmap(mBackgroundImage);
        tank = new Tank();
    }

    private void updatePhysics(long timer) {
        timer=(long) (timer);
        tank.update(timer);
        Bullet b ;
        int i=0;
        while (!bullets.isEmpty() && bullets.size()>i){
                b = bullets.get(i);
                if (b.isOver())
                        bullets.remove(i);
                else {
                        b.update(timer);
                        i++;
                }
        }

        Ufo a ;
        i=0;
        while (!ufos.isEmpty() && ufos.size()>i){
                a = ufos.get(i);
                a.update(timer);
                i++;
        }
        newUfoTimer+=timer;
        if (newUfoTimer>3000){
                ufos.add(new Ufo());
                newUfoTimer=0;
        }
    }

    public void feedInput(InputObject input) {
        synchronized(inputQueueMutex) {
                try {
                        inputQueue.put(input);
                } catch (InterruptedException e) {
                }
        }
    }

    private void processInput() {
        synchronized(inputQueueMutex) {
                ArrayBlockingQueue<InputObject> inputQueue = this.inputQueue;
                while (!inputQueue.isEmpty()) {
                        try {
                                InputObject input = inputQueue.take();
                                if (input.eventType == InputObject.EVENT_TYPE_TOUCH) {
```

```
                                        processMotionEvent(input);
                                }
                                input.returnToPool();
                        } catch (InterruptedException e) {
                        }
                }
        }
    }

    private void processMotionEvent(InputObject input) {
        if( input.action==InputObject.ACTION_TOUCH_DOWN){
            tank.pressFire();
            tank.setTarget(input.x, input.y);
        }
        if( input.action==InputObject.ACTION_TOUCH_UP){
            tank.setTarget(input.x, input.y);
            tank.releaseFire();
        }
        if( input.action==InputObject.ACTION_TOUCH_MOVE)
            tank.setTarget(input.x, input.y);
    }

    public static void shootBullet(float angle, int power,int x, int y) {
        bullets.add(new Bullet(power, angle, x, y));
    }
}
```

At the end of this iteration we have created our enemies with a specific movements but, for the moment, they are invulnerable to our bullets.



Step 5 – Hit!

In this step we will manage the collision between bullets and enemies.
We want also to keep the count of victims and other indicators, as shooting power.
First of all there is the need of a new class for displaying the crucial information on the screen.
We will call it head-up display, in short HUD.

1) In src folder go inside your package (for this step the package has been renamed to pmm.antiaircraft5) and create a new class called Hud.
2) This class contains the position on the screen of the text and the other visual elements we want to present.
3) The most important method is draw that contains the code for the correct display of information.
4) The information are: angle, power and number of impacts (enemies destroyed). For this purpose we need a method that registers the handle of the tank.
5) There is a method addImpact that raises the number of impacts.

The code of this class is therefore similar to the following one.

```
package pmm.antiaircraft5;

import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.LinearGradient;
import android.graphics.Paint;
import android.graphics.Paint.Style;
import android.graphics.RectF;
import android.graphics.Shader;

public class Hud {
	private static final int BAR_POWER_LEFT_MARGIN = 10;
	private static final int BAR_POWER_RIGHT_MARGIN = 200;
	private static final int BAR_POWER_BOTTOM_MARGIN = 10;
	private static final int BAR_POWER_HEIGHT = 10;
	private static final int TEXT_INFO_LEFT_MARGIN = 10;
	private static final int TEXT_ANGLE_TOP_MARGIN = 25;
	private static final int TEXT_POWER_TOP_MARGIN = 50;
	private static final int TEXT_COUNTER_TOP_MARGIN = 75;
	private static final int TEXT_SIZE = 20;
	private float barPwrLeft;
	private float barPwrTop;
	private float barPwrRight;
	private float barPwrBottom;
	private int impactCounter;
	LinearGradient gradient;
	Paint paint;
	private Tank tank;

	public Hud(){
		paint = new Paint();

		barPwrLeft = BAR_POWER_LEFT_MARGIN;
		barPwrTop = 480 - BAR_POWER_BOTTOM_MARGIN - BAR_POWER_HEIGHT;
		barPwrRight = 320 - BAR_POWER_RIGHT_MARGIN;
		barPwrBottom = 480 - BAR_POWER_HEIGHT;
		impactCounter =0;
```

```
            gradient= new LinearGradient (barPwrLeft, barPwrTop, barPwrRight, barPwrTop,
new int[]{Color.GREEN,Color.YELLOW, Color.RED},null, Shader.TileMode.CLAMP);
      }

      public void draw(Canvas c){
            int progress = ((int) (barPwrLeft + tank.getPower() * (barPwrRight-
barPwrLeft) /100));
            paint.setAlpha(255);
            paint.setShader(gradient);
            paint.setStyle(Style.FILL);
            c.drawRoundRect(new RectF(barPwrLeft, barPwrTop, progress, barPwrBottom), 4,
4, paint);
            paint.setShader(null);
            paint.setStyle(Style.STROKE);
            paint.setColor(Color.BLACK);
            paint.setStrokeWidth(0);
            c.drawRoundRect(new RectF(barPwrLeft, barPwrTop, barPwrRight, barPwrBottom),
4, 4, paint);

            paint.setColor(Color.BLACK);
            paint.setStyle(Paint.Style.STROKE);
            paint.setStrokeWidth(0);
            paint.setAntiAlias(true);
            paint.setTextSize(TEXT_SIZE);
            int
powerInfo=(tank.getPower()==0?tank.getLastBulletPower():tank.getPower());
            c.drawText("Angle: "+Math.abs((int) (tank.getAngle()*180/Math.PI))+"°",
TEXT_INFO_LEFT_MARGIN, TEXT_ANGLE_TOP_MARGIN, paint);
            c.drawText("Power: "+powerInfo, TEXT_INFO_LEFT_MARGIN, TEXT_POWER_TOP_MARGIN,
paint);
            c.drawText("Impacts: "+impactCounter, TEXT_INFO_LEFT_MARGIN,
TEXT_COUNTER_TOP_MARGIN, paint);
      }

      public void register(Tank tank) {
            this.tank = tank;
      }

      public void addImpact() {
            impactCounter++;
      }
}
```

Note how we drawn a power charging bar.
Now we have to update the GameThread class to call Hud methods.

1) First of all draw has to invoke the draw of hud.
2) In createGraphics a new Hud object is created and the tank is registered.
3) In updatePhysics we have to add a check for collisions between bullets and ufos.
4) If a collision has been detected then the addImpact method of the Hud and the setImpact methods of ufo and bullet are called.

The updated class has the following code.

```
package pmm.antiaircraft5;
```

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.concurrent.ArrayBlockingQueue;

import android.content.Context;
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.os.Handler;
import android.util.DisplayMetrics;
import android.view.SurfaceHolder;

public class GameThread extends Thread {
	private SurfaceHolder mSurfaceHolder;
	public DisplayMetrics metrics = new DisplayMetrics();
	private static Bitmap mBackgroundImage;
	public static Bitmap tankImg;
	public static Bitmap cannonImg;
	public static Bitmap ufoImg;
	public static boolean mRun = false;
	private Tank tank;
	public static Resources res;
	private Object inputQueueMutex = new Object();
	private ArrayBlockingQueue<InputObject> inputQueue = new
ArrayBlockingQueue<InputObject>(30);
	private static ArrayList<Bullet> bullets;
	private static ArrayList<Ufo> ufos;
	private long lastUpdateTime;
	private long newUfoTimer;
	private Hud hud;

	public GameThread(SurfaceHolder surfaceHolder, Context context, Handler handler) {
		mSurfaceHolder = surfaceHolder;
		Resources res = context.getResources();
		mBackgroundImage = BitmapFactory.decodeResource(res,R.drawable.background);
		tankImg = BitmapFactory.decodeResource(res,R.drawable.tank);
		cannonImg = BitmapFactory.decodeResource(res,R.drawable.cannon);
		ufoImg= BitmapFactory.decodeResource(res,R.drawable.ufo);
		bullets = new ArrayList<Bullet>();
		ufos = new ArrayList<Ufo>();
	}

	@Override
	public void run() {
		while (mRun) {
			Canvas c = null;
			try {
				c = mSurfaceHolder.lockCanvas(null);
				synchronized (mSurfaceHolder) {
					long currentTime = System.currentTimeMillis();
					long delta = (long) (currentTime - lastUpdateTime);
					lastUpdateTime = currentTime;
					processInput();
					updatePhysics(delta);
					draw(c);
```

```
                        }
                } finally {
                        if (c != null) {
                                mSurfaceHolder.unlockCanvasAndPost(c);
                        }
                }
        }
}

public void setRunning(boolean b) {
        mRun = b;
}

private void draw(Canvas canvas) {
        canvas.drawBitmap(mBackgroundImage, 0, 0, null);
        for(Iterator<Bullet> it = bullets.iterator();it.hasNext();){
                Bullet b = (Bullet) it.next();
                if (b!=null)
                        b.draw(canvas);
        }

        for(Iterator<Ufo> it = ufos.iterator();it.hasNext();){
                Ufo a = (Ufo) it.next();
                if (a!=null)
                        a.draw(canvas);
        }

        tank.draw(canvas);
        hud.draw(canvas);
}

public void createGraphics() {
        mBackgroundImage=Bitmap.createBitmap(mBackgroundImage);
        tank = new Tank();
        hud = new Hud();
        hud.register(tank);
}

private void updatePhysics(long timer) {
        timer=(long) (timer);
        tank.update(timer);
        Bullet b ;
        int i=0;
        while (!bullets.isEmpty() && bullets.size()>i){
                b = bullets.get(i);
                if (b.isOver())
                        bullets.remove(i);
                else {
                        b.update(timer);
                        i++;
                }
        }

        Ufo a ;
        i=0;
        while (!ufos.isEmpty() && ufos.size()>i){
                a = ufos.get(i);
```

```
                if (a.isOver())
                        ufos.remove(i);
                else {
                        a.update(timer);
                        int j=0;
                        while (j<bullets.size()){
                                b = bullets.get(j);
                                if (a.isFlying() && b.isFlying())
                                        if(a.impactDetected(b)){
                                                hud.addImpact();
                                                a.setImpact();
                                                b.setImpact();
                                                ufos.add(new Ufo());
                                        }
                                j++;
                        }
                        i++;
                }
        }
        newUfoTimer+=timer;
        if (newUfoTimer>3000){
                ufos.add(new Ufo());
                newUfoTimer=0;
        }
}

public void feedInput(InputObject input) {
        synchronized(inputQueueMutex) {
                try {
                        inputQueue.put(input);
                } catch (InterruptedException e) {
                }
        }
}

private void processInput() {
        synchronized(inputQueueMutex) {
                ArrayBlockingQueue<InputObject> inputQueue = this.inputQueue;
                while (!inputQueue.isEmpty()) {
                        try {
                                InputObject input = inputQueue.take();
                                if (input.eventType == InputObject.EVENT_TYPE_TOUCH) {
                                        processMotionEvent(input);
                                }
                                input.returnToPool();
                        } catch (InterruptedException e) {
                        }
                }
        }
}

private void processMotionEvent(InputObject input) {
        if( input.action==InputObject.ACTION_TOUCH_DOWN){
                tank.pressFire();
                tank.setTarget(input.x, input.y);
        }
        if( input.action==InputObject.ACTION_TOUCH_UP){
```

```
                        tank.setTarget(input.x, input.y);
                        tank.releaseFire();
                }
                if( input.action==InputObject.ACTION_TOUCH_MOVE)
                        tank.setTarget(input.x, input.y);
        }

        public static void shootBullet(float angle, int power,int x, int y) {
                bullets.add(new Bullet(power, angle, x, y));
        }
}
```

So now we have to add the described methods to Ufo class.

1) Add a method to detect impacts.
2) Add a method to change the status of the bullet.

The code of the class is below here.

```
package pmm.antiaircraft5;

import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Matrix;

public class Ufo {
        private static final int STATUS_FLYING = 0;
        private static final int STATUS_OVER = 1;
        private static final float TIME_FLYING = 5;

        private float lowerPosition;
        private long time;
        private int status;
        private float posX;
        private float posY;
        private double iniSpeedX;
        private double iniSpeedY;
        private float posX0;
        private float drawX;
        private float drawY;
        private int angle;
        private int direction;
        private static float acceleration;

        public Ufo(){
                time = 0;
                angle = 0;
                status = STATUS_FLYING;
                direction = (Math.random()<0.5?1:-1);
                posX0 = (float) (160 + (320*-direction*Math.random()));

                lowerPosition = 480;

                acceleration = (float) (lowerPosition/Math.pow(TIME_FLYING/2,2));

                iniSpeedX = (320 / TIME_FLYING);
```

```
		iniSpeedY = (TIME_FLYING * acceleration / 2) - 1 / TIME_FLYING;

		GameThread.ufoImg = Bitmap.createBitmap(GameThread.ufoImg);
	}

	public void draw(Canvas c) {
		Matrix m = new Matrix();
		m.postTranslate(drawX, drawY);

		if (status == STATUS_FLYING)
			c.drawBitmap(GameThread.ufoImg, m, null);
	}

	public void update(long elapsedTime) {
		double t2;
		switch (status){
			case STATUS_FLYING:
				time+=elapsedTime;
				t2=(double)(time)/1000;
				posX = (float) (posX0 +  iniSpeedX*t2*direction);
				posY = (float) (iniSpeedY * t2 - (acceleration/2 *
Math.pow(t2,2) ));
				angle = (direction>0?180:0) + (int) (Math.atan((iniSpeedY-
acceleration*t2)/iniSpeedX*direction)*180/Math.PI);
				if (posY<0)
					status=STATUS_OVER;
			break;
		}

		drawX = posX - 30;
		drawY = posY - 20;
	}

	public boolean isOver() {
		return (status==STATUS_OVER);
	}

	public boolean isFlying(){
		return (status == STATUS_FLYING);
	}

	public boolean impactDetected(Bullet b) {
		boolean impact = false;
		float diffX=Math.abs(posX-b.getPosX());
		float diffY=Math.abs(posY-b.getPosY());
		if (diffX<19 && diffY<19){
			impact = true;
		}
		return impact;
	}

	public void setImpact(){
		status = STATUS_OVER;
	}
}
```

Finally, in Tank class we need to add the getters so that the hud can retrieve up to date information.

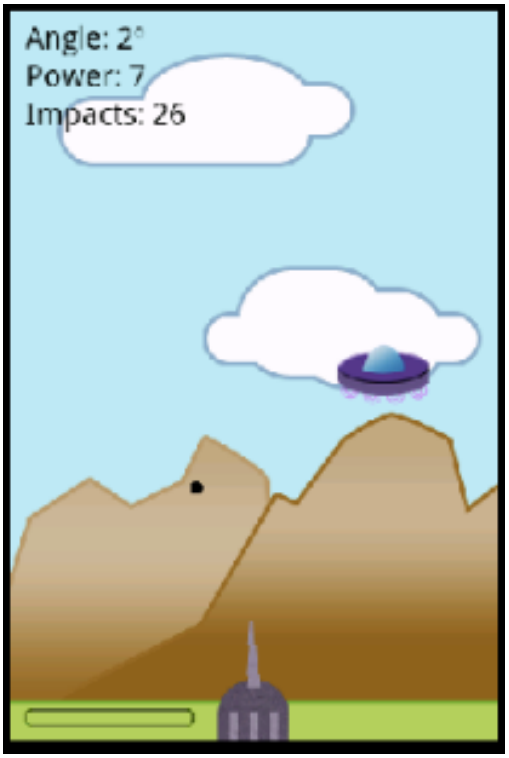
1) Add getAngle method.
2) Add getPower method.
3) Add getLastBulletPower method.

At this point the Tank class should look like this.

```
package pmm.antiaircraft5;

import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Matrix;

public class Tank {
	private static final int TANK_HEIGHT = 50;
	private static final int TANK_WIDTH = 50;
	private static final int TANK_TOP = 480 - TANK_HEIGHT;
	private static final int CANNON_WIDTH = 10;
	private static final int GUNBARREL_LENGTH = 46;
	private static final int STATUS_IDLE=0;
	private static final int STATUS_POWERING=1;
	private static final long POWERING_TIMER_LIMIT = 1200;
	private float angle;
	public static int cannonEndX;
	public static int cannonEndY;
	private int status;
	private long poweringTimer;
	private int power;
	private int lastBulletPower;

	public Tank(){
		GameThread.tankImg=Bitmap.createBitmap(GameThread.tankImg);
		GameThread.cannonImg=Bitmap.createBitmap(GameThread.cannonImg);
	}

	public void draw(Canvas c) {
		c.drawBitmap(GameThread.tankImg, (320-TANK_WIDTH)/2, TANK_TOP, null);
		Matrix m = new Matrix();
		m.postTranslate((320-CANNON_WIDTH)/2, TANK_TOP - 30);
		m.postRotate((float) ((-angle)*180 /Math.PI),160,446);
		c.drawBitmap(GameThread.cannonImg, m, null);
		cannonEndX = (int) (160 + (Math.cos(angle) * GUNBARREL_LENGTH));
		cannonEndY = (int) (446 - (Math.sin(angle) * GUNBARREL_LENGTH));
	}

	public void setTarget(int x, int y) {
		if (x==160)
			angle=0;
		else{
			if(y>=446){
				angle=(float)Math.PI/2;
				if(x>160)
					angle-=2*angle;
			}
			else
```

```
                        angle = (float) Math.atan((float)(x-160)/(float)(y-446));
                }
        }

        public void update(long elapsedTime){
                if (status == STATUS_POWERING){
                        poweringTimer=poweringTimer+elapsedTime;
                        if (poweringTimer>POWERING_TIMER_LIMIT)
                                poweringTimer=POWERING_TIMER_LIMIT;
                        power = (int) (((float)poweringTimer / POWERING_TIMER_LIMIT) *100);
                }
        }

        public void pressFire( ) {
                status = STATUS_POWERING;
                power = 0;
                poweringTimer = 0;
        }

        public void releaseFire() {
                GameThread.shootBullet(angle+(float)Math.PI/2,40+power*60/100,160,446);
                lastBulletPower = power;
                status = STATUS_IDLE;
                power = 0;
        }

        public int getPower() {
                return power;
        }

        public float getAngle() {
                return angle;
        }

        public int getLastBulletPower() {
                return lastBulletPower;
        }
}
```

At this point our game is complete: we have a Tank that shoots Bullets against Ufo(s) and all the information is shown by a Hud.
But we want to make our game more intriguing so let's proceed with the last step.



Step 6 – Sounds

Let's add some sounds to improve quality and fun!
An obvious pre-requirement for this step is to actually have the sound for:

- the main theme of gameplay;
- the shoot;
- the motion of the cannon;
- the explosion of the enemy.

Once we found/created this sounds, we can design a new class for managing the sounds.

1) In src folder go inside your package (for this step the package has been renamed to pmm.antiaircraft6) and create a new class called SoundManager.
2) In the constructor the sounds are loaded.
3) There are specific methods to play the different sounds.
4) There is a method called update to restart the main theme and manage the sound of the cannon motion.

The code of the class is provided here.

```
package pmm.antiaircraft6;

import android.content.Context;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.media.SoundPool;

public class SoundManager {
```

```
	private static SoundPool sounds;
	private static int shoot;
	private static int explode;

	private static int movegun;
	private static boolean movegunPlaying ;
	private static long movegunTimer;
	private static MediaPlayer musicTheme;
	private static int theme;

	public static void loadSound(Context context) {
		movegunPlaying = false;
		sounds = new SoundPool(5, AudioManager.STREAM_MUSIC, 0);
		shoot = sounds.load(context, R.raw.shoot, 1);
		explode = sounds.load(context, R.raw.explode, 1);
		movegun = sounds.load(context, R.raw.movegun, 1);
		musicTheme = MediaPlayer.create(context, R.raw.theme);
	}


	public static final void playMusicTheme() {
		if (!musicTheme.isPlaying()) {
			musicTheme.seekTo(0);
			musicTheme.start();
		}
	}

	public static final void pauseMusic() {
		if (musicTheme.isPlaying()) musicTheme.pause();
	}

	public static void playShoot() {
		sounds.play(shoot, 1, 1, 1, 0, 1);
	}

	public static void playExplode() {
		sounds.play(explode, 1, 1, 1, 0, 1);
	}

	public static void playMovegun() {
		if (!movegunPlaying){
			sounds.play(movegun, 1, 1, 1, 0, 1);
			movegunPlaying = true;
			movegunTimer=0;
		}
	}

	public static void update(long time){
		playMusicTheme();
		if (movegunPlaying){
			movegunTimer+=time;
			if (movegunTimer>302){
				movegunPlaying=false;
				movegunTimer=0;
			}
		}
	}
```

```
}
```

Now we can invoke this methods in the other classes.
In GameThread.

1) We create an instance of the SoundManager when the thread is created.
2) During his run we invoke update.
3) When we shoot a bullet we playShoot.

The final GameThread class his the following one.

```
package pmm.antiaircraft6;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.concurrent.ArrayBlockingQueue;

import android.content.Context;
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.os.Handler;
import android.util.DisplayMetrics;
import android.view.SurfaceHolder;

public class GameThread extends Thread {
	private SurfaceHolder mSurfaceHolder;
	public DisplayMetrics metrics = new DisplayMetrics();
	private static Bitmap mBackgroundImage;
	public static Bitmap tankImg;
	public static Bitmap cannonImg;
	public static Bitmap ufoImg;
	public static boolean mRun = false;
	private Tank tank;
	public static Resources res;
	private Object inputQueueMutex = new Object();
	private ArrayBlockingQueue<InputObject> inputQueue = new
ArrayBlockingQueue<InputObject>(30);
	private static ArrayList<Bullet> bullets;
	private static ArrayList<Ufo> ufos;
	private long lastUpdateTime;
	private long newUfoTimer;
	private Hud hud;

	public GameThread(SurfaceHolder surfaceHolder, Context context, Handler handler) {
		mSurfaceHolder = surfaceHolder;
		Resources res = context.getResources();
		mBackgroundImage = BitmapFactory.decodeResource(res,R.drawable.background);
		tankImg = BitmapFactory.decodeResource(res,R.drawable.tank);
		cannonImg = BitmapFactory.decodeResource(res,R.drawable.cannon);
		ufoImg= BitmapFactory.decodeResource(res,R.drawable.ufo);
		bullets = new ArrayList<Bullet>();
		ufos = new ArrayList<Ufo>();
		SoundManager.loadSound(context);
	}
```

```
@Override
public void run() {
	while (mRun) {
		Canvas c = null;
		try {
			c = mSurfaceHolder.lockCanvas(null);
			synchronized (mSurfaceHolder) {
				long currentTime = System.currentTimeMillis();
				long delta = (long) (currentTime - lastUpdateTime);
				lastUpdateTime = currentTime;
				processInput();
				updatePhysics(delta);
				SoundManager.update(delta);
				draw(c);
			}
		} finally {
			if (c != null) {
				mSurfaceHolder.unlockCanvasAndPost(c);
			}
		}
	}
}

public void setRunning(boolean b) {
	mRun = b;
}

private void draw(Canvas canvas) {
	canvas.drawBitmap(mBackgroundImage, 0, 0, null);
	for(Iterator<Bullet> it = bullets.iterator();it.hasNext();){
		Bullet b = (Bullet) it.next();
		if (b!=null)
			b.draw(canvas);
	}

	for(Iterator<Ufo> it = ufos.iterator();it.hasNext();){
		Ufo a = (Ufo) it.next();
		if (a!=null)
			a.draw(canvas);
	}

	tank.draw(canvas);
	hud.draw(canvas);
}

public void createGraphics() {
	mBackgroundImage=Bitmap.createBitmap(mBackgroundImage);
	tank = new Tank();
	hud = new Hud();
	hud.register(tank);
}

private void updatePhysics(long timer) {
	timer=(long) (timer);
	tank.update(timer);
	Bullet b ;
```

```
		int i=0;
		while (!bullets.isEmpty() && bullets.size()>i){
			b = bullets.get(i);
			if (b.isOver())
				bullets.remove(i);
			else {
				b.update(timer);
				i++;
			}
		}

		Ufo a ;
		i=0;
		while (!ufos.isEmpty() && ufos.size()>i){
			a = ufos.get(i);
			if (a.isOver())
				ufos.remove(i);
			else {
				a.update(timer);
				int j=0;
				while (j<bullets.size()){
					b = bullets.get(j);
					if (a.isFlying() && b.isFlying())
						if(a.impactDetected(b)){
							hud.addImpact();
							a.setImpact();
							b.setImpact();
							ufos.add(new Ufo());
						}
					j++;
				}
				i++;
			}
		}
		newUfoTimer+=timer;
		if (newUfoTimer>3000){
			ufos.add(new Ufo());
			newUfoTimer=0;
		}
	}

	public void feedInput(InputObject input) {
		synchronized(inputQueueMutex) {
			try {
				inputQueue.put(input);
			} catch (InterruptedException e) {
			}
		}
	}

	private void processInput() {
		synchronized(inputQueueMutex) {
			ArrayBlockingQueue<InputObject> inputQueue = this.inputQueue;
			while (!inputQueue.isEmpty()) {
				try {
					InputObject input = inputQueue.take();
					if (input.eventType == InputObject.EVENT_TYPE_TOUCH) {
```

```
                                        processMotionEvent(input);
                                }
                                input.returnToPool();
                        } catch (InterruptedException e) {
                        }
                }
        }
    }

    private void processMotionEvent(InputObject input) {
        if( input.action==InputObject.ACTION_TOUCH_DOWN){
            tank.pressFire();
            tank.setTarget(input.x, input.y);
        }
        if( input.action==InputObject.ACTION_TOUCH_UP){
            tank.setTarget(input.x, input.y);
            tank.releaseFire();
        }
        if( input.action==InputObject.ACTION_TOUCH_MOVE)
            tank.setTarget(input.x, input.y);
    }

    public static void shootBullet(float angle, int power,int x, int y) {
        bullets.add(new Bullet(power, angle, x, y));
        SoundManager.playShoot();
    }
}
```

In Tank.

1) When we set the new target we calculate the difference between the previous and the actual angle and if it is not 0 then we playMovegun.

The final class for Tank follows here.

```
package pmm.antiaircraft6;

import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Matrix;

public class Tank {
    private static final int TANK_HEIGHT = 50;
    private static final int TANK_WIDTH = 50;
    private static final int TANK_TOP = 480 - TANK_HEIGHT;
    private static final int CANNON_WIDTH = 10;
    private static final int GUNBARREL_LENGTH = 46;
    private static final int STATUS_IDLE=0;
    private static final int STATUS_POWERING=1;
    private static final long POWERING_TIMER_LIMIT = 1200;
    private float angle;
    public static int cannonEndX;
    public static int cannonEndY;
    private int status;
    private long poweringTimer;
    private int power;
```

```
	private int lastBulletPower;

	public Tank(){
		GameThread.tankImg=Bitmap.createBitmap(GameThread.tankImg);
		GameThread.cannonImg=Bitmap.createBitmap(GameThread.cannonImg);
	}

	public void draw(Canvas c) {
		c.drawBitmap(GameThread.tankImg, (320-TANK_WIDTH)/2, TANK_TOP, null);
		Matrix m = new Matrix();
		m.postTranslate((320-CANNON_WIDTH)/2, TANK_TOP - 30);
		m.postRotate((float) ((-angle)*180 /Math.PI),160,446);
		c.drawBitmap(GameThread.cannonImg, m, null);
		cannonEndX = (int) (160 + (Math.cos(angle) * GUNBARREL_LENGTH));
		cannonEndY = (int) (446 - (Math.sin(angle) * GUNBARREL_LENGTH));
	}

	public void setTarget(int x, int y) {
		float previousAngle=angle;
		if (x==160)
			angle=0;
		else{
			if(y>=446){
				angle=(float)Math.PI/2;
				if(x>160)
					angle-=2*angle;
			}
			else
				angle = (float) Math.atan((float)(x-160)/(float)(y-446));
		}
		if (Math.abs( (int) ((previousAngle-angle)*180/Math.PI))>0)
			SoundManager.playMovegun();
	}

	public void update(long elapsedTime){
		if (status == STATUS_POWERING){
			poweringTimer=poweringTimer+elapsedTime;
			if (poweringTimer>POWERING_TIMER_LIMIT)
				poweringTimer=POWERING_TIMER_LIMIT;
			power = (int) (((float)poweringTimer / POWERING_TIMER_LIMIT) *100);
		}
	}

	public void pressFire( ) {
		status = STATUS_POWERING;
		power = 0;
		poweringTimer = 0;
	}

	public void releaseFire() {
		GameThread.shootBullet(angle+(float)Math.PI/2,40+power*60/100,160,446);
		lastBulletPower = power;
		status = STATUS_IDLE;
		power = 0;
	}

	public int getPower() {
```

```
		return power;
	}

	public float getAngle() {
		return angle;
	}

	public int getLastBulletPower() {
		return lastBulletPower;
	}
}
```

In Ufo

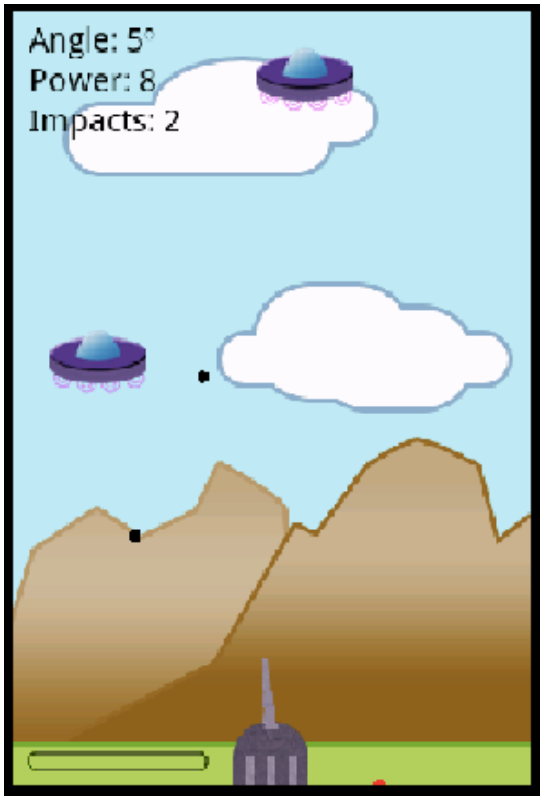
1) When an impact is detected playExplode has to be invoked.

The final Ufo class is the following.

```
package pmm.antiaircraft6;

import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Matrix;

public class Ufo {
	private static final int STATUS_FLYING = 0;
	private static final int STATUS_OVER = 1;
	private static final float TIME_FLYING = 5;

	private float lowerPosition;
	private long time;
	private int status;
	private float posX;
	private float posY;
	private double iniSpeedX;
	private double iniSpeedY;
	private float posX0;
	private float drawX;
	private float drawY;
	private int angle;
	private int direction;
	private static float acceleration;

	public Ufo(){
		time = 0;
		angle = 0;
		status = STATUS_FLYING;
		direction = (Math.random()<0.5?1:-1);
		posX0 = (float) (160 + (320*-direction*Math.random()));

		lowerPosition = 480;

		acceleration = (float) (lowerPosition/Math.pow(TIME_FLYING/2,2));

		iniSpeedX = (320 / TIME_FLYING);
		iniSpeedY = (TIME_FLYING * acceleration / 2) - 1 / TIME_FLYING;
```

```
			GameThread.ufoImg = Bitmap.createBitmap(GameThread.ufoImg);
		}

		public void draw(Canvas c) {
			Matrix m = new Matrix();
			m.postTranslate(drawX, drawY);

			if (status == STATUS_FLYING)
				c.drawBitmap(GameThread.ufoImg, m, null);
		}

		public void update(long elapsedTime) {
			double t2;
			switch (status){
				case STATUS_FLYING:
					time+=elapsedTime;
					t2=(double)(time)/1000;
					posX = (float) (posX0 +  iniSpeedX*t2*direction);
					posY = (float) (iniSpeedY * t2 - (acceleration/2 *
Math.pow(t2,2) ));
					angle = (direction>0?180:0) + (int) (Math.atan((iniSpeedY-
acceleration*t2)/iniSpeedX*direction)*180/Math.PI);
					if (posY<0)
						status=STATUS_OVER;
				break;
			}

			drawX = posX - 30;
			drawY = posY - 20;
		}

		public boolean isOver() {
			return (status==STATUS_OVER);
		}

		public boolean isFlying(){
			return (status == STATUS_FLYING);
		}

		public boolean impactDetected(Bullet b) {
			boolean impact = false;
			float diffX=Math.abs(posX-b.getPosX());
			float diffY=Math.abs(posY-b.getPosY());
			if (diffX<19 && diffY<19){
				impact = true;
				SoundManager.playExplode();
			}
			return impact;
		}

		public void setImpact(){
			status = STATUS_OVER;
		}
}
```

Now we can play our game with sounds. That's great!



5 Further improvement

There are a lot of possible improvements that allow the reader to improve their skills as Android developers.
Among the others:

- scale graphics to allow more screen resolutions;
- create different enemies with different abilities and different rewards;
- allow buying of tank upgrades (different bullets, different cannons, etc.);
- move the tank.

6 Conclusions

In this tutorial we have moved the firsts steps of Android development, learning what is necessary to download and how to set up the software artefacts.
Then we learnt some Android programming techniques and we created, step by step, a non-trivial application.
In particular the reader should now have understood how to:

- create and run projects using Android emulators;
- create Activities;
- manage Views;
- use custom graphics;
- place objects;
- interact with touch events;

- display information on screen;
- manage interaction between game elements;
- use sounds.

A list of possible improvements has been provided in order to encourage the user to deepen Android SDK and tools.

Appendix – Final classes code

The final classes are reported here to allow the user to access the final code quickly without the need to go back and forth.

Bullet

```
package pmm.antiaircraft6;

import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;

public class Bullet {
	private static final int STATUS_FLYING = 0;
	private static final int STATUS_BOOM = 1;
	private static final int STATUS_OVER = 2;
	private static final int BULLET_RADIUS  = 4;

	private static final long TIME_EXPLODING  = 100;

	private long time;

	private int status;
	private int power;
	private float posX0;
	private float posY0;
	private float posX;
	private float posY;
	private float iniSpeedX;
	private float iniSpeedY;

	private Paint paint;

	private long explodingTimer;

	public Bullet(int power,float angle,int x0,int y0){
		time=0;
		paint = new Paint();
		this.power = power;
		posX0 = x0;
		posY0 = y0;
		iniSpeedY=(float) (Math.sin(angle)*power*1.5) ;
		iniSpeedX=(float) (Math.cos(angle)*power*1.5);
	}

	public void draw(Canvas c) {
		paint.setColor(Color.RED);
		if (status == STATUS_FLYING)
			paint.setColor(Color.BLACK);
		c.drawCircle(posX, posY, BULLET_RADIUS, paint);
	}

	public void update(long elapsedTime) {
		switch (status){
```

```
			case STATUS_FLYING:
				time+=elapsedTime;
				double t2=(double)(time)/100;
				posX=(float) (posX0 + iniSpeedX*t2);
				posY=(float) (posY0 - (iniSpeedY*t2 - (9.8 *Math.pow(t2, 2)/2)));

				if (posY>=480) {
					status=STATUS_BOOM;
					posY = 480;
					explodingTimer = 0;
				}
				break;
			case STATUS_BOOM:
				explodingTimer+=elapsedTime;
				if (explodingTimer>TIME_EXPLODING)
					status=STATUS_OVER;
				break;
			}
		}

		public void setImpact(){
			status = STATUS_BOOM;
			explodingTimer=0;
		}

		public float getPosX() {
			return posX;
		}

		public float getPosY() {
			return posY;
		}

		public boolean isFlying(){
			return (status == STATUS_FLYING);
		}

		public boolean isExploding() {
			return (status==STATUS_BOOM);
		}

		public boolean isOver() {
			return (status==STATUS_OVER);
		}

}
```

GameActivity

```
package pmm.antiaircraft6;

import java.util.concurrent.ArrayBlockingQueue;

import android.app.Activity;
import android.view.MotionEvent;
import android.view.Window;
```

```
import android.view.WindowManager;

public class GameActivity extends Activity{

	private GameThread mGameThread;
	public static final int INPUT_QUEUE_SIZE = 30;
	public ArrayBlockingQueue<InputObject> inputObjectPool;

	public void onStart() {
		super.onStart();
		requestWindowFeature(Window.FEATURE_NO_TITLE);
	  getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
		setContentView(R.layout.main);
		mGameThread = ((GameView) findViewById(R.id.game)).getThread();
		mGameThread.createGraphics();
		createInputObjectPool();
	}

	private void createInputObjectPool() {
		inputObjectPool = new ArrayBlockingQueue<InputObject>(INPUT_QUEUE_SIZE);
		for (int i = 0; i < INPUT_QUEUE_SIZE; i++) {
			inputObjectPool.add(new InputObject(inputObjectPool));
		}
	}

	@Override
	public boolean onTouchEvent(MotionEvent event) {
		try {
			int hist = event.getHistorySize();
			for (int i = 0; i < hist; i++) {
				InputObject input = inputObjectPool.take();
				input.useEventHistory(event, i);
				mGameThread.feedInput(input);
			}
			InputObject input = inputObjectPool.take();
			input.useEvent(event);
			mGameThread.feedInput(input);
		} catch (InterruptedException e) {
		}
		try {
			Thread.sleep(16);
		} catch (InterruptedException e) {
		}
		return true;
	}
}
```

GameThread

```
package pmm.antiaircraft6;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.concurrent.ArrayBlockingQueue;
```

```
import android.content.Context;
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.os.Handler;
import android.util.DisplayMetrics;
import android.view.SurfaceHolder;

public class GameThread extends Thread {
	private SurfaceHolder mSurfaceHolder;
	public DisplayMetrics metrics = new DisplayMetrics();
	private static Bitmap mBackgroundImage;
	public static Bitmap tankImg;
	public static Bitmap cannonImg;
	public static Bitmap ufoImg;
	public static boolean mRun = false;
	private Tank tank;
	public static Resources res;
	private Object inputQueueMutex = new Object();
	private ArrayBlockingQueue<InputObject> inputQueue = new
ArrayBlockingQueue<InputObject>(30);
	private static ArrayList<Bullet> bullets;
	private static ArrayList<Ufo> ufos;
	private long lastUpdateTime;
	private long newUfoTimer;
	private Hud hud;

	public GameThread(SurfaceHolder surfaceHolder, Context context, Handler handler) {
		mSurfaceHolder = surfaceHolder;
		Resources res = context.getResources();
		mBackgroundImage = BitmapFactory.decodeResource(res,R.drawable.background);
		tankImg = BitmapFactory.decodeResource(res,R.drawable.tank);
		cannonImg = BitmapFactory.decodeResource(res,R.drawable.cannon);
		ufoImg= BitmapFactory.decodeResource(res,R.drawable.ufo);
		bullets = new ArrayList<Bullet>();
		ufos = new ArrayList<Ufo>();
		SoundManager.loadSound(context);
	}

	@Override
	public void run() {
		while (mRun) {
			Canvas c = null;
			try {
				c = mSurfaceHolder.lockCanvas(null);
				synchronized (mSurfaceHolder) {
					long currentTime = System.currentTimeMillis();
					long delta = (long) (currentTime - lastUpdateTime);
					lastUpdateTime = currentTime;
					processInput();
					updatePhysics(delta);
					SoundManager.update(delta);
					draw(c);
				}
			} finally {
				if (c != null) {
```

```
                                mSurfaceHolder.unlockCanvasAndPost(c);
                        }
                }
        }
}

public void setRunning(boolean b) {
        mRun = b;
}

private void draw(Canvas canvas) {
        canvas.drawBitmap(mBackgroundImage, 0, 0, null);
        for(Iterator<Bullet> it = bullets.iterator();it.hasNext();){
                Bullet b = (Bullet) it.next();
                if (b!=null)
                        b.draw(canvas);
        }

        for(Iterator<Ufo> it = ufos.iterator();it.hasNext();){
                Ufo a = (Ufo) it.next();
                if (a!=null)
                        a.draw(canvas);
        }

        tank.draw(canvas);
        hud.draw(canvas);
}

public void createGraphics() {
        mBackgroundImage=Bitmap.createBitmap(mBackgroundImage);
        tank = new Tank();
        hud = new Hud();
        hud.register(tank);
}

private void updatePhysics(long timer) {
        timer=(long) (timer);
        tank.update(timer);
        Bullet b ;
        int i=0;
        while (!bullets.isEmpty() && bullets.size()>i){
                b = bullets.get(i);
                if (b.isOver())
                        bullets.remove(i);
                else {
                        b.update(timer);
                        i++;
                }
        }

        Ufo a ;
        i=0;
        while (!ufos.isEmpty() && ufos.size()>i){
                a = ufos.get(i);
                if (a.isOver())
                        ufos.remove(i);
                else {
```

```
                        a.update(timer);
                        int j=0;
                        while (j<bullets.size()){
                              b = bullets.get(j);
                              if (a.isFlying() && b.isFlying())
                                    if(a.impactDetected(b)){
                                          hud.addImpact();
                                          a.setImpact();
                                          b.setImpact();
                                          ufos.add(new Ufo());
                                    }
                              j++;
                        }
                        i++;
                  }
            }
            newUfoTimer+=timer;
            if (newUfoTimer>3000){
                  ufos.add(new Ufo());
                  newUfoTimer=0;
            }
      }

      public void feedInput(InputObject input) {
            synchronized(inputQueueMutex) {
                  try {
                        inputQueue.put(input);
                  } catch (InterruptedException e) {
                  }
            }
      }

      private void processInput() {
            synchronized(inputQueueMutex) {
                  ArrayBlockingQueue<InputObject> inputQueue = this.inputQueue;
                  while (!inputQueue.isEmpty()) {
                        try {
                              InputObject input = inputQueue.take();
                              if (input.eventType == InputObject.EVENT_TYPE_TOUCH) {
                                    processMotionEvent(input);
                              }
                              input.returnToPool();
                        } catch (InterruptedException e) {
                        }
                  }
            }
      }

      private void processMotionEvent(InputObject input) {
            if( input.action==InputObject.ACTION_TOUCH_DOWN){
                  tank.pressFire();
                  tank.setTarget(input.x, input.y);
            }
            if( input.action==InputObject.ACTION_TOUCH_UP){
                  tank.setTarget(input.x, input.y);
                  tank.releaseFire();
            }
```

```
			if( input.action==InputObject.ACTION_TOUCH_MOVE)
				tank.setTarget(input.x, input.y);
		}

		public static void shootBullet(float angle, int power,int x, int y) {
			bullets.add(new Bullet(power, angle, x, y));
			SoundManager.playShoot();
		}
}
```

GameView

```
package pmm.antiaircraft6;

import android.content.Context;
import android.util.AttributeSet;
import android.view.SurfaceHolder;
import android.view.SurfaceView;

public class GameView extends SurfaceView implements SurfaceHolder.Callback {
		private GameThread thread;

		public GameView(Context context, AttributeSet attrs) {
			super(context, attrs);
			SurfaceHolder holder = getHolder();
			holder.addCallback(this);
			thread = new GameThread(holder, context,null);
		}

		public GameThread getThread() {
			return thread;
		}

		public void surfaceCreated(SurfaceHolder holder) {
			thread.setRunning(true);
			thread.start();
		}

		@Override
		public void surfaceChanged(SurfaceHolder holder, int format, int width,
				int height) {}

		@Override
		public void surfaceDestroyed(SurfaceHolder holder) {}
}
```

Hud

```
package pmm.antiaircraft6;

import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.LinearGradient;
import android.graphics.Paint;
import android.graphics.Paint.Style;
```

```
import android.graphics.RectF;
import android.graphics.Shader;

public class Hud {
	private static final int BAR_POWER_LEFT_MARGIN = 10;
	private static final int BAR_POWER_RIGHT_MARGIN = 200;
	private static final int BAR_POWER_BOTTOM_MARGIN = 10;
	private static final int BAR_POWER_HEIGHT = 10;
	private static final int TEXT_INFO_LEFT_MARGIN = 10;
	private static final int TEXT_ANGLE_TOP_MARGIN = 25;
	private static final int TEXT_POWER_TOP_MARGIN = 50;
	private static final int TEXT_COUNTER_TOP_MARGIN = 75;
	private static final int TEXT_SIZE = 20;
	private float barPwrLeft;
	private float barPwrTop;
	private float barPwrRight;
	private float barPwrBottom;
	private int impactCounter;
	LinearGradient gradient;
	Paint paint;
	private Tank tank;

	public Hud(){
		paint = new Paint();

		barPwrLeft = BAR_POWER_LEFT_MARGIN;
		barPwrTop = 480 - BAR_POWER_BOTTOM_MARGIN - BAR_POWER_HEIGHT;
		barPwrRight = 320 - BAR_POWER_RIGHT_MARGIN;
		barPwrBottom = 480 - BAR_POWER_HEIGHT;
		impactCounter =0;
		gradient= new LinearGradient (barPwrLeft, barPwrTop, barPwrRight, barPwrTop,
new int[]{Color.GREEN,Color.YELLOW, Color.RED},null, Shader.TileMode.CLAMP);
	}

	public void draw(Canvas c){
		int progress = ((int) (barPwrLeft + tank.getPower() * (barPwrRight-
barPwrLeft) /100));
		paint.setAlpha(255);
		paint.setShader(gradient);
		paint.setStyle(Style.FILL);
		c.drawRoundRect(new RectF(barPwrLeft, barPwrTop, progress, barPwrBottom), 4,
4, paint);
		paint.setShader(null);
		paint.setStyle(Style.STROKE);
		paint.setColor(Color.BLACK);
		paint.setStrokeWidth(0);
		c.drawRoundRect(new RectF(barPwrLeft, barPwrTop, barPwrRight, barPwrBottom),
4, 4, paint);

		paint.setColor(Color.BLACK);
		paint.setStyle(Paint.Style.STROKE);
		paint.setStrokeWidth(0);
		paint.setAntiAlias(true);
		paint.setTextSize(TEXT_SIZE);
		int
powerInfo=(tank.getPower()==0?tank.getLastBulletPower():tank.getPower());
		c.drawText("Angle: "+Math.abs((int) (tank.getAngle()*180/Math.PI))+"°",
```

```
TEXT_INFO_LEFT_MARGIN, TEXT_ANGLE_TOP_MARGIN, paint);
            c.drawText("Power: "+powerInfo, TEXT_INFO_LEFT_MARGIN, TEXT_POWER_TOP_MARGIN,
paint);
            c.drawText("Impacts: "+impactCounter, TEXT_INFO_LEFT_MARGIN,
TEXT_COUNTER_TOP_MARGIN, paint);
      }

      public void register(Tank tank) {
            this.tank = tank;
      }

      public void addImpact() {
            impactCounter++;
      }
}
```

InputObject

```
package pmm.antiaircraft6;

import java.util.concurrent.ArrayBlockingQueue;
import android.view.MotionEvent;

public class InputObject {
      public static final byte EVENT_TYPE_KEY = 1;
      public static final byte EVENT_TYPE_TOUCH = 2;
      public static final int ACTION_KEY_DOWN = 1;
      public static final int ACTION_KEY_UP = 2;
      public static final int ACTION_TOUCH_DOWN = 3;
      public static final int ACTION_TOUCH_MOVE = 4;
      public static final int ACTION_TOUCH_UP = 5;
      public ArrayBlockingQueue<InputObject> pool;
      public byte eventType;
      public long time;
      public int action;
      public int keyCode;
      public int x;
      public int y;

      public InputObject(ArrayBlockingQueue<InputObject> pool) {
            this.pool = pool;
      }

      public void useEvent(MotionEvent event) {
            eventType = EVENT_TYPE_TOUCH;
            int a = event.getAction();
            switch (a) {
            case MotionEvent.ACTION_DOWN:
                  action = ACTION_TOUCH_DOWN;
                  break;
            case MotionEvent.ACTION_MOVE:
                  action = ACTION_TOUCH_MOVE;
                  break;
            case MotionEvent.ACTION_UP:
                  action = ACTION_TOUCH_UP;
                  break;
```

```
			default:
				action = 0;
			}
			time = event.getEventTime();
			x = (int) event.getX() ;
			y = (int) event.getY();
		}

		public void useEventHistory(MotionEvent event, int historyItem) {
			eventType = EVENT_TYPE_TOUCH;
			action = ACTION_TOUCH_MOVE;
			time = event.getHistoricalEventTime(historyItem);
			x = (int) event.getHistoricalX(historyItem);
			y = (int) event.getHistoricalY(historyItem);
		}

		public void returnToPool() {
			pool.add(this);
		}
}
```

SoundManager

```
package pmm.antiaircraft6;

import android.content.Context;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.media.SoundPool;

public class SoundManager {

		private static SoundPool sounds;
		private static int shoot;
		private static int explode;

		private static int movegun;
		private static boolean movegunPlaying ;
		private static long movegunTimer;
		private static MediaPlayer musicTheme;
		private static int theme;

		public static void loadSound(Context context) {
			movegunPlaying = false;
			sounds = new SoundPool(5, AudioManager.STREAM_MUSIC, 0);
			shoot = sounds.load(context, R.raw.shoot, 1);
			explode = sounds.load(context, R.raw.explode, 1);
			movegun = sounds.load(context, R.raw.movegun, 1);
			musicTheme = MediaPlayer.create(context, R.raw.theme);
		}


		public static final void playMusicTheme() {
			if (!musicTheme.isPlaying()) {
				musicTheme.seekTo(0);
				musicTheme.start();
```

```
			}
		}

		public static final void pauseMusic() {
			if (musicTheme.isPlaying()) musicTheme.pause();
		}

		public static void playShoot() {
			sounds.play(shoot, 1, 1, 1, 0, 1);
		}

		public static void playExplode() {
			sounds.play(explode, 1, 1, 1, 0, 1);
		}

		public static void playMovegun() {
			if (!movegunPlaying){
				sounds.play(movegun, 1, 1, 1, 0, 1);
				movegunPlaying = true;
				movegunTimer=0;
			}
		}

		public static void update(long time){
			playMusicTheme();
			if (movegunPlaying){
				movegunTimer+=time;
				if (movegunTimer>302){
					movegunPlaying=false;
					movegunTimer=0;
				}
			}
		}
}
```

Tank

```
package pmm.antiaircraft6;

import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Matrix;

public class Tank {
		private static final int TANK_HEIGHT = 50;
		private static final int TANK_WIDTH = 50;
		private static final int TANK_TOP = 480 - TANK_HEIGHT;
		private static final int CANNON_WIDTH = 10;
		private static final int GUNBARREL_LENGTH = 46;
		private static final int STATUS_IDLE=0;
		private static final int STATUS_POWERING=1;
		private static final long POWERING_TIMER_LIMIT = 1200;
		private float angle;
		public static int cannonEndX;
		public static int cannonEndY;
		private int status;
```

```
	private long poweringTimer;
	private int power;
	private int lastBulletPower;

	public Tank(){
		GameThread.tankImg=Bitmap.createBitmap(GameThread.tankImg);
		GameThread.cannonImg=Bitmap.createBitmap(GameThread.cannonImg);
	}

	public void draw(Canvas c) {
		c.drawBitmap(GameThread.tankImg, (320-TANK_WIDTH)/2, TANK_TOP, null);
		Matrix m = new Matrix();
		m.postTranslate((320-CANNON_WIDTH)/2, TANK_TOP - 30);
		m.postRotate((float) ((-angle)*180 /Math.PI),160,446);
		c.drawBitmap(GameThread.cannonImg, m, null);
		cannonEndX = (int) (160 + (Math.cos(angle) * GUNBARREL_LENGTH));
		cannonEndY = (int) (446 - (Math.sin(angle) * GUNBARREL_LENGTH));
	}

	public void setTarget(int x, int y) {
		float previousAngle=angle;
		if (x==160)
			angle=0;
		else{
			if(y>=446){
				angle=(float)Math.PI/2;
				if(x>160)
					angle-=2*angle;
			}
			else
				angle = (float) Math.atan((float)(x-160)/(float)(y-446));
		}
		if (Math.abs( (int) ((previousAngle-angle)*180/Math.PI))>0)
			SoundManager.playMovegun();
	}

	public void update(long elapsedTime){
		if (status == STATUS_POWERING){
			poweringTimer=poweringTimer+elapsedTime;
			if (poweringTimer>POWERING_TIMER_LIMIT)
				poweringTimer=POWERING_TIMER_LIMIT;
			power = (int) (((float)poweringTimer / POWERING_TIMER_LIMIT) *100);
		}
	}

	public void pressFire( ) {
		status = STATUS_POWERING;
		power = 0;
		poweringTimer = 0;
	}

	public void releaseFire() {
		GameThread.shootBullet(angle+(float)Math.PI/2,40+power*60/100,160,446);
		lastBulletPower = power;
		status = STATUS_IDLE;
		power = 0;
	}
```

```
	public int getPower() {
		return power;
	}

	public float getAngle() {
		return angle;
	}

	public int getLastBulletPower() {
		return lastBulletPower;
	}
}
```

Ufo

```
package pmm.antiaircraft6;

import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Matrix;

public class Ufo {
	private static final int STATUS_FLYING = 0;
	private static final int STATUS_OVER = 1;
	private static final float TIME_FLYING = 5;

	private float lowerPosition;
	private long time;
	private int status;
	private float posX;
	private float posY;
	private double iniSpeedX;
	private double iniSpeedY;
	private float posX0;
	private float drawX;
	private float drawY;
	private int angle;
	private int direction;
	private static float acceleration;

	public Ufo(){
		time = 0;
		angle = 0;
		status = STATUS_FLYING;
		direction = (Math.random()<0.5?1:-1);
		posX0 = (float) (160 + (320*-direction*Math.random()));

		lowerPosition = 480;

		acceleration = (float) (lowerPosition/Math.pow(TIME_FLYING/2,2));

		iniSpeedX = (320 / TIME_FLYING);
		iniSpeedY = (TIME_FLYING * acceleration / 2) - 1 / TIME_FLYING;

		GameThread.ufoImg = Bitmap.createBitmap(GameThread.ufoImg);
```

```
	}

	public void draw(Canvas c) {
		Matrix m = new Matrix();
		m.postTranslate(drawX, drawY);

		if (status == STATUS_FLYING)
			c.drawBitmap(GameThread.ufoImg, m, null);
	}

	public void update(long elapsedTime) {
		double t2;
		switch (status){
			case STATUS_FLYING:
				time+=elapsedTime;
				t2=(double)(time)/1000;
				posX = (float) (posX0 +  iniSpeedX*t2*direction);
				posY = (float) (iniSpeedY * t2 - (acceleration/2 *
Math.pow(t2,2) ));
				angle = (direction>0?180:0) + (int) (Math.atan((iniSpeedY-
acceleration*t2)/iniSpeedX*direction)*180/Math.PI);
				if (posY<0)
					status=STATUS_OVER;
			break;
		}

		drawX = posX - 30;
		drawY = posY - 20;
	}

	public boolean isOver() {
		return (status==STATUS_OVER);
	}

	public boolean isFlying(){
		return (status == STATUS_FLYING);
	}

	public boolean impactDetected(Bullet b) {
		boolean impact = false;
		float diffX=Math.abs(posX-b.getPosX());
		float diffY=Math.abs(posY-b.getPosY());
		if (diffX<19 && diffY<19){
			impact = true;
			SoundManager.playExplode();
		}
		return impact;
	}

	public void setImpact(){
		status = STATUS_OVER;
	}
}
```