Introduction to RGA

One major limitation of the Open C/C++ offering on the S60 platform is that it does not offer developers the possibility to create an User Interface for their programs or to access platform specific features (contacts application & messaging, camera, video and audio playback, etc.)

A partial solution for these limitation is provided through the RGA library (RGA stands for Real-time Graphics and Audio). It offers Open C/C++ like APIs as wrappers for S60 platform's native APIs simplifying thus the access to the supported features from Open C code, especially for developers which are not familiar with Symbian C++ and would thus find it difficult to manage mixed Open C and Symbian C++ code.

RGA enables developers to write feature rich 2D graphical applications by providing access to Device Screen, also allows users to manipulate bitmap images. It provides API's to playback audio and video streams. RGA provides API's to write display animated content and also provides API's to draw text.

The RGA is in fact a subset of the N-Gage SDK and this recommends it to games developers. Of course, besides games, also many other types of multimedia rich applications can also be developed based on these APIs.

The set of features accessible through the RGA APIs includes:

- 2D graphics & text
- Sound Effects and music
- Video Playback
- Input Devices
- Timers
- Cameras
- Lights
- Device Status and Capabilities
- Calendar Alerts/Todo
- Messaging

See the complete list of APIs below.

The RGA APIs

- **High Level Audio**

The High Level Audio API uses the Runtime API to instantiate the general audio management objects.

- **CompressedAudio**

The Compressed Audio API provides low-level configurable audio playback and recording, mixing functionality, playback of MIDI and compressed audio files such as MP3, audio stream playback and audio clips playback .

- **Alerts**

The Alerts API provides interfaces for managing alerts set by the game to the user.

- **Fonts API**

The IFontFactory interface class provides APIs to get the installed system font count and system font information, as well as install system fonts, uninstall system fonts, creating system font objects, and create bitmap font objects.

- **Backbuffer**

In order to avoid tearing effects and provide device independence the Back Buffer API provides a double buffered graphics scheme consisting of a screen buffer and a secondary buffer called the back buffer, which developers can access and to which they can draw.

- **Bitmap**

The purpose of the Bitmap API component is to support the developer with a variety of bitmap manipulation functions.

- **Camera**

The purpose of the Camera API component is to support the developer with interfaces to access the phone’s camera devices.

- **Display**

The Display API provides the developer with system-independent window handles for all of the phone’s displays.

- **Lights**

The Lights API is a library which provides interfaces to set the Light State, such as light on, light off, and light blink.

- **VideoPlayback**

The Video Playback API component of the Nokia Gaming APIs provides necessary functionality for video playback.

- **Input API**

The Input API provides an interface to retrieve general information on devices connected to the phone and interact with them.

- **Text Input**

The Text Input API is a library which provides interfaces for text input.

- **Keypad Capabilities API**

The KeypadCapabilities API provides an interface to retrieve general information about the default keypad device connected to the phone.

- **DeviceCapabilites**

The Device Capabilities API allows the developer to access and retrieve information from a Series 60 RGA capable phone.

- **Device Status**

The Device Status API is a library which provides interfaces to retrieve Device Status and also Device Events like telephony events, network events, battery events, accessory events, profile events and alarm events.

- **Runtime and Application API**

The Runtime API, including the CRuntime and IIdle interface classes, offers functionality related to creating and instantiating APIs and handling background tasks. The Application State API offers functionality related to the application's state. All this is necessary when creating the basic framework for a game.

- **Themes API**

The Themes API provides an interface to change themes. Timing The Timing API provides necessary functionality for a Periodic Timer and a One Shot Timer.

- **Vibra API**

The Vibra API provides access to vibra motor.

- **Virtual Code API**

The Virtual Code API provides the Alloc interface to allocate a region of memory for copying code.

Limited availability

To be noted that not all S60 phones that support Open C/C++ would also support the RGA library thus the applications created using this library will have to target a limited set of phones. RGA was introduced in Open C/C++ 1.3 (see the release history) but it is no longer available since Open C/C++ 1.6. For UI development using non-Symbian APIs, developers may wish to explore Qt for S60

As with the rest of the Open C/C++ libraries, developers will have to ensure that the applications designed using the RGA APIs can only be installed on the supported devices and that the required binary files are also delivered with the application (through an embedded SIS).

Get the plug-in

- Open C/C++ Plug-ins for S60 3rd Edition

Nokia Developer Examples

- RGA: Tetromino Game Example
- RGA: Biowaste Game Example v1.0 Beta

Related wiki articles

- Simple RGA application
- Using RGA with Carbide.c++