Apéndice A

Desarrollo de la aplicación para el tablet NVIDIA Tegra

El prototipo tablet Tegra 3 de NVIDIA es uno de los dispositivos existentes compatibles con las librerías de código abierto de la FrankenCamera (Fcam). Estas librerías, escritas en lenguaje C++, permiten controlar a bajo nivel todas las características relacionadas con la cámara que incorpora el dispositivo móvil, siendo así la herramienta ideal para el desarrollo de la aplicación móvil necesaria en este proyecto. Dado que el tablet funciona bajo el sistema operativo Android, la incorporación de estas librerías no resulta trivial. Este anexo detalla la estructura de la aplicación móvil desarrollada, que sigue el esquema presentado en la figura A.1.

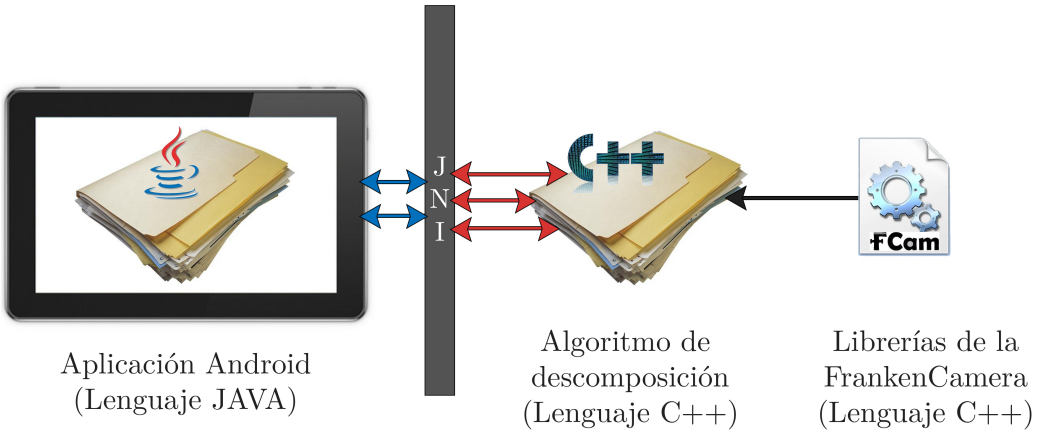

Figura A.1: *Diagrama de componentes correspondiente a la aplicación móvil desarrollada.*

A.1. Android

Android es un sistema operativo pensado para dispositivos móviles tales como smartphones o tablets. Su éxito sobre sus competidores (iOS, Symbian o Blackberry OS) radica en su núcleo de sistema basado en Linux, es decir, es libre, gratuito y multiplataforma. Como se observa en la figura A.2, la capa más baja del sistema es el núcleo Linux. Él es el encargado de interactuar con el hardware del dispositivo y de realizar las gestiones propias del sistema operativo (gestión de memoria, procesos y seguridad). En el segundo nivel más bajo se encuentran las librerías y la máquina virtual, también conocida como *Dalvik* (Android Runtime). A diferencia de la máquina virtual de Java (JVM), Dalvik esta basada en una máquina de registros y no en una máquina de pila. A continuación se sitúa la capa mediante la cual Android proporciona las herramientas necesarias para desarrollar aplicaciones (*Application framework*). Por último, en la capa de mayor nivel, es donde se sitúan las aplicaciones instaladas en el dispositivo, ya sean pertenecientes al usuario o nativas del sistema.

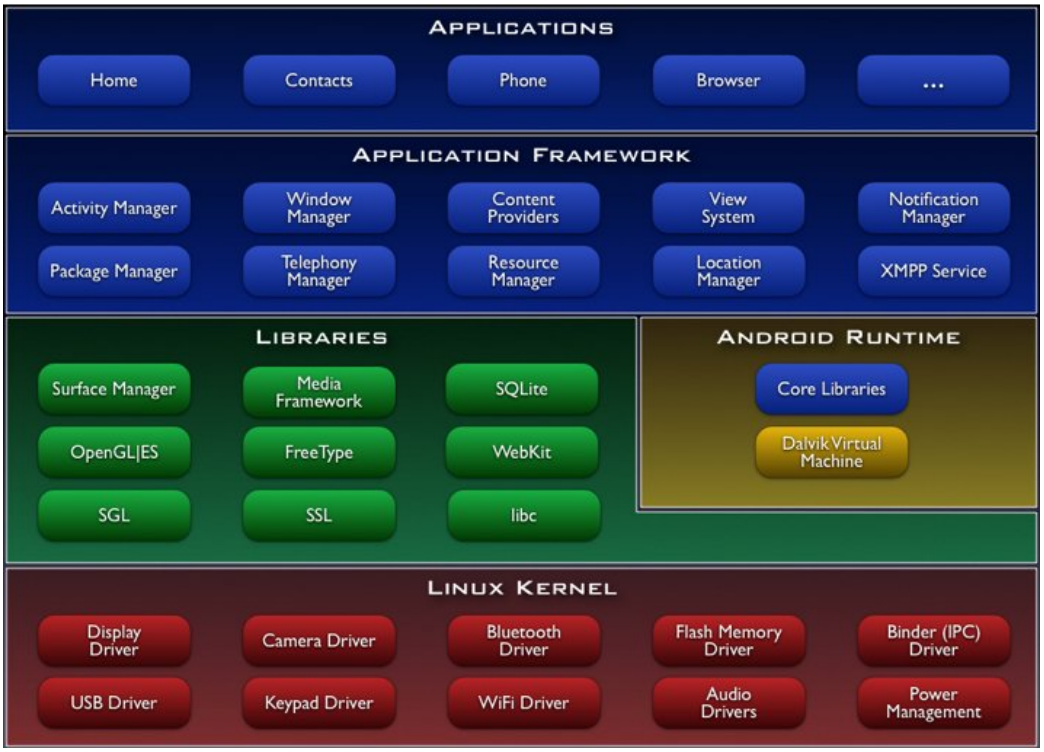

Figura A.2: *Arquitectura del sistema operativo Android. Imagen obtenida de developer.android.com.*

Las aplicaciones desarrolladas para Android se escriben en código Java. Una vez compiladas y generado el correspondiente bytecode, éste se convierte al formato de ejecución soportado por Dalvik (.dex). Para facilitar el desarrollo de aplicaciones, Android proporciona un kit de desarrollo software (*Software development kit o SDK*) gratuito a través de su página web (developer.android.com). Este kit incluye, además del entorno gráfico de desarrollo, una herramienta que permite depurar el código generado (*Dalvik Debug Monitor Server o ddms*) así como un emulador para probar las aplicaciones desarrolladas antes de instalarlas en el dispositivo físico (*Android Emulator*).

A.2. Java Native Interface (JNI)

No obstante, teniendo en cuenta que las librerías de la FrankenCamera (FCam) necesarias para el desarrollo de esta aplicación se encuentran escritas en código C++, el lenguaje de programación Java no constituye una herramienta suficiente para este proyecto. Para poder solucionar este problema es necesario hacer uso del *Java Native Interface (JNI)*.

El JNI es un framework de programación que permite ejecutar código nativo desde Java y viceversa. El código nativo representa funciones escritas en otros lenguajes de programación como C, C++ o ensamblador. Por lo tanto, gracias a esta herramienta, las librerías de la FCam pueden ser independientemente compiladas, e incluidas en los ficheros que implementan los algoritmos de descomposición escritos en C++. Este código nativo es ejecutado en el tablet NVIDIA y se divide en dos etapas: en primer lugar se realiza la captura de la escena mediante la cámara integrada en el dispositivo móvil, y en segundo lugar se procesan los datos y se calculan las dos componentes de iluminación de la escena: directa y global.

A.3. Librerías de la FrankenCamera (FCam)

FCam es el resultado de la *Cámara 2.0*, proyecto de investigación llevado a cabo por Marc LeVoy (Universidad de Standford) y Kari Pulli (Nokia Research Center en Palo Alto, trasladado a NVIDIA en abril del 2011) que fue presentado en el congreso internacional de gráficos SIGGRAPH 2010 [**?**]. En la actualidad existen tres dispositivos compatibles con estas librerías: el dispositivo móvil Nokia N900, la cámara construida en los laboratorios F2 y el tablet NVIDIA Tegra.

F2 N900 tablet NVIDIA

Figura A.3: *Dispositivos compatibles con las librerías FCam.*

Las librerías de la FrankenCamera forman una API (*Application programming interface*) de código abierto que permiten el control fácil y preciso de las cámaras digitales. Permiten controlar completamente a bajo nivel todos los parámetros de la cámara para la captura de cada fotograma, haciendo posible capturar una ráfaga de imágenes cada una con propiedades diferentes. De esta forma es posible obtener efectos tan sorprendentes como el que se muestra en la figura A.4

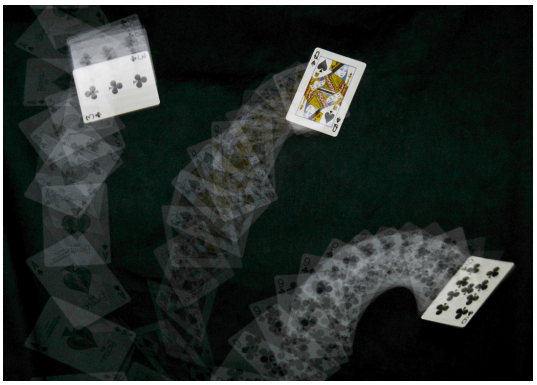

Figura A.4: *Para tomar esta imagen la cámara dispuso de dos flash, cada uno ajustado con un tiempo de duración distinto.*

La API se compone de cuatro clases principales: *Shot*, *Sensor*, *Device* y *Frame*. La clase *Shot* especifica los parámetros de captura y post-proceso de una única imagen. Una instancia de la clase *Shot* especifica parámetros del sensor tales como ganancia, tiempo de exposición, resolución, formato, balance de blancos,etc. Una instancia de la clase *Frame* contiene la imagen resultante junto con la información generada por hardware como histogramas o mapas de contorno. La clase *Device* puede ser programada para ejecutar acciones (p.ej: disparar el flash). En la figura A.5 se observa la relación entre cada una de las clases.

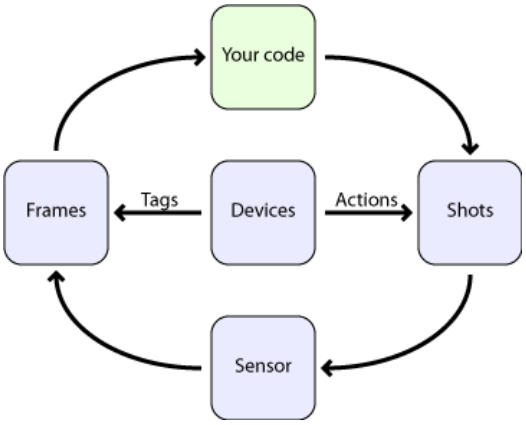

Figura A.5: *Para usar la API de la FrankenCamera, se pasan instancias de la clase Shot al sensor de la cámara (clase Sensor), que devuelve asíncronamente Frames, los cuales contienen la imagen resultante.*

A.4. Estructura de la aplicación móvil

Una vez analizados todos los componentes necesarios (Android, JNI y FCam), el siguiente paso a seguir consiste en combinar todos ellos para poder desarrollar la aplicación móvil deseada. En Android, las actividades (*Activity*) son clases públicas que representarán cada una de las pantallas de nuestra aplicación. Cada actividad cuenta con, al menos, un método llamado *onCreate()*. Este método es el primero en ser invocado cuando la actividad se inicia. Su análogo es el método *onCompletion()* que se invoca cuando la actividad finaliza.

La aplicación móvil desarrollada para este proyecto cuenta con dos pantallas distintas: una para realizar el proceso de captura de la escena y otra para tratar los datos capturados y obtener la descomposición. Estas pantallas se visualizan a modo de pestañas dentro de una tercera, la pantalla principal. Por lo tanto, tal y como muestra el diagrama de clases de la figura A.6, se tiene una clase distinta para cada una de estas tres pantallas. La pantalla principal (*Main_Activity*) constituye el punto de entrada de la aplicación. Cada una de las clases correspondientes a las otras dos pestañas (*Process_Activity* y *Capture_Activiry*) cuenta con los métodos *onCreate()*, *onCompletion()* y con la declaración del método nativo *run()*. Este último, es un método abstracto cuya implementación, en código C++, se encuentra en la clase que implementa el framework de JNI (*Java_Adapter*). Así pues, en esta clase se tienen, entre otros, los métodos *CaptureActivity_run()* y *ProcessActivity_run()* que a su vez, invocan a los métodos *fcam_thread_capture()*

y *fcam_thread_process()* respectivamente para lanzar sendos hilos que ejecuten el código C++ correspondiente. En dicho código, es posible llamar a las funciones de la librería FCam con tan sólo incluir su cabecera al inicio del mismo (#include <FCam/Tegra.h>).

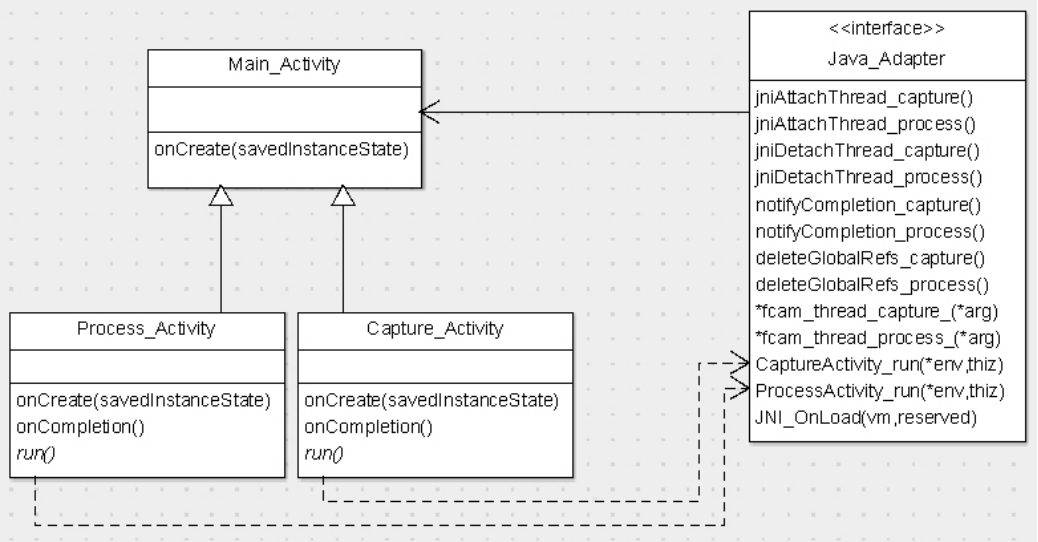

Figura A.6: *Diagrama de clases para la aplicación móvil.*

A.5. Arquitectura NVIDIA Tegra 3

Los procesadores NVIDIA Tegra 3 cuentan con cuatro núcleos y ofrecen un excelente rendimiento en navegación y contenidos flash así como en juegos gracias a la GPU NVIDIA GeForce ULP de bajo consumo. Las principales innovaciones que presenta la última versión de este chip con respecto de su predecesor (Tegra 2) son las siguientes:

1. **Tecnología 4Plus-1.** La nomenclatura 4 + 1 se refiere a la arquitectura de procesadores de Tegra 3, formada por cuatro núcleos de CPU más un quinto núcleo dedicado a ahorrar batería. Se trata de una arquitectura SMP variable que permite utilizar los cuatro núcleos de alto rendimiento para las tareas más pesadas, de manera que cada uno de ellos se activa o desactiva de forma independiente y automática en función de la carga de trabajo. El núcleo de ahorro energético (o coprocesador) maneja tareas que necesitan menos potencia, como el estado

de espera activa o la reproducción de vídeo y música, y lo hace de forma transparente para el SO y las aplicaciones.
A diferencia de los ordenadores de sobremesa, en los cuales se produjo una transición lenta desde la aparición de los primeros procesadores con múltiples núcleos hasta su completo aprovechamiento, en el campo de los procesadores móviles esta transición ha sido mucho más rápida en el tiempo. Ya la antigua versión 2.3 del sistema operativo Android incluía soporte para procesado con múltiples núcleos y hoy en día, con la versión 4.0 de este sistema operativo, el rendimiento ha mejorado considerablemente.

2. **NVIDIA DirectTouch.** Esta tecnología mejora la respuesta de la pantalla táctil y traslada al procesador Tegra 3 una parte del trabajo que habitualmente realizan los mecanismos de control táctil del dispositivo reduciendo de este modo el consumo de energía.

3. **Juego en 3D estereoscópico.** Esta opción permite aprovechar la galardonada tecnología 3D Vision de NVIDIA para convertir automáticamente (y en tiempo real) juegos y aplicaciones basados en OpenGL al formato 3D estereoscópico.

4. **Tecnología de pantalla NVIDIA PRISM.** La tecnología PRISM (Pixel Rendering Intensity and Saturation Management) reduce la potencia de retroiluminación del dispositivo móvil al mismo tiempo que mejora el color de los pixeles para brindar la misma calidad de visualización, pero con mucho menos gasto de batería.

El resto de las especificaciones técnicas de los dispositivos NVIDIA Tegra 3 quedan reflejadas en la siguiente tabla:

PROCESADOR	
Cpu	4 núcleos + 1 de bajo consumo
Máx. frecuencia	Hasta 1.5 GHz
Caché nivel L2	1 MB
Caché nivel L1	32KB por núcleo
MEMORIA	
Frecuencia	DDR3-L a 1500Mhz
Tamaño	Hasta 2GB
GPU	
Arquitectura	GeForce ULP (bajo consumo)
Núcleos de procesamiento	12
Compatibilidad 3D estereoscópico	SI
Versión de OpenGL ES	2.0
Versión de OpenVG	1.1
CÁMARA	
Cámara principal	32 Megapíxeles
Cámara secundaria	5 Megapíxeles
Megapíxeles por segundo	300
Zoom digital	Hasta 16x
Descodificación/Codificación JPEG	80 Megapíxeles por segundo
Estabilización de imágenes fijas	SI
Estabilización de video	SI