

Σχεδιασμός και Ανάπτυξη Εφαρμογών Κινητού Υπολογισμού

icsd11186 Κοτζιάς Αριστείδης
icsd11044 Παντελής Ζούπης

Όνομα εφαρμογής: Shame

Περιγραφή

Η εφαρμογή ανήκει στην κατηγορία ψυχαγωγίας και έχει ως στόχο ο χρήστης να μπορεί να αναπαραστήσει την σκηνή Shame από την δημοφιλή σειρά Game of Thrones. Ο χρήστης κουνά την συσκευή του και αναπαράγεται ο συγκεκριμένος ήχος.

Η εφαρμογή χρησιμοποιεί το accelerometer από html5/javascript. Όταν ολοκληρωθεί η αναπαραγωγή του ήχου τότε αποθηκεύετε σε βάση δεδομένων ο αριθμός των αναπαραγωγών ήχου (score) από κάθε χρήστη που χρησιμοποίησε την εφαρμογή.

Για εγκατάσταση χρειάζεται μόνο το .apk.

Documentation

MainActivity



Είναι η κύρια Activity της εφαρμογής. Περιέχει την κλάση WebAppInterface και ένα Webview.

```
//javascript interface
webAppInterface = new WebAppInterface(this);

WebView webView = (WebView) findViewById(R.id.webView);
webView.getSettings().setJavaScriptEnabled(true);
webView.addJavascriptInterface(webAppInterface, "Android");
webView.loadUrl("file:///android_asset/index.html");
```

Webview

Το Webview αποτελείται από το index.html το οποίο περιέχει από μία εικόνα, τα αρχεία shame.js, jquery.js, jquery-ui.js, style.css και ένα <p>.

```
<script src="file:///android_asset/shame.js"></script>

<script src="file:///android_asset/jquery.js"></script>
<script src="file:///android_asset/jquery-ui.js"></script>
<link rel="stylesheet" href="file:///android_asset/style.css">
```

Χρησιμοποιούμε jquery για να γίνει animation το <div> που περιέχει το μήνυμα, όταν πατήσει στο webview ο χρήστης.

```
<script>

    $( document ).click(function() {
        $( "#shake" ).effect( "shake" );
    });
</script>
<br><br>
<div id='shake'>
    <p>&lt;&lt;&lt;Shake your phone to shame&gt;&gt;&gt;</p>
</div>
```

Για δυναμικό περιεχόμενο στο webview και υποστήριξη διάφορου μεγέθους οθόνες χρησιμοποιούμε το εξής viewport.

```
<meta name="viewport" content="width=device-width,initial-scale=1.0, user-scalable=yes">
```

style.css

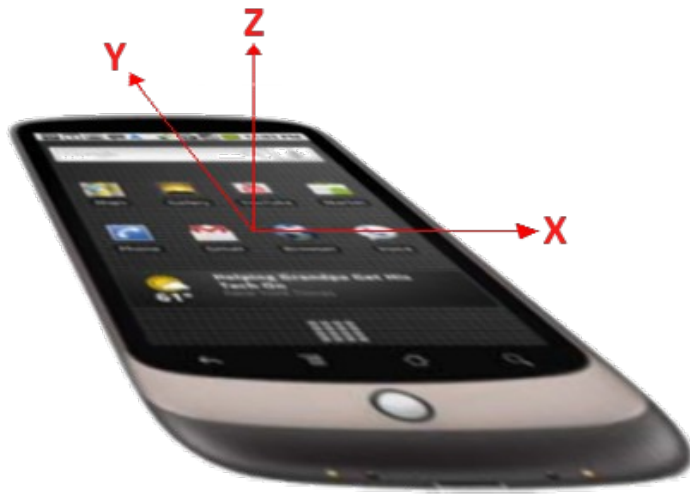
για δυναμικές εικόνες

```
img {

    max-width: 100%;
    height: auto;
}
```

shame.js

html5 accelerometer



```
var x =  
event.accelerationIncludingGravity.x;  
var y =  
event.accelerationIncludingGravity.y;  
var z =  
event.accelerationIncludingGravity.z;  
var r = event.rotationRate;  
if (max_a<r.alpha) max_a=r.alpha;  
if (max_b<r.beta) max_b=r.beta;  
if (max_g<r.gamma) max_g=r.gamma;
```


Το accelerometer καταγράφει συνεχώς τιμές για τους άξονες x,y,z. Οι γωνίες α, β, γ είναι η μεταβολή που έγινε κατά την διάρκεια της κίνησης της συσκευής, για τους άξονες x,y,z αντίστοιχα. Καταγράφουμε τη μέγιστη μεταβολή στις μεταβλητές max_a,max_b,max_g αντίστοιχα.

10:23

Project

STATISTICS

alpha: 35.731653273942854
beta: 17.920637358382226
gamma: 8.120584968395264
Acceleration:
x: -1.1875240802764893
y: 2.7964274883270264
z: 8.9064302444458
Rotation rate:
alpha: 0.05742133408784866
beta: -0.059864792972803116
gamma: -0.0403171069920063
max_a: 7.495316505432129
max_b: 11.602774620056152
max_g: 3.802025318145752



Όταν το max_g δηλαδή όταν η διαφορά της γωνίας στον άξονα z ξεπεράσει τις 10 μοίρες τότε ενεργοποιείται το shake. Μόλις περάσει στην συνθήκη αυτή τότε εμφανίζεται toast με την τιμή της max_g και η μεταβλητή μηδενίζεται. Επίσης με την συνάρτηση ding() αναπαράγεται ήχος.

```
//shake triggered
if (max_g > 10.0){
    Android.showToast(max_g);
    max_g = 0.0;
    Android.ding();
}
```

WebAppInterface

Σε αυτή την κλάση γίνεται διαχείριση των βάσεων δεδομένων. Επίσης χρησιμοποιείται σαν μέσο επικοινωνίας του κώδικα javascript στον webview με τον android κώδικα (javascript interface με το όνομα Android).

```
WebView webView = (WebView) findViewById(R.id.webView);
webView.getSettings().setJavaScriptEnabled(true);
webView.addJavascriptInterface(webAppInterface, "Android");
webView.loadUrl("file:///android_asset/index.html");
```

Constructor

```
WebAppInterface(Context c) {

    mContext = c;
    mediaPlayer = MediaPlayer.create(mContext, R.raw.ding);
    top10 = new ArrayList<>();
    initDB();
}
```

Αρχικοποιεί το context από την MainActivity για να εμφανίζει toast, MediaPlayer για αναπαραγωγή ήχου και μία ArrayList που θα περιέχει του top 10 χρήστες.

Για διαχείριση των ΒΔ χρησιμοποιήσαμε το Parse framework (<https://parse.com/>).

```
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.1.1'
    compile 'com.android.support:design:23.1.1'
    //parse
    compile 'com.parse.bolts:bolts-android:1.4.0'
    compile 'com.parse:parse-android:1.12.0'
}
```

Το parse αυτοματοποιεί το object_id που αποθηκεύεται. Αποθηκεύουμε το user_id τοπικά σε Shared Preferences για να ξέρουμε ποιός είναι ο χρήστης της εφαρμογής και να μπορούμε να δείξουμε λεπτομέρεις σχετικά με το score του. Όταν ο χρήστης κουνήσει την συσκευή του τότε παίρνει +1 στο score.

```
public void initDB(){
```

```
Parse.enableLocalDatastore(mContext);  
Parse.initialize(mContext);
```

Δημιουργία local και remote ΒΔ.

```
//load user_id  
SharedPreferences sp = mContext.getSharedPreferences("prefs", 0);  
user_id = sp.getString("user_id", "notset");
```

Διαβάζει το user_id του χρήστη από τα SharedPreferences.

```
if (user_id.equals("notset") ){  
    //create new user_id  
    createNewObject();
```

Αν δεν βρεθεί το user_id τότε δημιουργείται νέο ParseObject. Για αυτή την διαδικασία χρειάζεται συνδεση στο διαδύκτιο μόνο την 1η φορά, για να δημιουργηθεί id από το Parse και να μην υπάρχει conflict μεταξύ τα id των χρηστών.

```
gameScore.save();  
gameScore.pin();
```

Όταν δημιουργηθεί το αντικείμενο στην remote ΒΔ (save()) τότε αποθηκεύεται και στην local (pin()).

```
ParseQuery<ParseObject> query = ParseQuery.getQuery("ding");  
query.fromLocalDatastore();  
query.getInBackground(user_id, new GetCallback<ParseObject>() {  
    @Override  
    public void done(ParseObject object, com.parse.ParseException e) {
```

```
...
```

Αν υπήρχε ήδη user_id στα SharedPreferences τότε κάνει load το αντικείμενο από την local ΒΔ.

```
@JavascriptInterface  
public void ding() {  
    if (!mediaPlayer.isPlaying()) {  
        mediaPlayer.start();  
        gameScore.increment("count", 1);  
        gameScore.saveEventually();  
    }  
}
```

Αν δεν απαράζεται ήδη ο ήχος τότε ξεκινά η αναπαραγωγή του ήχου και αυξάνεται κατά 1 στο score του χρήστη. Η συνάρτηση saveEventually() αποθηκεύει το αντικείμενο στην local ΒΔ και μόλις συνδεθεί ο χρήστης στο internet τότε το κάνει upload και στην remote ΒΔ.

```

private static String calculateTime(int score) {
    int totalSeconds = score*7;//to ding pai 7 defterolepta
    int hours = totalSeconds/60/60;
    totalSeconds -= hours*60*60;
    int minutes = totalSeconds/60;
    totalSeconds -= minutes*60;
    return hours+"h "+minutes+"m "+totalSeconds+"s";
}

```

Υπολογισμός σε ώρες,λεπτά,δευτερόλεπτα του χρόνου που έκανε “shaming” ο χρήστης. 0 υπολογισμός γίνεται σύμφωνα με το τον χρόνο αναπαραγωγής του ήχου (7 “).

```

public static void getTotalShames(final Context context, final boolean localDB){
    final ParseQuery<ParseObject> query = ParseQuery.getQuery("ding");
    query.whereGreaterThan("count", 0);
    query.addDescendingOrder("count");
}

```

Παίρνει όλους τους χρήστες από την ΒΔ (true για local, false για remote) οι οποίοι έχουν score μεγαλύτερο του 0. Υπολογίζει το top10 και το rank του χρήστη καθώς και το συνολικό score και time όλων το χρηστών.

```

public static void getUserScore(final Context context,boolean localDB){
    //get user id from prefs
    SharedPreferences sp = context.getSharedPreferences("prefs", 0);
    final String user_id = sp.getString("user_id", "notset");
    //get object from db
    ParseQuery<ParseObject> query = ParseQuery.getQuery("ding");
    if (localDB)
        query.fromLocalDatastore();
    query.getInBackground(user_id, new GetCallback<ParseObject>() {
}

```

...

Παίρνει το αντικείμενο του χρήστη και αποθηκεύει το score και το time του στα TextFields.

```

...
} else {
    //error
    showToast2(context, e.getMessage());
    showToast2(context,"You need to connect to the internet for the first time.");
}

```

Ενημερώνει τον χρήστη να ενεργοποιήσει το internet για πρώτη φορά.

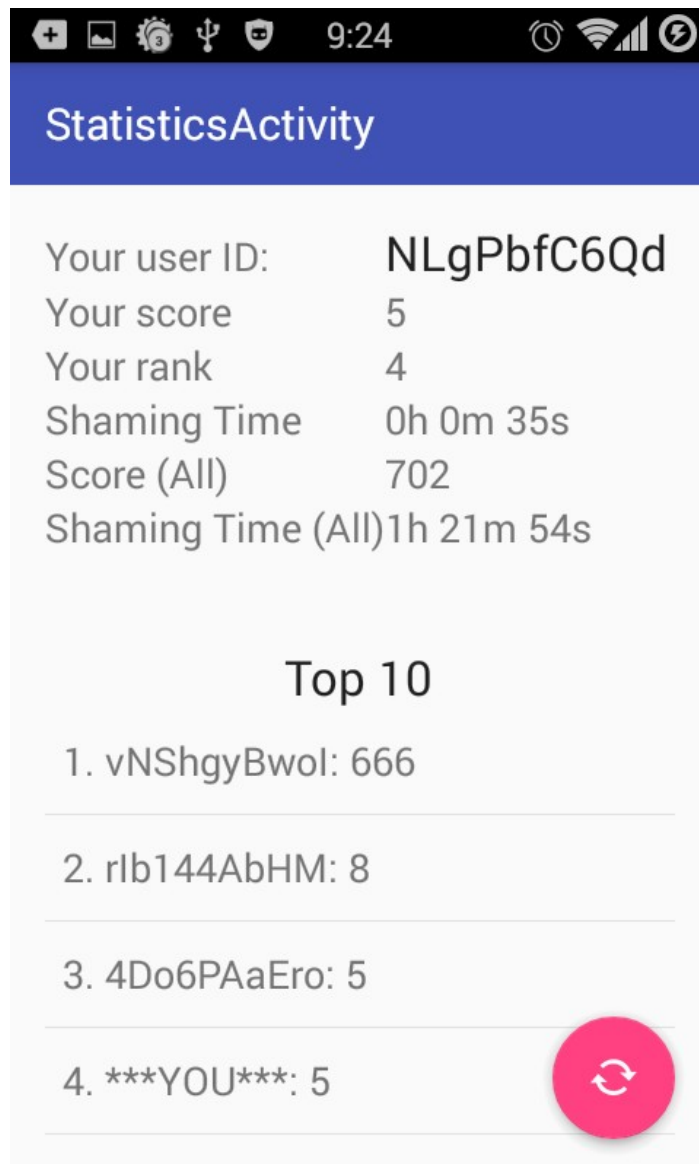
```

//refresh
public static void refreshFromRemoteDB(Context context){
    getUserScore(context,false);
    getTotalShames(context, false);
}

```

Ανανεώνει τα δεδομένα από την remote ΒΔ.

StatisticsActivity



Αποτελείται από 12 TextViews που δείχνουν το user ID, το score, time, rank του χρήστη αλλά και τα συνολικά score, time από όλους του χρήστες. Επίσης υπάρχει ListView που δείχνει τους top10 χρήστες. Η ανανέωση των δεδομένων από την remote ΒΔ γίνεται με το ροζ FloatingActionButton.

Η ανανέωση των TextFields γίνεται από τις στατικές συναρτήσεις της WebAppInterface.

```
Ανανέωση στοιχείων χρήστη: WebAppInterface.getUserScore(this, true);
Ανανέωση συνολικών στοιχείων: WebAppInterface.getTotalShames(this, true);
Ανανέωση από remote ΒΔ:
FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        WebAppInterface.refreshFromRemoteDB(StatisticsActivity.this);
        Snackbar.make(view, "Getting Data from Remote DB", Snackbar.LENGTH_LONG)
            .setAction("Action", null).show();
    }
});
```