图形架构

每位开发者都应了解有关 Surface、SurfaceHolder、EGLSurface、SurfaceView、GLSurfaceView、SurfaceTexture、TextureView、SurfaceFlinger 和 Vulkan 方面的知识。

本页将介绍 Android 系统级图形架构的基本要素，并介绍应用框架和多媒体系统如何使用这些要素。我们会重点介绍图形数据的缓冲区是如何在系统中移动的。如果您想了解 SurfaceView 和 TextureView 为何采用现有的运行方式，或者想要了解 Surface 与 EGLSurface 的交互方式，本文会为您逐一解答。

假设您对 Android 设备和应用开发已有一定了解。您不需要掌握有关应用框架的详细知识，本文也很少提及 API 调用，但本材料与其他公开文档互不重叠。本文旨在详细介绍渲染帧以进行输出涉及的重要事件，从而帮助您在设计应用时做出明智的选择。为此，我们自下而上地介绍了 UI 类的工作原理，而不是它们的使用方法。

本部分包括多个页面，从背景材料到 HAL 细节再到用例，进行了全面介绍。首先是对 Android 图形缓冲区进行了解释，并说明了合成和显示机制，然后继续介绍为合成器提供数据的更高级别的机制。我们建议您按照下列顺序阅读相关页面，而不要直接跳到感兴趣的主题。

低级别组件

- BufferQueue 和 gralloc (https://source.android.com/devices/graphics/arch-bq-gralloc.html)。BufferQueue 将可生成图形数据缓冲区的组件（生产者）连接到接受数据以便进行显示或进一步处理的组件（消费者）。通过供应商专用 HAL 接口实现的 gralloc 内存分配器将用于执行缓冲区分配任务。
- SurfaceFlinger、Hardware Composer 和虚拟显示屏 (https://source.android.com/devices/graphics/arch-sf-hwc.html)。SurfaceFlinger 接受来自多个源的数据缓冲区，然后将它们进行合成并发送到显示屏。Hardware Composer HAL (HWC) 确定使用可用硬件合成缓冲区的最有效的方法，虚拟显示屏使合成输出可在系统内使用（录制屏幕或通过网络发送屏幕）。
- Surface、Canvas 和 SurfaceHolder (https://source.android.com/devices/graphics/arch-sh.html)。Surface 可生成一个通常由 SurfaceFlinger 使用的缓冲区队列。当渲染到 Surface 上时，结果最终将出现在传送给消费者的缓冲区中。Canvas API 提供一种软件实现方法（支持硬件加速），用于直接在 Surface 上绘图（OpenGL ES 的低级别替代方案）。与视图有关的任何内容均涉及到 SurfaceHolder，其 API 可用于获取和设置 Surface 参数（如大小和格

式）。

- EGLSurface 和 OpenGL ES (https://source.android.com/devices/graphics/arch-egl-opengl.html)。OpenGL ES (GLES) 定义了用于与 EGL 结合使用的图形渲染 API。EGI 是一个规定如何通过操作系统创建和访问窗口的库（要绘制纹理多边形，请使用 GLES 调用；要将渲染放到屏幕上，请使用 EGL 调用）。此页还介绍了 ANativeWindow，它是 Java Surface 类的 C/C++ 等价类，用于通过原生代码创建 EGL 窗口表面。
- Vulkan (https://source.android.com/devices/graphics/arch-vulkan.html)。Vulkan 是一种用于高性能 3D 图形的低开销、跨平台 API。与 OpenGL ES 一样，Vulkan 提供用于在应用中创建高质量实时图形的工具。Vulkan 的优势包括降低 CPU 开销以及支持 SPIR-V 二进制中间 (https://www.khronos.org/spir)语言。

高级别组件

- SurfaceView 和 GLSurfaceView (https://source.android.com/devices/graphics/arch-sv-glsv.html)。SurfaceView 结合了 Surface 和 View。SurfaceView 的 View 组件由 SurfaceFlinger（而不是应用）合成，从而可以通过单独的线程/进程渲染，并与应用界面渲染隔离。GLSurfaceView 提供帮助程序类来管理 EGL 上下文、线程间通信以及与“Activity 生命周期”的交互（但使用 GLES 时并不需要 GLSurfaceView）。
- SurfaceTexture (https://source.android.com/devices/graphics/arch-st.html)。SurfaceTexture 将 Surface 和 GLES 纹理相结合来创建 BufferQueue，而您的应用是 BufferQueue 的消费者。当生产者将新的缓冲区排入队列时，它会通知您的应用。您的应用会依次释放先前占有的缓冲区，从队列中获取新缓冲区并执行 EGL 调用，从而使 GLES 可将此缓冲区作为外部纹理使用。Android 7.0 增加了对安全纹理视频播放的支持，以便用户能够对受保护的视频内容进行 GPU 后处理。
- TextureView (https://source.android.com/devices/graphics/arch-tv.html)。TextureView 结合了 View 和 SurfaceTexture。TextureView 对 SurfaceTexture 进行包装，并负责响应回调以及获取新的缓冲区。在绘图时，TextureView 使用最近收到的缓冲区的内容作为其数据源，根据 View 状态指示，在它应该渲染的任何位置和以它应该采用的任何渲染方式进行渲染。View 合成始终通过 GLES 来执行，这意味着内容更新可能会导致其他 View 元素重绘。

上次更新日期：八月 24, 2017