Mobile App Development With PhoneGap

Introduction – Building mobile apps

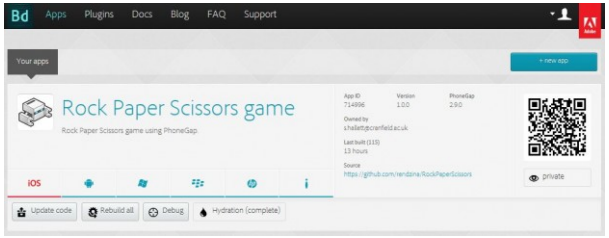
One area of computing we are developing a keen interest in here at Cranfield University is the development of mobile apps. The future is firmly mobile and meeting the explosive growth in mobile phone and tablet computing throws traditional software development approaches in the air. New approaches are needed to develop app tools for location-based mapping and GIS. With much to learn to achieve this, this article outlines some of the basic steps needed to develop apps for a mobile device. The app developed here will be a simple 'RockPaperScissors' game, ported to an Android phone, and using the PhoneGap development platform. You will need an Android device.

Cross-platform development

The first issue to recognise is the sheer diversity of mobile devices and operating systems available – with Apple IOS, Google Android, Windows Phone and Blackberry to choose from by example. Each platform has its own preferred development tools and deployment approaches. Developing the personal skills to develop native code apps for each of these platforms would be a huge task. So ideally a means is needed to allow the development of one set of code that can then be ported across these platforms. A number of tools exist that achieve this, but one that stands out for us is the combined offering of Apache Cordova (http://cordova.apache.org/) and Adobe PhoneGap (https://build.phonegap.com/).

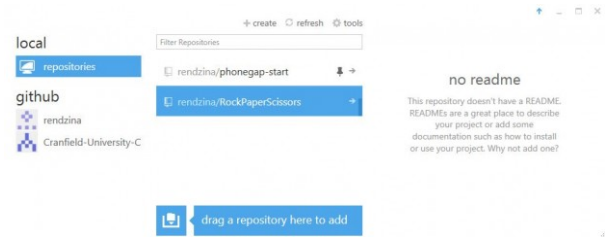
PhoneGap and Cordova

Apache Cordova is a platform for building native mobile applications using HTML, CSS and JavaScript. If you can develop a webpage, you should be able to build a Cordova app. Apache note that "Cordova graduated in October 2012 as a top level project within the Apache Software Foundation (ASF)" and that "it will always remain free and open source". The software company Adobe have then picked up Cordova and used it to produce their own SDK 'PhoneGap'. Like Cordova, PhoneGap allows native mobile app development using only HTML, CSS and JavaScript. In addition to PhoneGap itself, Adobe also provide a cloud-based mobile app compiler for PhoneGap apps, capable of producing compiled code for each of the major mobile architectures, PhoneGap Build (https://build.phonegap.com/): this is used in developing the app here.

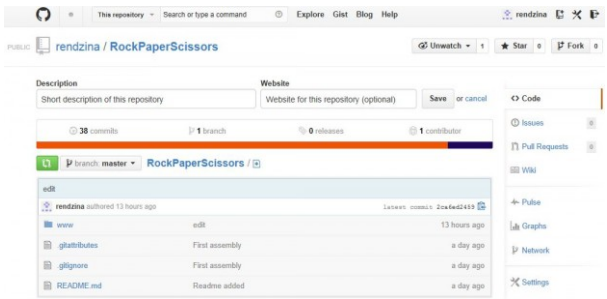


Setup – preparing to build a mobile app

PhoneGap Build accepts a package of the HTML, CSS and JavaScript code. PhoneGap Build is designed to retrieve code held in the GitHub repository (https://github.com/). Therefore, if you don't already have accounts, for this exercise you will need to create personal accounts in both PhoneGap Build, and in GitHub. You must also install and configure a copy of Git on your local development computer. For Windows PCs, the best option is 'GitHub for Windows' (http://windows.github.com/).



Once installed, Git can be linked by association with your github.com account. The local Git acts to hold a set of files linked to a software project (a 'repository'), these can then be uploaded to the GitHub website (automatically), and from there, loaded straight into the PhoneGap Build tool. Local project files (HTML, CSS and JavaScript code) can be edited by any generic text editor, for example NotePad++ (http://notepad-plus-plus.org/).



Summary of setup tasks:

1. Create account at GitHub (https://github.com/).
2. Create account at PhoneGap Build Build (https://build.phonegap.com/).
3. Install a local copy of 'GitHub for Windows' (http://windows.github.com/) – link to GitHub.
4. If needed, install a good text editor such as NotePad++ (http://notepad-plus-plus.org/).

Assembling the source code

GitHub already contains a basic PhoneGap example app to get you going. This 'repo' (repository) can be copied and then adapted to create your own custom project. To achieve this, first search for, then download as a 'zip' file the existing 'phonegap / phonegap-start' repo from the GitHub website. After creating a new local Git repo, the 'www' folder from the 'phonegap-start' app can be copied to the new repo folder, and then adapted by editing.

The basic workflow for developing a basic mobile app is as follows:

1. Log in to GitHub and locate the existing code 'phonegap / phonegap-start' (https://github.com/phonegap/phonegap-start).
2. Download this example repo as a zip file to a local file – all required source code is included.
3. Use 'GitHub for Windows' to create your own repo, called 'RockPaperScissors'.
4. Now extract the 'www' folder from the downloaded zip file and copy to the new repo folder.
5. Next, you can edit and adapt the code. For this rock paper scissors game tutorial, replace the contents of the file 'index.htm' with the following code:

```
<!DOCTYPE html>
<html>
        <head>
                <meta charset="utf-8" />
                <meta name="format-detection" content="telephone=no" />
                <meta name="viewport" content="user-scalable=no, initial-scale=1, maximum-scale=1, minimum-scale=1, width=device-width,
                <link rel="stylesheet" type="text/css" href="css/index.css" />
                <title>Rock Paper Scissors Game</title>
                <style type="text/css">
                        p, form {font-size:25px;}
                        button  {font-size:45px; width:200px; margin:25px 50px;}
                </style>
        </head>
        <body>
```

```
<h1>Rock Paper Scissors Game</h1>
<br />
<p><i>What do you choose?</i></p>
<button type="button" onclick="PlayGame('R')">Rock</button></br>
<button type="button" onclick="PlayGame('P')">Paper</button></br>
<button type="button" onclick="PlayGame('S')">Scissors</button></br>
<span id="result"> </span><br />
<span id="tally"> </span><br />
<span id="log"> </span><br />

<script type="text/javascript">
        var CHOICE = { // set up enumerations
                ROCK     : {value: 1, name: "Rock", code: "R"},
                PAPER    : {value: 2, name: "Paper", code: "P"},
                SCISSORS : {value: 3, name: "Scissors", code: "S"}
        };
        var humanWins = 0;
        var compWins = 0;

        function PlayGame(play) {
                var humanChoice;
                var compChoice;

                if (play==='R') {
                        humanChoice = CHOICE.ROCK;
                } else if (play==='P') {
                        humanChoice = CHOICE.PAPER;
                } else if (play==='S') {
                        humanChoice = CHOICE.SCISSORS;
                }

                var computer = (Math.floor( Math.random() * 3 + 0.5 ));
                        if ( computer === 1 ) {
                                compChoice = CHOICE.ROCK }
                        else { if ( computer === 2 ) {
                                compChoice = CHOICE.PAPER }
```

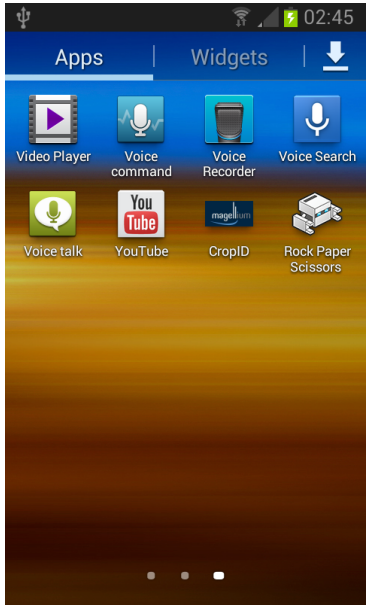
```
                        else {
                                compChoice = CHOICE.SCISSORS }
                        }
                document.getElementById('result').innerHTML="<p>You chose '" + humanChoice.name + "' and I chose '" + compChoice
                //document.getElementById('log').innerHTML+="H:" + humanChoice.code + " C:" + compChoice.code + "-";

                var win = humanChoice.value - compChoice.value;
                if ( win === 0 ) {
                        document.getElementById('result').innerHTML += "<p><b>So we draw</b></p>" }
                else {
                        if ( win === -2 || win === 1 ) {
                                document.getElementById('result').innerHTML += "<p><b>So you win</b></p>"; humanWins++; }
                        else {
                                if ( win === -1 || win === 2 ) {
                                        document.getElementById('result').innerHTML += "<p><b>So I win</b></p>"; compWins++; }
                                else {}
                        }
                }
                document.getElementById('tally').innerHTML = "<p><i>Leaderboard:</i><br />Your wins: " + humanWins + " &nbs
                if ( compWins < humanWins ) {
                        document.getElementById('tally').innerHTML += "<p>You're in the lead</p>"; }
                else if ( compWins > humanWins ) {
                        document.getElementById('tally').innerHTML += "<p>I'm in the lead</p>"; }
                else if ( compWins == humanWins ) {
                        document.getElementById('tally').innerHTML += "<p>We're running head to head</p>"; }
        }
        </script>
        <script type="text/javascript" src="js/index.js"></script>
        <script type="text/javascript">
                app.initialize();
        </script>
    </body>
</html>
```

6. Once the file is saved, be sure the local Git recognises these files as having been added to the local repo, then commit the new repo up to the GitHub website. If further edits are made to the code, be sure to 'sync' the fileset in the local Git with the GitHub, thus ensuring the latest files are copied across.

Note, a quicker alternative to these steps just to get you going, is just to create a new 'fork' (copy) of our working app – search for the public repo 'rendzina / Rock-PaperScissors' (https://github.com/rendzina/RockPaperScissors).

Compiling and installing the app

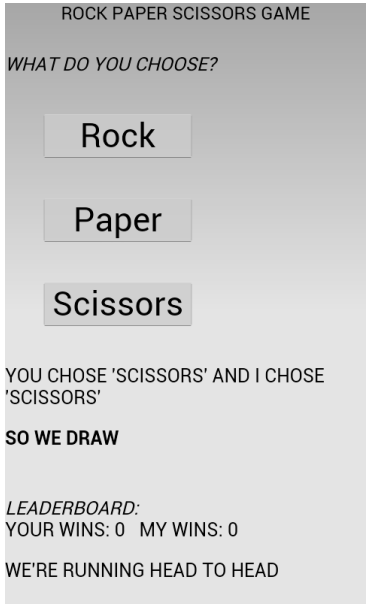
Once the app is completed to your satisfaction in the local repo, and then committed and uploaded to the GitHub, it can be accessed by PhoneGap build. Once the final version of the app is ready in GitHub, open PhoneGap Build in a browser. At this point also, you can connect the Android device physically to the local computer via a USB cable. The default USB connection will allow you to upload the compiled Android Package (apk) file to the device's 'downloads' folder, from where it can be installed.



The basic workflow for compiling and installing the app is as follows:

1. Log in to PhoneGap Build and select 'new app'.
2. You can now either upload a zipfile of the local app fileset, or direct PhoneGap Build to the Github repo by URL (e.g. https://github.com/<yourusername>/RockPaperScissors).
3. Once loaded, select 'Ready to build' to compile the app. Note that if the code was sourced from GitHub, the 'Hydration' option can usefully be selected for managing updates once it is installed on a device.
4. Once the app is compiled, select the Android icon to save the compiled 'apk' file off locally.
5. Once saved, the 'apk' file can be uploaded to the device's 'download' folder. Once uploaded, the file can then be installed directly from file on the device. Note that to do this you may need to set an option allowing apk files uploaded in this manner to be installed.

Hopefully, you can now run your app on the device!

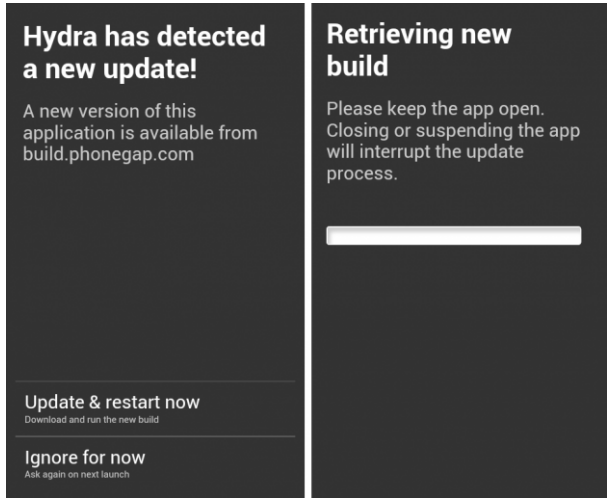


Making further edits to your code

Making further edits to the code involves simply re-editing source code files, uploading it and then recompiling the app. If GitHub was used, then making any further committed edits to the web GitHub code fileset, compiled in PhoneGap Build, will result in your being given an option to update the app directly on the device.

The basic workflow for implementing further code edits is as follows:

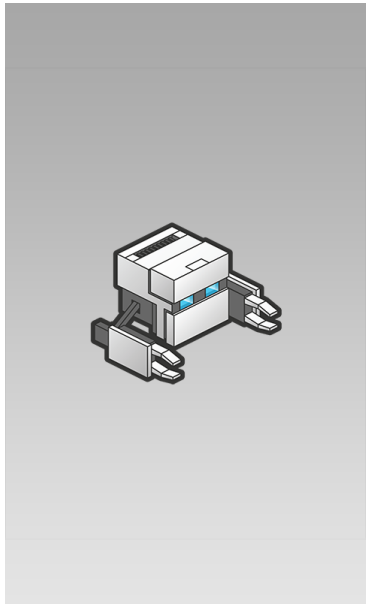
1. Make required code changes to the local app code fileset (HTML, CSS and JavaScript files).
2. In the local Git, make a new Commit for these edits, then ensure the newly committed fileset is synchronised with the web GitHub repo.
3. Once that is done, Select 'Update code' in PhoneGap Build (assuming the app source was GitHub) – the latest fileset is pulled over from GitHub and a new compile job queued.
4. If 'Hydration' was enabled, you can try running the app again and it should reload the updated app (by a wireless connection if available – you don't need to have a USB cable). If you didn't use Hydration, you will need to uninstall the app, copy over the latest apk file and reinstall before running.



Next steps

Hopefully by the end of this you will have a working mobile app using PhoneGap. However, this is just the start of mobile app development. Firstly note the user interface does not currently look like a native mobile app. This can be solved by developing the software using a further third party JavaScript library, such as Dojo (http://dojotoolkit.org/) or JQuery (http://jquery.com/), both of which have powerful mobile development options. We prefer the Dojo framework and a further blog article shows how to adapt the game here for Dojo. Developing in JavaScript also has advantages in that third party mapping and GIS APIs are also available for use, such as Leaflet (http://leafletjs.com/).

The development environment used can also be enhanced. The example above just uses a good text editor to edit the HTML, CSS and JavaScript code. A better approach is to use an Integrated Development Environment (IDE) such as Eclipse (http://www.eclipse.org/) or NetBeans (https://netbeans.org/). These are generic cross-platform tools. However, it is also perfectly possible to use the native platform-specific development tools to develop apps. For example, for Android this is the Google Android Developer Tools plugin (ADT) for Eclipse, or the new Android Studio (http://developer.android.com/tools/). These latter tools also usefully include device emulators to try out your app on before deployment. Establishing the 'software stack' required for this type of development with PhoneGap becomes more complex, as a number of dependent software tools need to be installed – this will form the basis for a later article here on GeoThread.



This entry was posted in Informatics, Mobile and tagged app, dojo, JavaScript, mobile, rock paper scissors on December 31, 2013 [http://www.geothread.net/mobile-app-development-with-phonegap/] .