Technical Disclosure Commons

Defensive Publications Series

June 2020

COMPATIBILITY TESTING FOR CAMERA APPLICATIONS TO FACILITATE SOFTWARE DEVELOPMENT

Clemenz Portmann

David Chen

Leslie Shaw

Vinit Modi

James Fung

COMPATIBILITY TESTING FOR CAMERA APPLICATIONS TO FACILITATE SOFTWARE DEVELOPMENT

ABSTRACT

A system is described that provides a consistent application programming interface and enables compatibility testing for applications that interface with cameras across various mobile devices. The system includes one or more controlled testing environment modules that isolate mobile devices from ambient light and provide test charts and consistent internal lighting for camera testing. The controlled testing environment modules enable operating system developers to capture both landscape and portrait images, access features, and test applications in consistent testing environments across various mobile devices. Such testing may enable development of software shims to facilitate interaction with a wide variety of cameras across different vendors and implementations within individual mobile devices.

DESCRIPTION

Application developers may develop applications that add effects to photos taken by mobile devices. With dozens of manufacturers developing mobile devices, there is great variation between camera implementations across these mobile devices from different manufacturers, which presents unique challenges associated with developing applications that interface with various camera modules. For example, different camera modules may require slightly different programming to get the same result. In order to achieve the same result across various camera modules, application developers may need to develop and release applications in different versions for different mobile devices, which can be time consuming and expensive.

Thus, it is desirable to develop software shims that enable a single function call to get the same result regardless of the underlying camera modules.

FIG. 1 is a conceptual diagram illustrating an example system configured to provide a consistent application programming interface (API) and enables compatibility testing for camera functionalities across various mobile devices. As shown in FIG. 1, system 100 includes companion computing device 102, network 104, computing device 106, and testing modules 108A–108N.

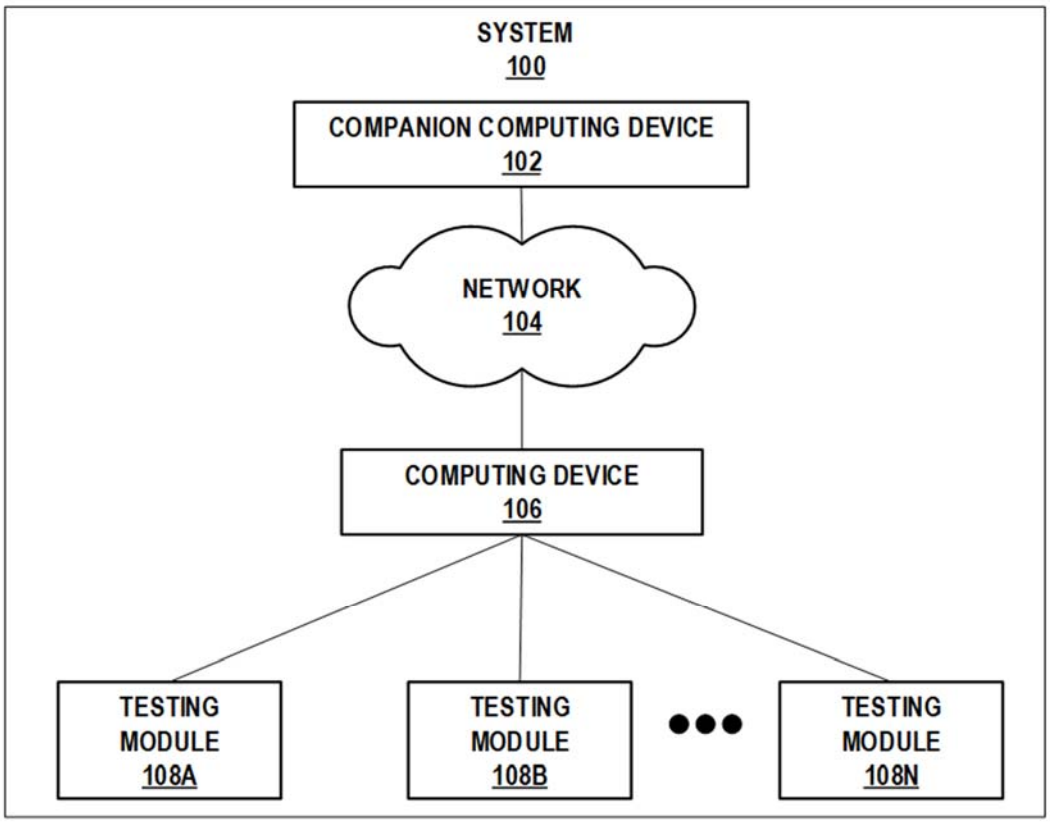

FIG. 1

In the example of FIG. 1, companion computing device 102 represents an individual mobile or non-mobile computing device that a developer may use to develop a camera application. Examples of companion computing device 102 include a mobile phone (including a

so-called smartphone), a tablet computer, a laptop computer, a desktop computer, or any other suitable mobile or non-mobile device capable of interfacing with computing device 106 via network 104. Computing device 106 represents one or more computing devices that are able to control testing modules 108A–108N. Examples of computing device 106 include a tablet computer, a laptop computer, a desktop computer, a server, a mainframe, a microcontroller, or any other suitable computing device capable of interfacing with testing modules 108A-108N and/or companion computing device 102 via network 104.

Companion computing device 102 may connect to computing device 106 via network 104. For example, companion computing device 102 may send control information to computing device 106 via network 104 to control testing modules 108A–108N. Network 104 represents a combination of any one or more public or private communication networks, for instance, television broadcast networks, cable or satellite networks, cellular networks, Wi-Fi networks, broadband networks, local area networks (LANs), and/or other type of network for transmitting data (e.g., telecommunications and/or media data) between various computing devices, systems, and other communications and media equipment.

Although described as interfacing with a single computing device 106, companion computing device 102 may interface with two or more computing devices 106, each of which may control some subset of control testing modules 108A-108N. In these instances, companion computing device 102 may transmit control information to each of computing devices 106 either concurrently or in series. In some examples, companion computing device 102 may broadcast commands to each of computing devices 106.

Computing device 106 may connect to testing modules 108A–108N via a network. Responsive to the control information received from companion computing device 102,

computing device 106 may control testing modules 108A–108N to perform certain actions, such as taking photos or controlling portrait/landscape rotators of testing modules 108A–108N.

Testing modules 108A–108N isolate various mobile devices from ambient lights and provide test charts and consistent internal lighting for camera testing. Each testing module of testing modules 108A–108N holds two mobile devices of the same model, which enables testing of a front camera and a back camera concurrently. It should be understood, however, that each testing module of testing modules 108A–108N may hold more or less than two mobile devices of the same model, so that more or less than two cameras may be tested concurrently. In some examples, testing modules 108A–108N may be used to test alternative sensors in the mobile devices, such as, but not limited to, accelerometers, gyroscopes, magnetometers, heart rate sensors, proximity sensors, pressure sensors, relative humidity sensors, and any other suitable sensors in the mobile devices.

FIG. 2 below is a conceptual diagram illustrating an example testing module 200. As shown in FIG. 2, the testing module includes ambient light isolated housing 202, sliding door 204 for phone access, mobile device fixture 206, rotator 208 connected with mobile device fixture 206, internal lights 210A and 210B (collectively "internal lights 210"), controller 212 for controlling rotator 208 and internal lights 210, test chart 214 (which may also be referred to as target scene), and two mobile devices of the same model attached to mobile device fixture 206.

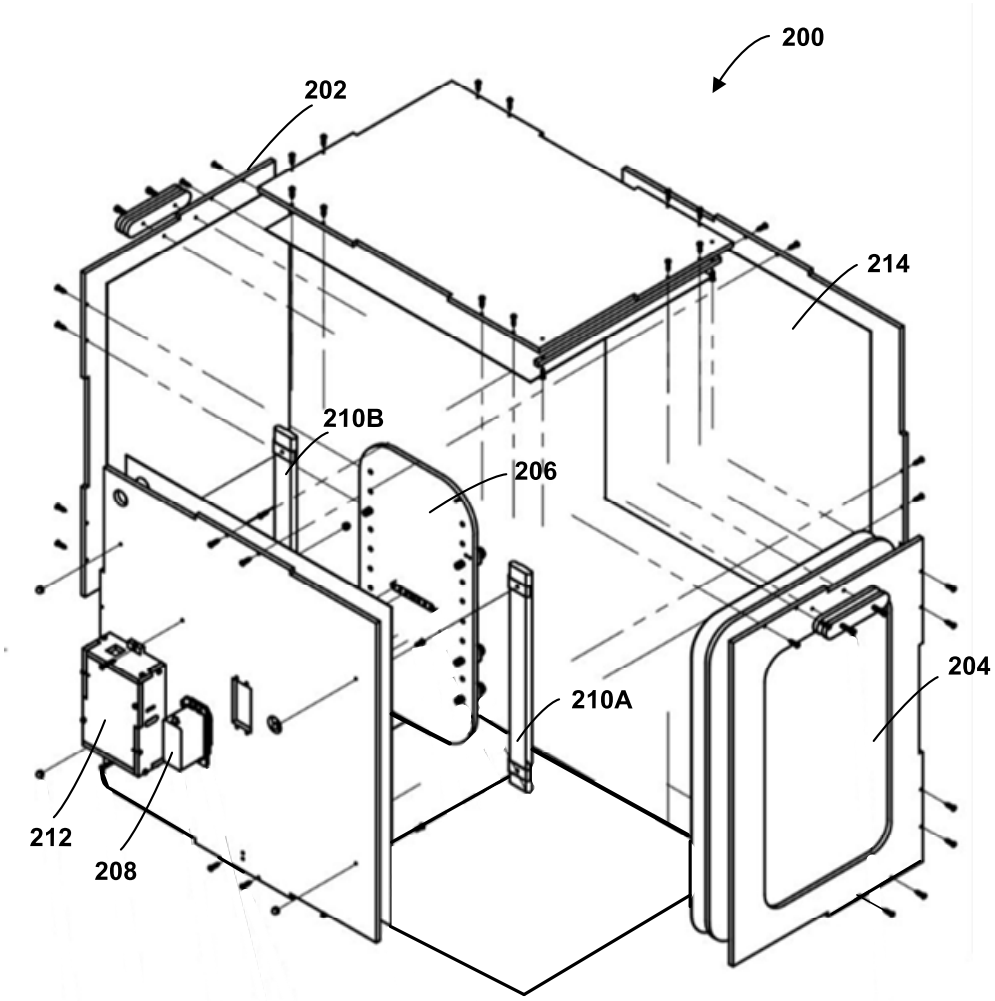

FIG. 2

In the example of FIG. 2, ambient light isolated housing 202 may be constructed of opaque Acrylonitrile-Butadiene-Stryene (ABS) plastic sheets which are suitable for applying wallpapers, silkscreens, or paint that are needed to control the internal lighting environment of testing module 200. Inner walls of ambient light isolated housing 202 may be covered with gray matte wallpaper to provide consistent internal lighting levels and reduce reflections. In some examples, ambient light isolated housing 202 may include sliding door 204, which provides easy accessibility to mobile devices.

Internal lights 210 may include two lighting bars fixed to an inner wall of ambient light isolated housing 202. Controller 212 may control internal lights 210 to provide adequate lighting for camera testing. For example, responsive to a user request, computing device 106 may send control information to controller 212 to raise or lower the brightness level of the lighting bars.

Test chart 214 may be attached to the inner wall of ambient light isolated housing 202 and may be used to test camera performance. In some examples, test chart 202 may include: QR code test chart for testing camera functions and image orientation, slant-edge target test chart for testing sharpness and lens performance, scale-invariant test chart for testing effects of signal processing on image texture, noise chart for measuring dynamic range and noise of the back camera, resolution test chart for measuring camera resolution, portraits for testing face detection, and color chart for testing color correctness. Mobile device fixture 206 may be attached to the inner wall of ambient light isolated housing 202 and may be connected to rotator 208, which enables capture of both landscape and portrait images.

FIG. 3 below is a conceptual diagram illustrating an example for mobile device fixture 300. As shown in FIG. 3, mobile device fixture 300 includes central hub 302 that is connected to rotator 208, and two mobile devices 306A and 306B (collectively "mobile devices 306") of the same model attached to mobile device fixture 206. One of the mobile devices 306A is facing up and the other mobile device 306B is facing down, which enables an operating system developer to test the front and back camera at the same time. In some examples, mobile devices fixture 300 may include mounts 304 (also called clamps) that are configured to be screwed into tapped holes of mobile devices fixture 300 to fix mobile devices 306 in desired locations. In some examples, the inner side of mounts 304 are covered with adhesive rubber pads 308A–308C (collectively

"adhesive rubber pads 308") to prevent mobile devices 306 from moving and to reduce damage to the outer surface of mobile devices 306.

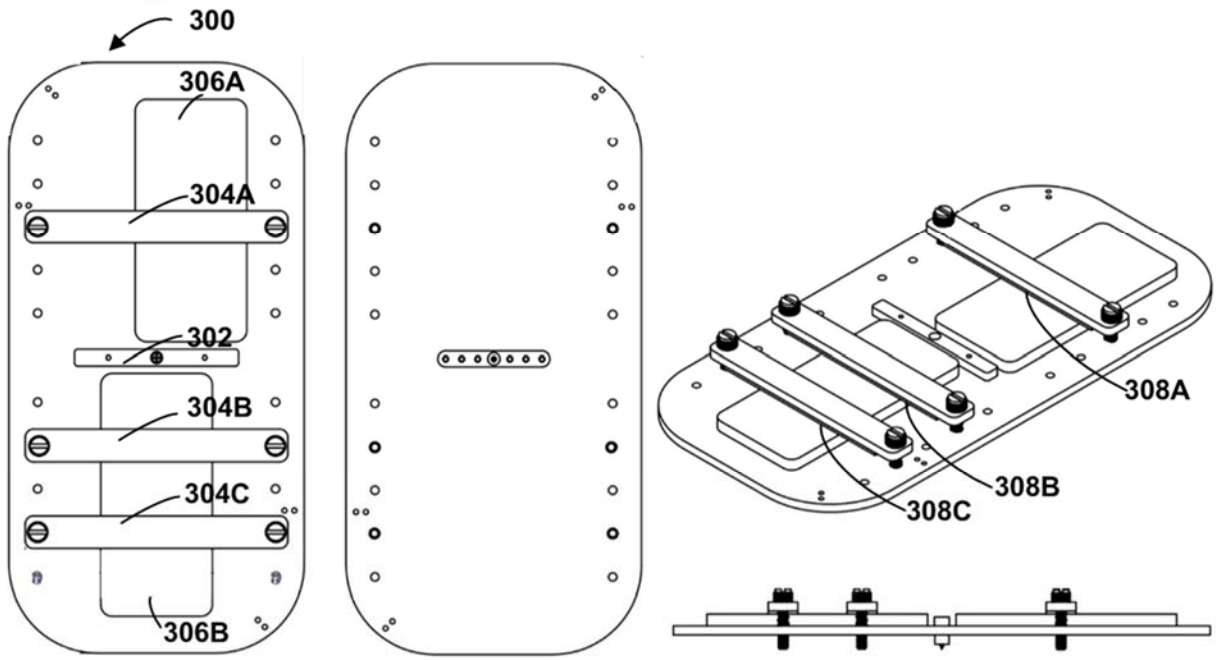

FIG. 3

Testing modules 108A–108N may be used to test a variety of camera behaviors across a range of devices, and test artifacts may be collected and used to develop software shims, allowing application developers to use one consistent function call to perform an operation for various devices. For example, an application developer (e.g., a user of companion computing device 102) may invoke a single camera function call to perform an operation (e.g., take a picture in a given mode) across various devices. For each device, the single camera function call invoked by the application developer may act as a software shim to invoke a device specific function call for the device, which comprises appropriate variables and settings for the device that are specific to the particular camera module utilized by the device (of a given model and brand). Shims developed based on the collected test artifacts may reduce the complexity that arises from incompatibilities between various camera modules among various mobile devices,

allowing application developers to use one consistent function call to perform an operation for various devices.

When the operating system developers run a test for a new camera module, a request may be sent from the companion computing device 102 to computing device 106. Computing device 106 may determine one or more devices in testing modules 108A-108N meeting the testing requirements and run a test for each device. For each test, computing device 106 may execute a sequence of simulated user action on the newly developed device. These actions could include user interface actions such as opening an app, pressing a take-photo button, or pressing a mode switch button.

As an example, the test may include a take-photo function call. Responsive to the function call, the software shims may accommodate the differences between various devices by changing actual parameters (e.g., reducing exposure time for overly sensitive cameras or extending exposure time for insensitive cameras) in the function calls to testing modules 108A–108N to accommodate differences between devices. The actions can also include physical actions such as rotating the phone to a different orientation. For example, the respective rotator in testing modules 108A–108N may rotate the respective mobile device fixture, which causes the respective mobile devices to rotate and enables capture of landscape photos and portrait photos. Computing device 106 may collect the test artifacts and the operating system developers may later query and inspect the test artifacts.

The operating system developers may utilize the test artifacts to develop the software shims such that there are uniform, or nearly uniform, results in the form of photo quality, exposure, balance, etc. between different camera modules and devices utilizing such camera modules when a given camera function is invoked. The operating system developers may update

parameters utilized by the shim responsive to the same function call invoked across devices. The shim, in other words, may be invoked by the camera function call exposed to application developers, where the shim may include a sequence of "if-then" and "else-if" statements (or a "switch" statement) that effectively switches between different camera modules and/or device models/manufacturer.

Responsive to invoking the shim, the shim may determine the camera module and/or device model/manufacturer of the device (executing the shim), and traverse the if-then/else-if statements until the determined camera module and/or device model/manufacturer is identified. The shim may then invoke a camera function call with the parameters determined through testing using the above noted system. In this manner, the shim may allow application developers to experience consistent results with the camera function call across camera modules and/or device models/manufacturers.

It is noted that the techniques of this disclosure may be combined with any other suitable technique or combination of techniques. As one example, the techniques of this disclosure may be combined with the techniques described in U.S. Patent Publication US 10,432,328 B2. As another example, the techniques of this disclosure may be combined with the techniques described in U.S. Patent Publication US 8,618,426 B2. As another example, the techniques of this disclosure may be combined with the techniques described in International Patent Publication WO 2015/063273 A1. As another example, the techniques of this disclosure may be combined with the techniques described in U.S. Patent Publication US 2018/0268378 A1.