A portable java API interface for user access to digital still cameras

A Portable Java API Interface for User Access to Digital Still Cameras.

Peter Corcoran, Ferenc Papai, Arpad Zoldi and Joe Desbonnet
Dept. of Electronic Engineering, University College Galway, Ireland

Abstract - *The digital camera is one of the main successes of recent years in consumer electronics. However a typical digital camera remains tied to proprietary software on a personal computer. In this paper we describe the design and implementation of a portable Java API to simplify end user access to digital cameras and to provide interconnectivity with a new generation of intelligent home applicances.*

Introduction

The digital camera has been one of the major success stories in new consumer electronic products during the past few years. Indeed there has been such explosive growth that there are now more than 20 manufacturers of competing products in this particular market niche. However, although many digital cameras are functionally similar each camera is accessed via its own proprietary software and protocols on the users personal computer.

In this paper we consider some of the issues involved in the design of a general Java API for digital cameras. We also consider how a new generation of intelligent home appliances would be capable of providing access to and functional integration with such a generic camera API.

System Architecture & Overview

As a practical implementation of our ideas we have studied several of the most popular low end digital still cameras and derived from these a consensus of the main features, services and user-interface components required in a generic digital camera API.

In addition we examine how suitable, lightweight and portable user-interfaces and protocol-interpreter engines can be implemented using some of today's key emerging technology standards. We place particular emphasis on modularizing components of the API. Thus the core software building-blocks, or "personality" which control a camera, and provide access to its inbuilt functionality and stored data may be loaded and unloaded dynamically as required. The end goal is to allow camera "personalities" to be loaded remotely by a PC or Internet Appliance (IA) from a wide area network such as the Internet, or even directly from the camera itself.

As an inital proof-of-concept we have implemented a prototype based on a standard desktop PC, to demonstrate how our portable API allows a number of different digital cameras, each from a different manufacturer, to be accessed from a common user-interface module. In each case the core elements of the camera personality are downloaded from a remote URL (universal resource locator) on a TCP/IP network and the user-interfaces and application modules may also be loaded remotely in the same way.

In a practical consumer application the PC might equally be replaced by an intelligent screen phone, set-top box or IA. Thus an end user, by connecting their digital camera to, say, a screen phone could automatically load from the Internet -

(i) a protocol interpreter to access the camera,

(ii) a user interface to download, edit and view pictures, and

(iii) additional application modules to archive, manage and index pictures either locally, or on a remote site and to transmit selected pictures via email.

What is important is that all of these functional modules are only loaded as required by the end-user and the exact modules loaded are determined from the behaviour of the camera connected to the serial port.

The detection of the camera and the initial loading of camera personality, user-interface and applications modules can all be managed by a very small and lightweight *daemon*-style program. This, in turn, makes the approach described in the paper very suitable for the new generation intelligent home devices.

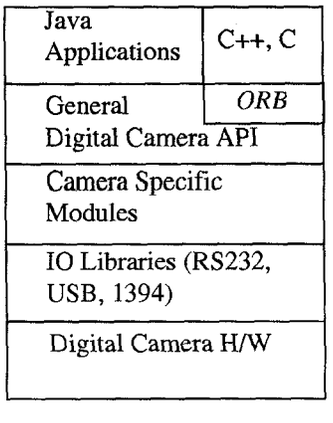

Fig 1:. Overview of the API Software Architecture

0-7803-4357-3/98/ $10.00 © 1998 IEEE

Generic Digital Camera API

The Digital Camera classes provide application software with a consistant abstraction of all digital cameras. Application software can use a camera without specific knowledge of what resolutions, encoding standards, shutter speeds, and focusing mechanisms are available.

In order to allow existing applications written in languages such as C and C++ to use this proposed software API infrastructure we further propose that a CORBA ORB should be incorporated into the higher API layers. We also envisage that the inclusion of an ORB will simplify the development of network aware applications and distributed applications for home and business use.

The camera specific layer implements all the necessary code to communicate with a specific digital still camera. This layer is designed so as to allow new camera personalities to be added.

The IO layer is a platform specific library to enable communication via RS232 (in our prototype) and ultimately via Universal Serial Bus or 1394 serial connections when suitable cameras become available.

Conclusions

The Java API interface was designed to provide a generic interface to communicate with a wide range of digital still cameras from a common user interface and common software API.

As a proof-of-concept we have developed software that demonstrates all aspects of the API architecture discussed in this paper and which works with several of the commonest digital still cameras available from leading manufacturers today.

Fig 2: Example of a simple user interface & basic application software to download and manage the pictures from a digital still camera; this has been programmed using the generic API and camera classes described in this paper.