OK Glass, compile my code

A quick overview about Glassware development at DroidCon Italy

by Roberto Orgiu

Timeline

- Formed by **640x360** pixels cards containing data from past, present and future, displaying in the center the **nearest events**
- Contains live and static cards
- Menu allowing to read aloud, share, reply and more
- Processes user input and starts Glassware
- **Home:** stays in timeline's centers, provides system wide touch and voice command

Live cards

- Contain **information important for the moment**
- Get removed when not relevant (or when system needs resources or is rebooting)
- Constantly updated with useful information
- Can be **only** created with GDK because they need access to low level components not available with Mirror API
- **High frequency**: renders many times a seconds and can show rich 2D and 3D content
- **Low frequency**: renders once every few seconds and it's mainly used status information not requiring a live update

Static cards

- Appear at the **right side** of clock home
- Created with both Mirror APIs and GDK
- Usually clear and easy to read
- Used as notifications, can stay in the timeline up to 7 days
- **Mirror APIs**: HTML and CSS, can create *Bundled* and *Paginated* cards
- **GDK**: `Card` object with `TimelineManager`

Immersions

- Created with **GDK** only
- Temporarily **take over the timeline** to provide user with a customized experience
- Can consume user input, **swipe gestures** included
- Closed by swiping down

Menus and inputs

- Created with Mirror APIs or within *Live cards* and *Immersions* with GDK
- Menu items always have a **white 50*50 px icon**
- Menu items' titles long about 15 characters with imperative sentences (Share, Record a sound)
- **Contextual voice input** available with Mirror APIs and GDK
- **Touch gestures** implementable with GDK

GDK features

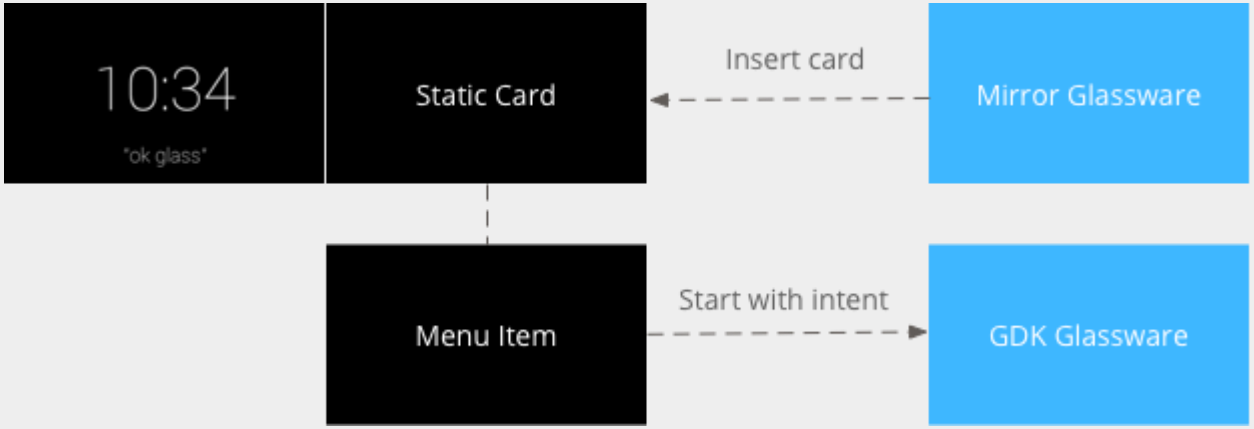
- Addon for Android SDK containing APIs for Glass-specific features
- Glassware coded with GDK runs on Glass directly
- Access to **hardware level features**
- Creation of Android standard packages (APK)
- Real-time user interaction
- Offline functionalities

Mirror APIs features

- Web-based services available in multiple languages
- Code is not running on Glass
- Possibility to interact with user's timeline

We can use them both!

- Mirror APIs can invoke GDK through a `Intent`



This method can merge a web experience with a full one

GDK – APIs overview

- More or less like usual **Android programming**
- Set of APIs that will let us interface with **lower level features**
- **Glass library** required to install Glassware
- Our apps will most probably **install and run on any Android device**, but many crashes will happen when trying to access Glass APIs

GDK – Environment setup

- Be sure you have both **Android 4.0.3 (API 15) SDK** and **Glass Development Kit Sneak Peek** installed
- Create a new project with:
  - Minimum and Target SDK Version: **15** (Glass runs on Android 4.0.3 only)
  - Compile with: **Glass Development Kit Sneak Peek**
  - Theme: **none** (So default Glass theme is applied)

GDK – Sample Project

- Project sources can be found on GitHub at **bit.ly/OKGlass-Github**
- We need Android **Support Library v4**
- And a **Google Glass**!

GDK - AndroidManifest

- Our application requires **com.google.android.glass** library
- **Intent-filter** is set on **VOICE_TRIGGER** action
- **Meta-data** will listen for our command

```
<uses-sdk    android:minSdkVersion="15"
             android:targetSdkVersion="15" />

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

<uses-library android:name="com.google.android.glass"
              android:required="true" />

<activity  android:name=".MainActivity" android:theme="@style/MenuTheme" >
             <intent-filter>
                         <action android:name="com.google.android.glass.action.VOICE_TRIGGER" />
             </intent-filter>
             <meta-data android:name="com.google.android.glass.VoiceTrigger"
                         android:resource="@xml/hello_show" />
</activity>
</application>
```

GDK – Let's start

- We create a **Menu** as we would with any Android app and set our actions
- We set a **trigger** element which contains the **keyword** that will make our software run

```
<trigger keyword="@string/show_helloworld_voice_trigger"/>
```

GDK - TextToSpeech

- Reference **goo.gl/GjE9Tb**

- Initialize **TextToSpeech** with **constructor** and **OnInitListener** in **onCreate** method

```
textToSpeech = new TextToSpeech(this,
      new TextToSpeech.OnInitListener() {
      @Override
      public void onInit(int i) { /* do nothing*/ } }
);
```

- Shut down **TextToSpeech** object and set it to **null** in custom method called by **onDestroy**

```
textToSpeech.shutdown();
textToSpeech = null;
```

- Use it through **speak** method when requested by user

```
textToSpeech.speak(getString(R.string.read_aloud),
            TextToSpeech.QUEUE_FLUSH,
            null);
```

GDK – Card and Timeline

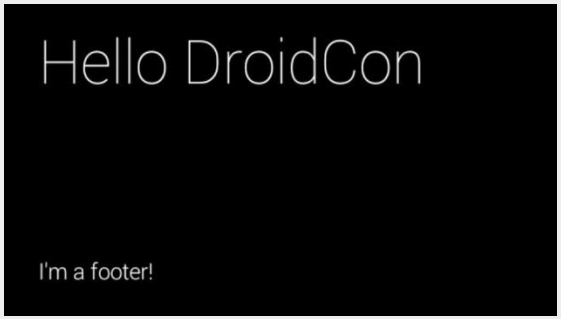
- First of all we shall instanciate a **TimelineManager** object (**62**)
- We then create a new **Card** (**64**)
- We set **Text** and **Footer** on jest created **Card** (**65-66**)
- Time to add the **Card** to our **Timeline**! (**68**)

```
61 private void addCard(){
62         TimelineManager timelineManager = TimelineManager.from(this);
63
64         Card mCard = new Card(this);
65         mCard.setText("Hello DroidCon");
66         mCard.setFootnote("I'm a footer!");
67
68         timelineManager.insert(mCard);
69 }
```

GDK – Let's see it on Glass

This will be shown on apps list

This will appear on history part of Timeline



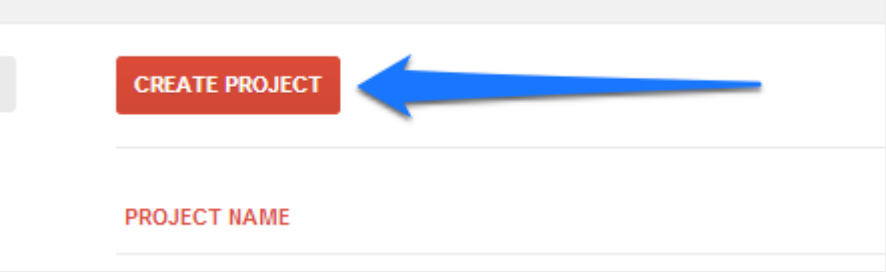
And this is the Card we created through code

Mirror APIs - Overview

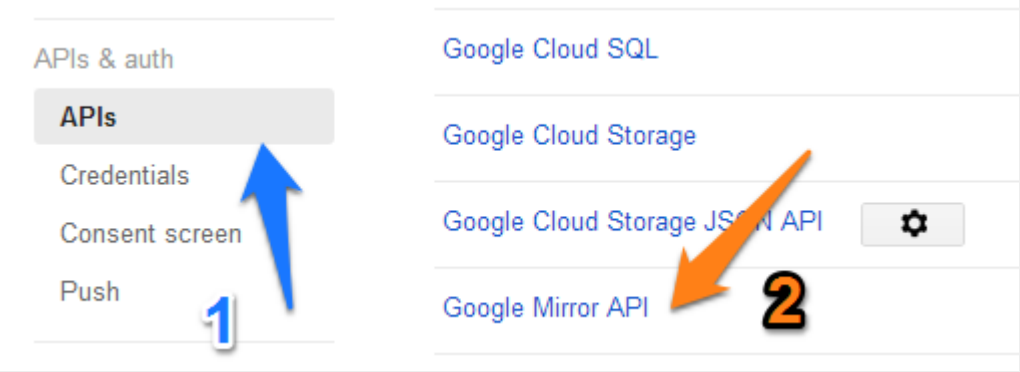
- Delivering content to users: **Mirror APIs can send content to users' timelines**
- **Adding contacts** to users' Glass
- **Getting users' location** and show nearest locations of interest
- Available in Go, Java, .NET, PHP, Python and Ruby

Mirror APIs – Enabling access

- Create a **new API Project** on Google Developers Console

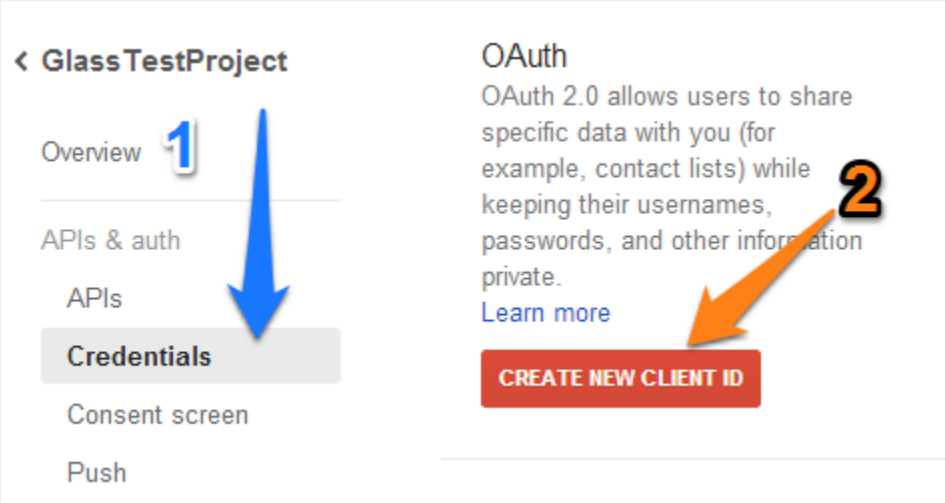


- Click on **APIs** (under *APIs & Auth*) and enable **Google Mirror API**



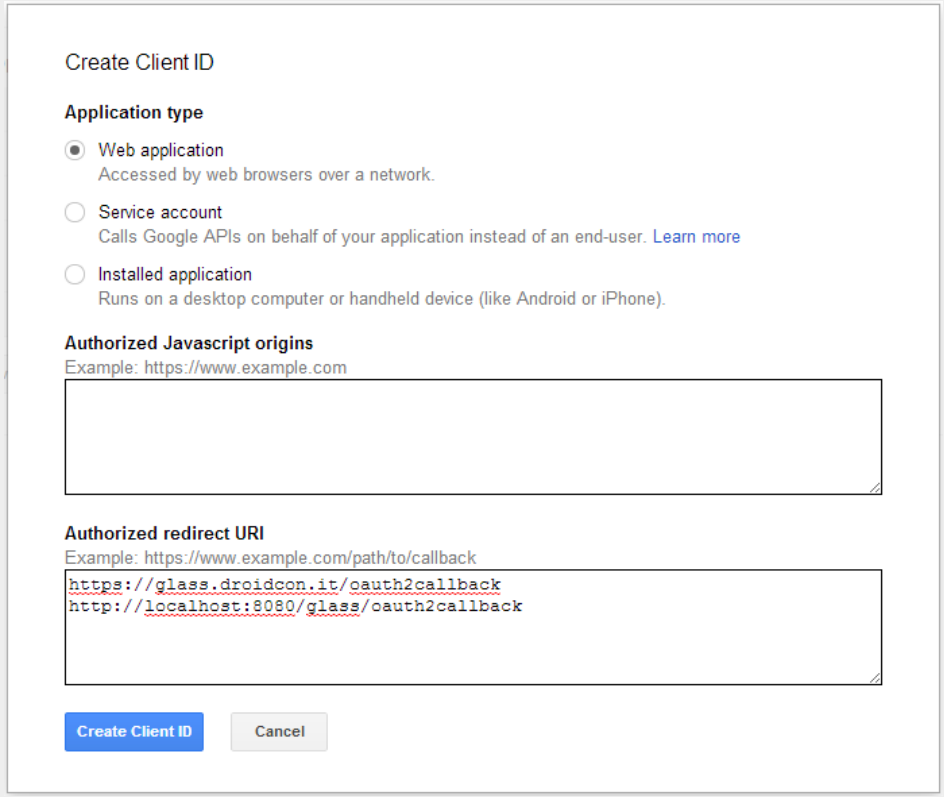
Mirror APIs – Enabling access

- Click on **Credentials** and then pick **Create new Client ID**



Mirror APIs – Enabling access

- Select the **Web application** radio button and paste in **Authorized redirect URI** callbacks for your development <u>and</u> deployment servers



Mirror APIs – Enabling access

- Finally, click on **Create Client ID** and write down resulting data

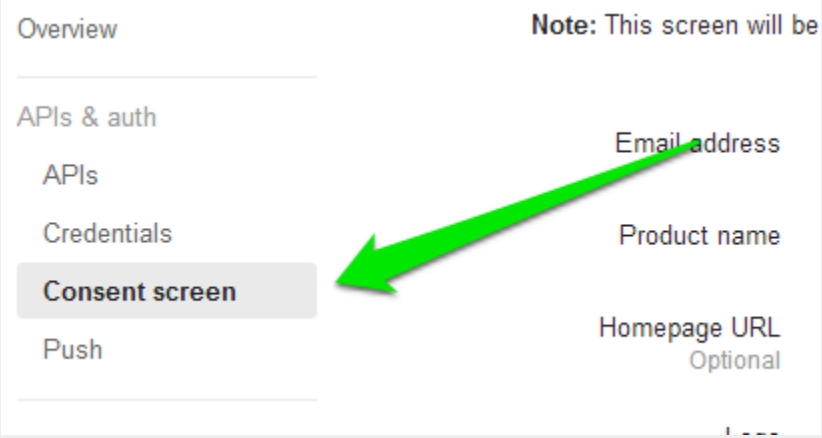
Client ID for web application

Client ID	6207185070-b4dt4vaqq748i9lc9hth3cm4ncn8csjj.apps.googleusercontent.com
Email address	6207185070-b4dt4vaqq748i9lc9hth3cm4ncn8csjj@developer.gserviceaccount.com
Client secret	YGrpdNPL6hAoLhClvlP7pMS7
Redirect URIs	http://localhost:8080/glass/oauth2callback https://www.droidcon.com/example/oauth2callback
Javascript Origins	none

Edit settings | Download JSON | Delete

Mirror APIs – Consent screen

- This window will appear **every time** we request access with our *Client ID*
- It will contain our **project name**, **URL and logo**
- We can customize it by clicking on **Consent screen** under *APIs & auth menu*



Mirror APIs – Consent screen

Consent screen

The consent screen will be shown to users whenever you request access to their private data using your client ID.
Note: This screen will be shown for all of your applications registered in this project

Email address: roberto.orgiu@gmail.com

Product name: Glass Test

Homepage URL (Optional): http://www.droidcon.it/glass-test/

Logo (Optional): http://imagizer.imageshack.us/v2/800x600q9

This is how your logo will look to end users.
Max size: 120x120 px

Privacy policy URL (Optional): http://www.droidcon.it/glass-test/privacy.htm

Terms of service URL (Optional): http://www.droidcon.it/glass-test/tos.html

Google+ Page (Optional): plus.google.com/ Page ID

Save Cancel

Logo

Product Name

Developer info

email:

This app would like to:

Know your name, basic info, and list of people you're connected to on Google+

Make your app activity and reviews available via Google, visible to you and:

Your circles + Add more people

Only you

Product Name and Google will use this information in accordance with their respective terms of service and privacy policies.

Cancel Accept

Mirror APIs – Configuring sample project

- Download PHP quickstart sample from **GitHub** and unzip in your workspace
- Open **config.php** and set our variables
  - **application_name** will appear in *User-Agent HTTP header*
  - **oath2_client_id** and **oath2_client_secret** is what we saved a few minutes ago
  - **oath2_redirect_uri** is, for development, localhost address we specified during *Client ID creation*
  - **developer_key** is our signature

Mirror APIs - Configuring sample project

- In our case, we would have something like this

```
global $apiConfig;
$apiConfig = array(
// True if objects should be returned by the service classes.// False if associative arrays should
be returned (default behavior).
'use_objects' => false,

// The application_name is included in the User-Agent HTTP header.'application_name' => '',

// OAuth2 Settings, you can get these keys at https://code.google.com/apis/console
'oauth2_client_id' => '6207185070-b4dt4vaqq748i9lc9hth3cm4ncn8csjj.apps.googleusercontent.com',
'oauth2_client_secret' => 'YGrpdNPL6hAoLhClvlP7pMS7',
'oauth2_redirect_uri' => 'http://localhost:8080/glass/oauth2callback',
```

Where these are all the values we got from the previous procedure.

We are now ready to deploy our test project!

Thanks for watching!

You can find these slides at… bit.ly/OKGlass-Github

… reach me out at… roberto.orgiu@gmail.com

… or just by scanning this QR Code