

# Project Report

## Group 4

Feng Chang  
Shufan Huang  
Xiangyu Lin  
Qingshan Yao

May 18, 2017

## 1 Introduction

This semester, our group successfully finished making an app named Gravity Snake, which is a combination of traditional "snake" game and acceleration sensor.

It is a simple game that everyone can enjoy. As an implementation of the classic Game Snake, you can control a serpent roaming around the starry sky looking for stars.

We referred to existing codes on the Internet and made our own adjustments. We will show some details of our work in the following parts of our report.

## 2 Basic Framework and Interface

### 2.1 Basic Framework

The app consists of two major activities: welcoming activity and game activity. Welcoming activity contains a button leading to the game activity.

To make the game easier to play, we change the app to a landscape one.

The code for welcoming activity is shown below:

```

public class MainActivity extends Activity {
    public static MediaPlayer mp1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE); //Set full screen
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN, WindowManager.LayoutParams.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_main);
        Button Start = (Button) findViewById(R.id.Start);
        mp1 = MediaPlayer.create(this, R.raw.music);
        final ToggleButton musicB = (ToggleButton) findViewById(R.id.musicB);
        Button Help = (Button) findViewById(R.id.Help);
        mp1.start();
        mp1.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
            //Music loop playing
            @Override
            public void onCompletion(MediaPlayer arg0) {
                mp1.start();
                mp1.setLooping(true);
            }
        });
    }
}

```

```

musicB.setOnClickListener(new View.OnClickListener() { //Switch on or off the music
    @Override
    public void onClick(View v) {
        if (musicB.isChecked()) {
            mp1.pause();
        }
        else {
            mp1.start();
        }
    }
});
Start.setOnClickListener(new View.OnClickListener() { //Start the game
    @Override
    public void onClick(View v) { //Switch to game activity
        Intent intent1 = new Intent(MainActivity.this, GameActivity.class);
        startActivity(intent1);
    }
});
Help.setOnClickListener(new View.OnClickListener() { //Toast instructions
    @Override
    public void onClick(View v) {
        Toast toast = Toast.makeText(getApplicationContext(), "Tilt your mobile phone to control the snake", Toast.LENGTH_LONG);
        toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
        toast.show();
    }
});
}

```

```

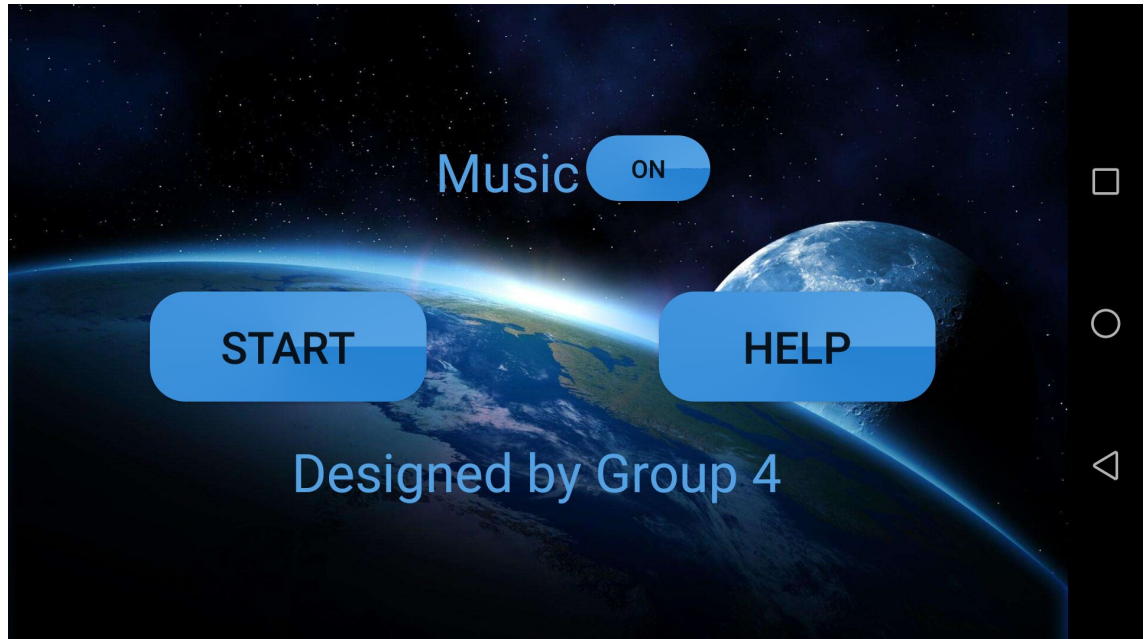
private long exitTime = 0;
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) { //Press twice to exit
    if (keyCode == KeyEvent.KEYCODE_BACK
        && event.getAction() == KeyEvent.ACTION_DOWN) {
        if ((System.currentTimeMillis() - exitTime) > 2000) {
            Toast.makeText(MainActivity.this, "再按一次退出程序", Toast.LENGTH_SHORT).show();
            exitTime = System.currentTimeMillis();
        } else {
            finish();
            System.exit(0);
        }
        return true;
    }
    return super.onKeyDown(keyCode, event);
}
}
}

```

## 2.2 Interface

We made the game appear in a "universe" style. To fit in with the style, we made targeted designing on background and button styles.

The welcoming page looks like this:



The code is shown below:

```

<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@mipmap/pic"
    tools:context="com.example.lenovo.basicinterface.MainActivity">

    <android.support.constraint.ConstraintLayout
        android:layout_width="0dp"
        android:layout_height="581dp"
        tools:layout_constraintRight_creator="1"
        app:layout_constraintRight_toRightOf="parent"
        tools:layout_constraintLeft_creator="1"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        tools:layout_constraintTop_creator="1"
        android:layout_marginTop="23dp"
        app:layout_constraintTop_toBottomOf="parent">

    </android.support.constraint.ConstraintLayout>

    <Button
        android:id="@+id/Start"
        android:layout_width="156dp"
        android:layout_height="62dp"
        android:layout_marginTop="162dp"
        android:textSize="25dp"
        android:background="@drawable/shapefile"
        android:text="Start"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:layout_constraintLeft_creator="1"
        tools:layout_constraintRight_creator="1"
        tools:layout_constraintTop_creator="1"
        app:layout_constraintHorizontal_bias="0.181" />

    <Button
        android:id="@+id/Help"
        android:layout_width="156dp"
        android:layout_height="0dp"
        android:textSize="25dp"
        android:background="@drawable/shapefile"
        android:text="Help"
        tools:layout_constraintRight_creator="1"
        tools:layout_constraintTop_creator="1"
        tools:layout_constraintBottom_creator="1"
        app:layout_constraintBottom_toBottomOf="@+id/Start"
        android:layout_marginEnd="75dp"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="@+id/Start"
        android:layout_marginRight="75dp"
        app:layout_constraintVertical_bias="0.0" />

```

```

<ToggleButton
    android:id="@+id/musicB"
    android:layout_width="70dp"
    android:layout_height="37dp"
    android:text="ToggleButton"
    android:textOff="On"
    android:textOn="Off"
    android:background="@drawable/shapefile"
    tools:layout_constraintBottom_creator="1"
    app:layout_constraintBottom_toTopOf="@+id/Help"
    tools:layout_constraintLeft_creator="1"
    app:layout_constraintLeft_toRightOf="@+id/text"
    android:layout_marginLeft="0dp"
    android:layout_marginBottom="51dp" />

```

```

<TextView
    android:id="@+id/text"
    android:layout_width="85dp"
    android:layout_height="37dp"
    android:text="Music"
    android:textSize="30dp"
    android:textColor="@color/blue"
    app:layout_constraintRight_toLeftOf="@+id/Help"
    tools:layout_constraintTop_creator="1"
    tools:layout_constraintRight_creator="1"
    tools:layout_constraintBottom_creator="1"
    app:layout_constraintBottom_toBottomOf="@+id/musicB"
    android:layout_marginEnd="41dp"
    app:layout_constraintTop_toTopOf="@+id/musicB"
    android:layout_marginRight="51dp"
    app:layout_constraintVertical_bias="0.0" />

```

```

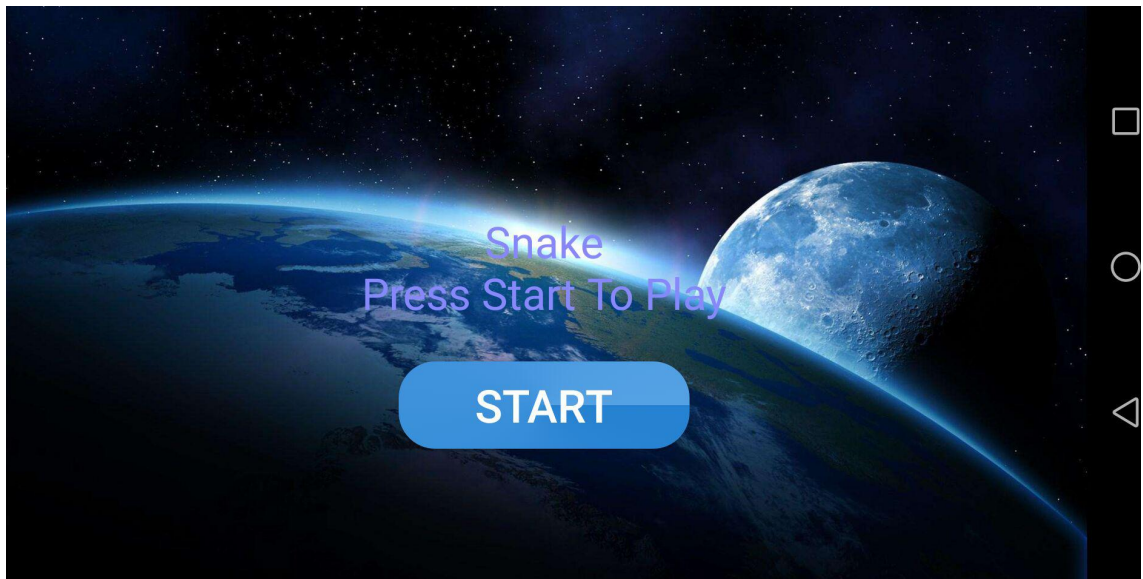
<TextView
    android:id="@+id/text2"
    android:layout_width="278dp"
    android:layout_height="52dp"
    android:text="Designed by Group 4"
    android:textSize="30dp"
    android:textColor="@color/blue"
    tools:layout_constraintTop_creator="1"
    tools:layout_constraintRight_creator="1"
    tools:layout_constraintBottom_creator="1"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    android:layout_marginTop="19dp"
    app:layout_constraintTop_toBottomOf="@+id/Help"
    tools:layout_constraintLeft_creator="1"
    android:layout_marginBottom="16dp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintVertical_bias="0.0" />
</android.support.constraint.ConstraintLayout>

```

We designed a new button style and all buttons are attached to it:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
  <corners
    android:radius="20dp" />
  <gradient
    android:startColor="#2581d1"
    android:endColor="#54a1e4"
    android:centerX="0.5"
    android:centerY="0.5"
    android:type="sweep" />
</shape>
```

the beginning page of game activity looks like this:



```

<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    >

    <com.example.lenovo.basicinterface.SnakeView
        android:id="@+id/snake"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tileSize="24"
        android:background="@mipmap/pic"
    />

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#00005500">

        <TextView
            android:id="@+id/text"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:gravity="center_horizontal"
            android:text=""
            android:textColor="#ff8888ff"
            android:textSize="24sp"
            android:visibility="visible" />

        <Button
            android:id="@+id/start"
            android:layout_width="160dp"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_centerHorizontal="true"
            android:layout_marginBottom="80dp"
            android:background="@drawable/shapefile"
            android:gravity="center"
            android:onClick="onClick_Event"
            android:text="Start"
            android:textColor="#ffffff"
            android:textSize="25dp"
            android:visibility="visible" />

    </RelativeLayout>
</FrameLayout>

```

Keep in mind that our layouts are all "land".

```

▼ layout
  activity_game.xml (land)
  activity_main.xml (land)

```

The following xml files are also used in this or other parts of this article.  
 @values/attrs.xml



```

<?xml version="1.0" encoding="utf-8"?>
<resources>
  <declare-styleable name="TileView">
    <attr name="tileSize" format="integer" />
  </declare-styleable>
</resources>

```

@values/colors.xml

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="colorPrimary">#273273</color>
  <color name="colorPrimaryDark">#303F9F</color>
  <color name="colorAccent">#ff4081</color>
  <color name="pink">#ff40fc</color>
  <color name="blue">#54a1e4</color>
</resources>

```

@values/strings.xml

```

<resources>
  <string name="app_name">basic interface</string>
  <string name="mode_ready">Snake\nPress Start To Play</string>
  <string name="mode_pause">Paused\nPress Start To Resume</string>
  <string name="mode_lose_prefix">Game Over\nScore: </string>
  <string name="mode_lose_suffix">\nPress Start To Play</string>
  <string name="button_start">Start Game!</string>
  <string name="snake_layout_text_text"/>
</resources>

```

## 3 Utilization of acceleration sensor

### 3.1 Basic utilization

Steps

```

getSystemService(SENSOR_SERVICE);
getDefaultSensor(Sensor.TYPE_ACCELEROMETER)
SensorEventListener
onSensorChanged, onAccuracyChanged
Test to get some statics used in later function:

```



```

<span style="font-family:'Comic Sans MS';font-size:18px;">↵
public class MainActivity extends Activity ↵
{ ↵
    private TextView accelerometerView; ↵
    private TextView orientationView; ↵
    private SensorManager sensorManager; ↵
    private MySensorEventListener sensorEventListener; ↵
    @Override ↵
    public void onCreate(Bundle savedInstanceState) ↵
    { ↵
        super.onCreate(savedInstanceState); ↵
        setContentView(R.layout.main); ↵
        ↵
        sensorEventListener = new MySensorEventListener(); ↵
        accelerometerView = (TextView) this.findViewById(R.id.accelerometerView); ↵
        orientationView = (TextView) this.findViewById(R.id.orientationView); ↵
        //获取感应器管理器 ↵
        sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE); ↵
    } ↵
    ↵
    @Override ↵
    protected void onResume() ↵
    { ↵
        //获取方向传感器 ↵
        Sensor orientationSensor = sensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION); ↵
        sensorManager.registerListener(sensorEventListener, orientationSensor, SensorManager.SEN
        SOR_DELAY_NORMAL); ↵
        ↵
        //获取加速度传感器 ↵
        Sensor accelerometerSensor = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMET
        ER); ↵
    }

```

```

| sensorManager.registerListener(sensorEventListener, accelerometerSensor, SensorManager.SEN
SOR_DELAY_NORMAL);
    super.onResume();
}
private final class MySensorEventListener implements SensorEventListener
{
    //可以得到传感器实时测量出来的变化值
    @Override
    public void onSensorChanged(SensorEvent event)
    {
        //得到方向的值
        if(event.sensor.getType()==Sensor.TYPE_ORIENTATION)
        {
            float x = event.values[SensorManager.DATA_X];
            float y = event.values[SensorManager.DATA_Y];
            float z = event.values[SensorManager.DATA_Z];
            orientationView.setText("Orientation: " + x + ", " + y + ", " + z);
        }
        //得到加速度的值
        else if(event.sensor.getType()==Sensor.TYPE_ACCELEROMETER)
        {
            float x = event.values[SensorManager.DATA_X];
            float y = event.values[SensorManager.DATA_Y];
            float z = event.values[SensorManager.DATA_Z];
            accelerometerView.setText("Accelerometer: " + x + ", " + y + ", " + z);
        }
    }
    //重写变化
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy)
    {

```

---

```

-
//重写变化 ↵
@Override ↵
public void onAccuracyChanged(Sensor sensor, int accuracy) ↵
{ ↵
} ↵
} ↵
↵
//暂停传感器的捕获 ↵
@Override ↵
protected void onPause() ↵
{ ↵
    sensorManager.unregisterListener(sensorEventListener); ↵
    super.onPause(); ↵
} ↵
} </span> ↵

```

### 3.2 Sensor to main function

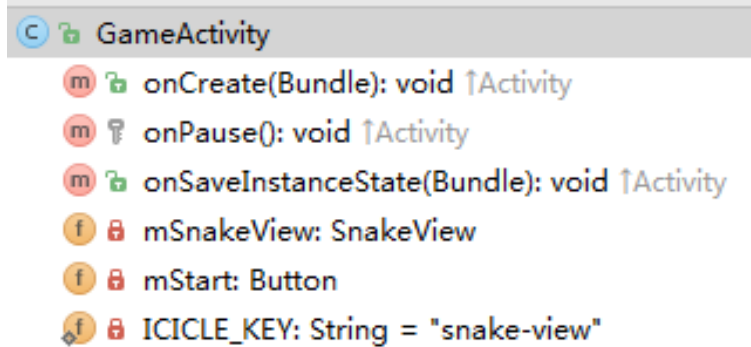
1. First step—obtain: Use tools above to get the measurable statics and store them in the valuables in common.
2. Second step—manage: Manage the threshold values and other conditions which control the movement of the snake to make it more fluent.
3. Third step—debug: According to the main function, adjust the values.

Two main problems:

- (1)How to make the snake change the direction at certain condition.
- (2)How to detect two directions when the phone is not at the right direction.

## 4 Codes

### 4.1 GameActivity



In GameActivity, we created an object mSnakeView, to execute the action defined in SnakeView.java, and initialized the program.

This function will be called when activity is first created. Turns off the title bar, sets up the content views, and fires up the SnakeView.

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_game);

    mSnakeView = (SnakeView) findViewById(R.id.snake);

    mSnakeView.setTextView((TextView) findViewById(R.id.text));
    mSnakeView.setStartButton((Button) findViewById(R.id.start));
```

If the savedInstanceState is null, i.e, we were just launched, and this will create a new game.

```
if (savedInstanceState == null) {
    mSnakeView.setMode(SnakeView.READY);
}
```

Else, we are being restored.

```

        mSnakeView.setMode(SnakeView.RESTART);
    } else {
        Bundle map = savedInstanceState.getBundle(ICICLE_KEY);
        // ...
    }
}

```

Pause the game.

```

@Override
protected void onPause() {
    super.onPause();
    mSnakeView.setMode(SnakeView.PAUSE);
}

```

Save the game.

```

@Override
public void onSaveInstanceState(Bundle outState) {
    outState.putBundle(ICICLE_KEY, mSnakeView.saveState());
}
}

```

## 4.2 TitleView

In TitleView.java, we defined specific parameters of the whole canvas:

Parameters controlling the size of tiles and their range within view,  
 Width/Height are in pixels, and Drawables will be scaled to fit to these  
 dimensions.

X/Y tile counts are the number of tiles that will be drawn.

```

public class TileView extends View {
    protected static int mTileSize;

    protected static int mXTileCount;
    protected static int mYTileCount;

    private static int mXOffset;
    private static int mYOffset;

```

Then, define a array: bitmap[], which stores different kinds of bitmap.

During the game, kinds of bitmaps will be uploaded to this array through the function restTiles and loadTile.

```

private Bitmap[] mTileArray;

```

A two dimensional array of integers in which the number represents the index of the tile that should be drawn at that locations.

It can be regarded as the canvas which we will operate on.

```

private int[][] mTileGrid;

```

In this function, we can obtain the new attribute value of tileSize in attrs.xml.

```

public TileView(Context context, AttributeSet attrs, int defStyle)
{
    super(context, attrs, defStyle);
    TypedArray a = context.obtainStyledAttributes(attrs,
        R.styleable.TileView);

    mTileSize = a.getInt(R.styleable.TileView_tileSize, 50);

    a.recycle();
}

```

And, finally, we can draws the canvas onto the mobile phone through onDraw.
















































```

@Override
public void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    for (int x = 0; x < mXTileCount; x += 1) {
        for (int y = 0; y < mYTileCount; y += 1) {
            if (mTileGrid[x][y] > 0) {
                canvas.drawBitmap(mTileArray[mTileGrid[x]
                [y]],
                                mXOffset + x * mTileSize,
                                mYOffset + y * mTileSize,
                                mPaint);
            }
        }
    }
}
}
}

```

### 4.3 SnakeView

The main action of the game is defined in SnakeView.java.

- ▼   SnakeView
  - ▼  ◦ RefreshHandler
    -   handleMessage(Message): void ↑Handler
    -   sleep(long): void
  - ▶   Coordinate
    -   SnakeView(Context, AttributeSet)
    -   SnakeView(Context, AttributeSet, int)
    -   initSnakeView(): void
    -   initNewGame(): void
    -   coordArrayListToArray(ArrayList<Coordinate>): int[]
    -   saveState(): Bundle
    -   coordArrayToArrayList(int[]): ArrayList<Coordinate>
    -   restoreState(Bundle): void
    -   setTextView(Textview): void
    -   setStartButton(Button): void
    -   setMode(int): void
    -   addRandomApple(): void
    -   update(): void
    -   updateWalls(): void
    -   updateApples(): void
    -   updateSnake(): void
    -   onClick(View): void
    -   onAccuracyChanged(Sensor, int): void ↑SensorEventListener
    -   onSensorChanged(SensorEvent): void ↑SensorEventListener



The code in this file is too long, so we can only introduce the function of all the parts.

1. In `initSnakeView`, `mSensor` and `mSensorManager` is defined so that we can get the data of sensors, including the accelerometer sensor we needed in the game. We also initialized some other variables we need later.
2. In `initNewGame`, we created a new snake, whose direction is North. And we also set two random apples in the map.
3. `coordArrayListToArray` is used to convert a list of coordinates to an array consists of integers only, in a pattern like `[x1,y1,x2,y2,x3,y3...]`.
4. `SaveState` is used to save the state so that the game can continue after the pause. The saved variables includes `mAppleList`, `mDirection`, `mNextDirection`, `mMoveDelay`, `mScore` and `mSnakeTrail`.
5. `ArrayList` is the reverse of `coordArrayListToArray`, it converts an `ArrayList` to `coordArrayList`.
6. In `restorestate`, the saved state is imported so that the game can continue.
7. `setMode` is used when the state is changed and the content of `textView` is changed according to the condition.
8. `addRandomApples` are used when one apple is eaten. It will create a new apple if there is one apple missing, and add a new coordinate to the `mAppleList`.
9. `Update`, `updateWalls`, `updateSnake`: these three functions are used to do the action and judge whether the game has ended after handling.
10. `onClick`, `onAccuracyChanged`, and `onSensorChanged` monitors the action of buttons and the state of sensors, after this the `mdirectionnext` is changed so that the snake can change its direction.

## 5 Our Reflections

Beginning as new users of Android Studio, we met tons of difficulties and problems. We're happy that we have gone through all these in the end. Though not 100 percent perfect, our app works well on the whole. I hope you'll be satisfied with our work.

During this process, we've learn much about Android Studio and have obtained basic knowledge for making an app. I'm sure the benefits will reveal themselves in our later study or in our future job.

Thanks to the professor and the teaching assistants!