Text to Speech Engine

© 2008 Haim Michael

Introduction

- As of Android 1.6 (and above) the Android platform includes a multilingual speech synthesis engine, also known as Pico.
- Using this engine it is possible for our application to speak a string of text with an accent that matches the language.
- Using this engine, our application can interact with the users without having them looking back to the screen.

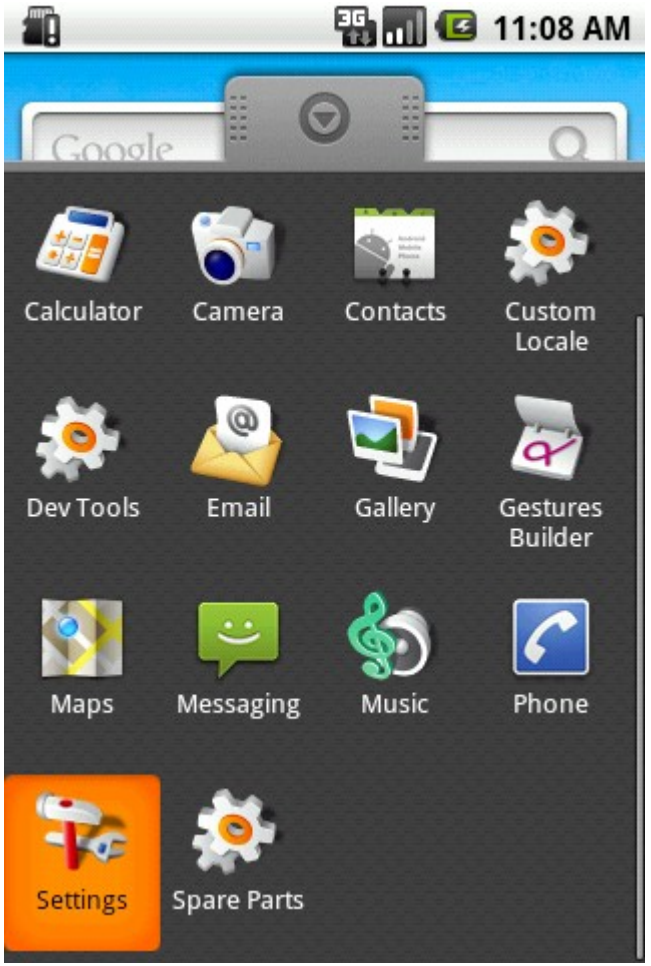
© 2008 Haim Michael

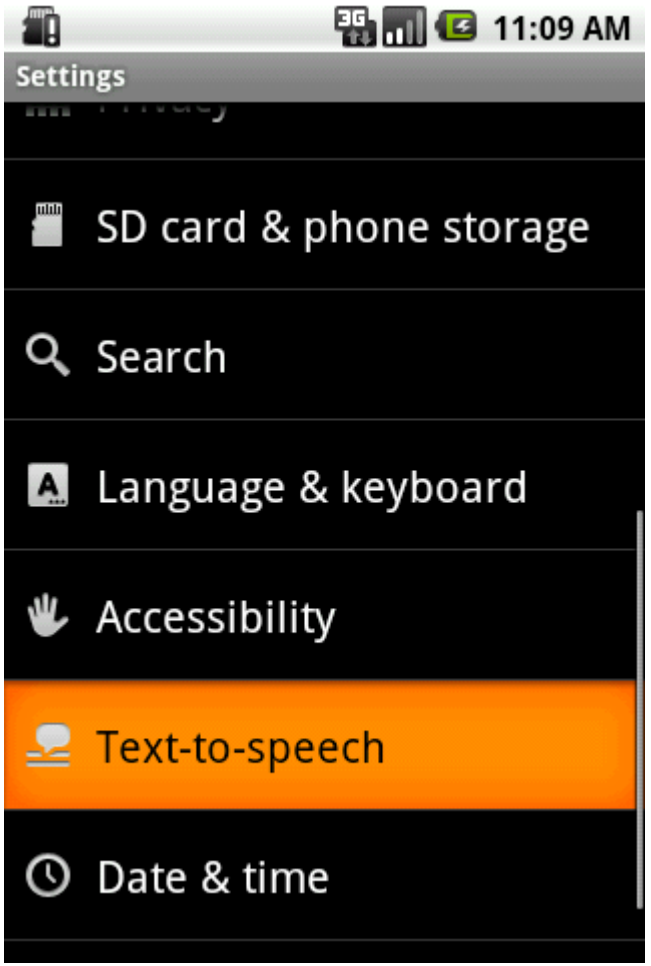
Text to Speech Demo

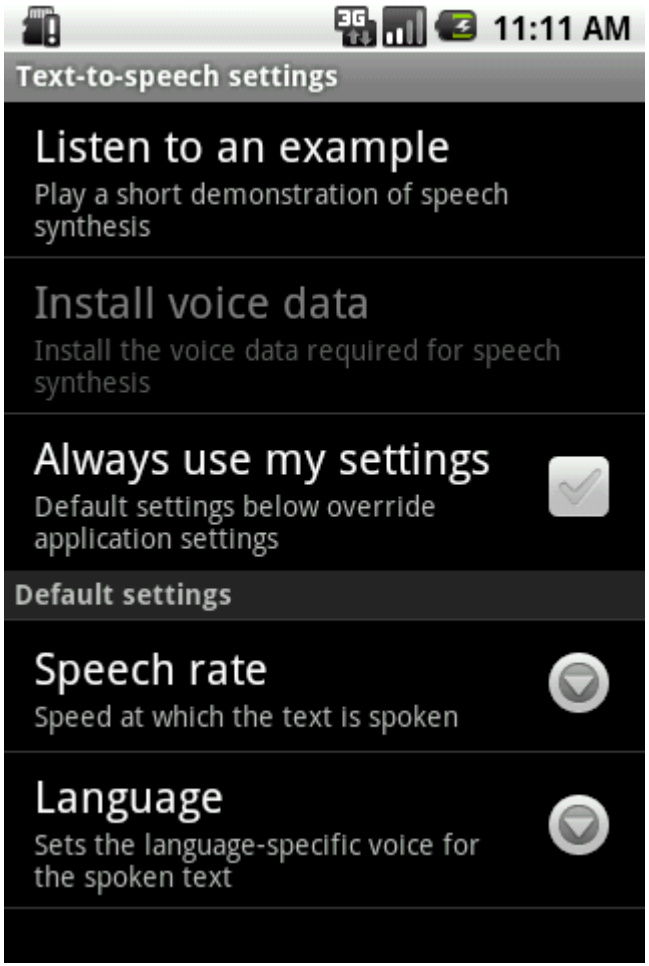
- On your android emulator desktop you can find the Setting icon. Tap that icon and get the Setting screen.
- On Setting screen you will find the Text-to-Speech option. Tap that option and get the Text-to-Speech screen.
- On top of the Text-to-Speech screen you will find the Listen to an example option. Choose that option and hear the android text to speech engine in action.

© 2008 Haim Michael

Text to Speech Demo







© 2008 Haim Michael

Single TTS Engine

- The Android platform has one TTS engine only. That engine is shared across all activities. Therefore, we can never be sure that our text will be indeed spoken.
- The Android SDK includes an interface for the TTS engine.

© 2008 Haim Michael

Code Sample

```
public class TTSEngineActivity extends Activity implements OnInitListener
{
    private EditText textToSpeachEditText = null;
    private Button talkButton = null;
    private static final int STATUS_CHECK = 0;
    private TextToSpeech engine;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        textToSpeachEditText = (EditText)findViewById(R.id.text_to_speech);
        talkButton = (Button) findViewById(R.id.talk);
        talkButton.setOnClickListener(new OnClickListener()
        {
            @Override
            public void onClick(View view)
            {
                engine.speak(    textToSpeachEditText.getText().toString(),
                                 TextToSpeech.QUEUE_ADD, null);
            }
        });
        Intent checkIntent = new Intent();
        checkIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);
        startActivityForResult(checkIntent, STATUS_CHECK);
    }
```

Passing over this value our text will be queued after other texts that are already waiting for be heard.

© 2008 Haim Michael

Code Sample

```
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
    if (requestCode == STATUS_CHECK)
    {
        switch (resultCode)
        {
            case TextToSpeech.Engine.CHECK_VOICE_DATA_PASS:
                engine = new TextToSpeech(this, this);
                break;
            case TextToSpeech.Engine.CHECK_VOICE_DATA_BAD_DATA:
            case TextToSpeech.Engine.CHECK_VOICE_DATA_MISSING_DATA:
            case TextToSpeech.Engine.CHECK_VOICE_DATA_MISSING_VOLUME:
                Intent installIntent = new Intent();
                installIntent.setAction(
                    TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA);
                startActivity(installIntent);
                break;
            case TextToSpeech.Engine.CHECK_VOICE_DATA_FAIL:
                //...
        }
    }
    else { //... }
}
```

Getting back this value means that everything is OK and we can proceed with instantiating the engine.

The second argument should be of the OnInitListener type. When the engine is ready the onInit method will be called.

© 2008 Haim Michael

Code Sample

```
    @Override
    public void onPause()
    {
        super.onPause();
        if (engine != null)
        {
            engine.stop();
        }
    }

    @Override
    public void onDestroy()
    {
        super.onDestroy();
        engine.shutdown();
    }

    @Override
    public void onInit(int status)
    {
        //...
    }
}
```

We better stop the engine when the activity is paused.

We better shutdown the engine when our activity ends its life and free resources the engine was using.

We can use onInit for enabling the 'Talk' button when been notified the engine is ready for use.

© 2008 Haim Michael

Code Sample

© 2008 Haim Michael

Text to Speech Engine

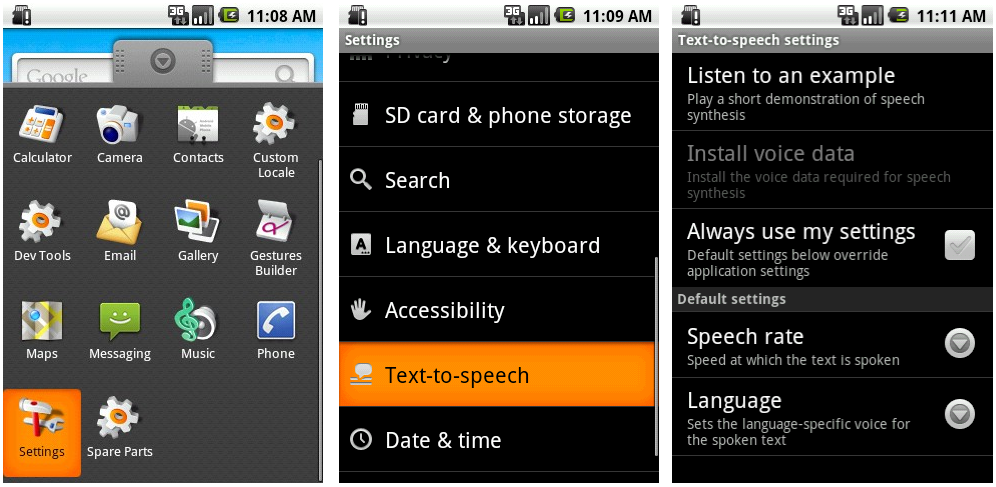
Introduction

- As of Android 1.6 (and above) the Android platform includes a multilingual speech synthesis engine, also known as Pico.
- Using this engine it is possible for our application to speak a string of text with an accent that matches the language.
- Using this engine, our application can interact with the users without having them looking back to the screen.

Text to Speech Demo

- On your android emulator desktop you can find the Setting icon. Tap that icon and get the Setting screen.
- On Setting screen you will find the Text-to-Speech option. Tap that option and get the Text-to-Speech screen.
- On top of the Text-to-Speech screen you will find the Listen to an example option. Choose that option and hear the android text to speech engine in action.

Text to Speech Demo



Single TTS Engine

- The Android platform has one TTS engine only. That engine is shared across all activities. Therefore, we can never be sure that our text will be indeed spoken.
- The Android SDK includes an interface for the TTS engine.

Code Sample

```
public class TTSEngineActivity extends Activity implements OnInitListener
{
    private EditText textToSpeachEditText = null;
    private Button talkButton = null;
    private static final int STATUS_CHECK = 0;
    private TextToSpeech engine;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        textToSpeachEditText = (EditText)findViewById(R.id.text_to_speech);
        talkButton = (Button) findViewById(R.id.talk);
        talkButton.setOnClickListener(new OnClickListener()
        {
            @Override
            public void onClick(View view)
            {
                engine.speak(    textToSpeachEditText.getText().toString(),
                                 TextToSpeech.QUEUE_ADD, null);
            }
        });
        Intent checkIntent = new Intent();
        checkIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);
        startActivityForResult(checkIntent, STATUS_CHECK);
    }
```

Passing over this value our text will be queued after other texts that are already waiting for be heard.

Code Sample

```
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
    if (requestCode == STATUS_CHECK)
    {
        switch (resultCode)
        {
            case TextToSpeech.Engine.CHECK_VOICE_DATA_PASS:
                engine = new TextToSpeech(this, this);
                break;
            case TextToSpeech.Engine.CHECK_VOICE_DATA_BAD_DATA:
            case TextToSpeech.Engine.CHECK_VOICE_DATA_MISSING_DATA:
            case TextToSpeech.Engine.CHECK_VOICE_DATA_MISSING_VOLUME:
                Intent installIntent = new Intent();
                installIntent.setAction(
                    TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA);
                startActivity(installIntent);
                break;
            case TextToSpeech.Engine.CHECK_VOICE_DATA_FAIL:
                //...
        }
    }
    else { //... }
}
```

Getting back this value means that everything is OK and we can proceed with instantiating the engine.

The second argument should be of the OnInitListener type. When the engine is ready the onInit method will be called.

Code Sample

```
    @Override
    public void onPause()
    {
        super.onPause();
        if (engine != null)
        {
            engine.stop();
        }
    }

    @Override
    public void onDestroy()
    {
        super.onDestroy();
        engine.shutdown();
    }

    @Override
    public void onInit(int status)
    {
        //...
    }
}
```

We better stop the engine when the activity is paused.

We better shutdown the engine when our activity ends its life and free resources the engine was using.

We can use onInit for enabling the 'Talk' button when been notified the engine is ready for use.

Code Sample