SurfaceTexture

SurfaceTexture 类是在 Android 3.0 中推出的。就像 SurfaceView 是 Surface 和 View 的组合一样，SurfaceTexture 是 Surface 和 GLES 纹理的粗略组合（包含几个注意事项）。

当您创建 SurfaceTexture 时，会创建一个应用是其消耗方的 BufferQueue。如果生产方将新的缓冲区加入队列，您的应用便会通过回调 (`onFrameAvailable()`) 获得通知。应用调用 `updateTexImage()`（这会释放先前保留的缓冲区），从队列中获取新的缓冲区，然后发出一些 EGL 调用，让缓冲区可作为外部纹理供 GLES 使用。

外部纹理

外部纹理 (`GL_TEXTURE_EXTERNAL_OES`) 与 GLES (`GL_TEXTURE_2D`) 创建的纹理并不完全相同：您对渲染器的配置必须有所不同，而且有一些操作是不能对外部纹理执行的。关键是，您可以直接从 BufferQueue 接收到的数据中渲染纹理多边形。gralloc 支持各种格式，因此我们需要保证缓冲区中数据的格式是 GLES 可以识别的格式。为此，当 SurfaceTexture 创建 BufferQueue 时，它将消耗方用法标记设置为 `GRALLOC_USAGE_HW_TEXTURE`，确保由 gralloc 创建的缓冲区均可供 GLES 使用。

由于 SurfaceTexture 会与 EGL 上下文交互，因此您必须小心地从正确的会话中调用其方法（详见类文档）。

时间戳和转换

如果您深入阅读类文档，您会看到两个奇怪的调用。一个调用会检索时间戳，而另一个调用是转换矩阵，每个调用的值都通过事先调用 `updateTexImage()` 来设置。事实证明，BufferQueue 不只是向消耗方传递缓冲区句柄。每个缓冲区都附有时间戳和转换参数。

提供转换是为了提高效率。在某些情况下，源数据可能以错误的方向传递给消耗方；但是，我们可以按照数据的当前方向发送数据并使用转换对其进行更正，而不是在发送数据之前对其进行旋转。在使用数据时，转换矩阵可以与其他转换合并，从而最大限度降低开销。

时间戳对于某些缓冲区来源非常有用。例如，假设您将生产方接口连接到相机的输出端（使用 `setPreviewTexture()`）。要创建视频，您需要为每个帧设置演示时间戳；不过您需要根据截

取帧的时间（而不是应用收到缓冲区的时间）来设置该时间戳。随缓冲区提供的时间戳由相机代码设置，从而获得一系列更一致的时间戳。

SurfaceTexture 和 Surface

如果仔细观察 API，您会发现应用只能通过一种方式来创建简单 Surface，即通过将 SurfaceTexture 作为唯一参数的构造函数来创建。（在 API 11 之前，根本没有用于 Surface 的公开构造函数。）如果将 SurfaceTexture 视为 Surface 和纹理的组合，这看起来可能有点落后。

深入来看，SurfaceTexture 称为 GLConsumer，它更准确地反映了其作为 BufferQueue 的所有方和消耗方的角色。从 SurfaceTexture 创建 Surface 时，您所做的是创建一个表示 SurfaceTexture 的 BufferQueue 生产方端的对象。

案例研究：Grafika 的连续拍摄

相机可以提供一个适合作为电影进行录制的帧流。要在屏幕上显示它，您可以创建一个 SurfaceView，将 Surface 传递给 `setPreviewDisplay()`，然后让生产方（相机）和消耗方 (SurfaceFlinger) 完成所有工作。要录制视频，您可以使用 MediaCodec 的 `createInputSurface()` 创建一个 Surface，将其传递给相机，然后便可高枕无忧了。若要边显示边录制，您必须使用更多内容。

在录制视频时，连续拍摄 Activity 会显示相机录制的视频。在这种情况下，已编码的视频将写入内存中的环形缓冲区，该缓冲区可随时保存到磁盘。实现起来非常简单，只要您跟踪所有内容所在的位置即可。

该流程涉及三个 BufferQueue，分别是由应用、SurfaceFlinger 和 mediaserver 所创建：

- 应用。该应用使用 SurfaceTexture 从相机接收帧，并将其转换为外部 GLES 纹理。
- **SurfaceFlinger**。该应用声明一个我们用来显示帧的 SurfaceView。
- **MediaServer**。您可以使用输入 Surface 配置 MediaCodec 编码器，以创建视频。

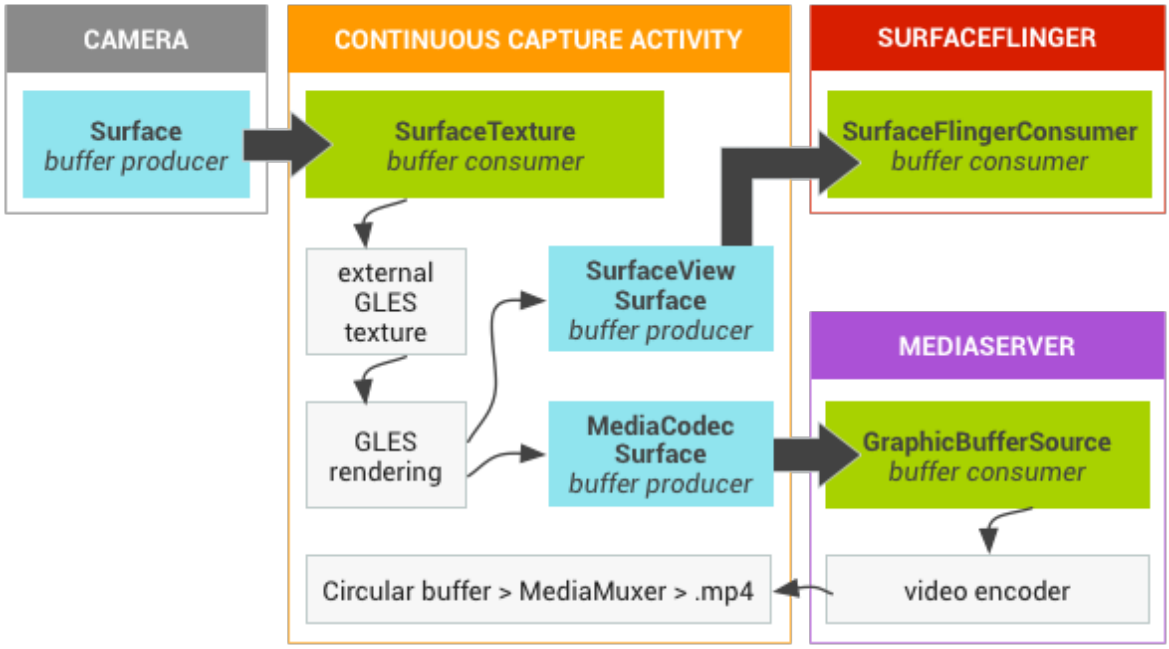

图 **1.**Grafika 的连续拍摄 Activity。箭头指示相机的数据传输路径，BufferQueue 则用颜色标示（生产方为青色，消耗方为绿色）。

已编码的 H.264 视频在应用进程中进入 RAM 中的环形缓冲区，并在点击拍摄按钮时使用 MediaMuxer 类写入到磁盘上的 MP4 文件中。

三个 BufferQueue 都在应用中通过单个 EGL 上下文处理，且 GLES 操作在 UI 线程上执行。通常，不鼓励在 UI 线程上执行 SurfaceView 渲染，但是由于我们执行的是由 GLES 驱动程序异步处理的简单操作，因此没有问题。（如果视频编码器锁定，并且我们阻止尝试将缓冲区移出队列，该应用便会停止响应。而此时，我们无论怎样操作都可能会失败。）对已编码数据的处理（管理环形缓冲区并将其写入磁盘）是在单独的线程中执行的。

大部分配置是在 SurfaceView 的 `surfaceCreated()` 回调中进行的。系统会创建 EGLContext，并为显示设备和视频编码器创建 EGLSurface。当新的帧到达时，我们会告知 SurfaceTexture 去获取它，并将其作为 GLES 纹理进行提供，然后使用 GLES 命令在每个 EGLSurface 上渲染它（从 SurfaceTexture 转发转换和时间戳）。编码器线程从 MediaCodec 拉取编码的输出内容，并将其存储在内存中。

安全纹理视频播放

Android 7.0 支持对受保护的视频内容进行 GPU 后处理。这允许将 GPU 用于复杂的非线性视频效果（例如扭曲），将受保护的视频内容映射到纹理，以用于常规图形场景（例如，使用 OpenGL ES）和虚拟现实 (VR)。

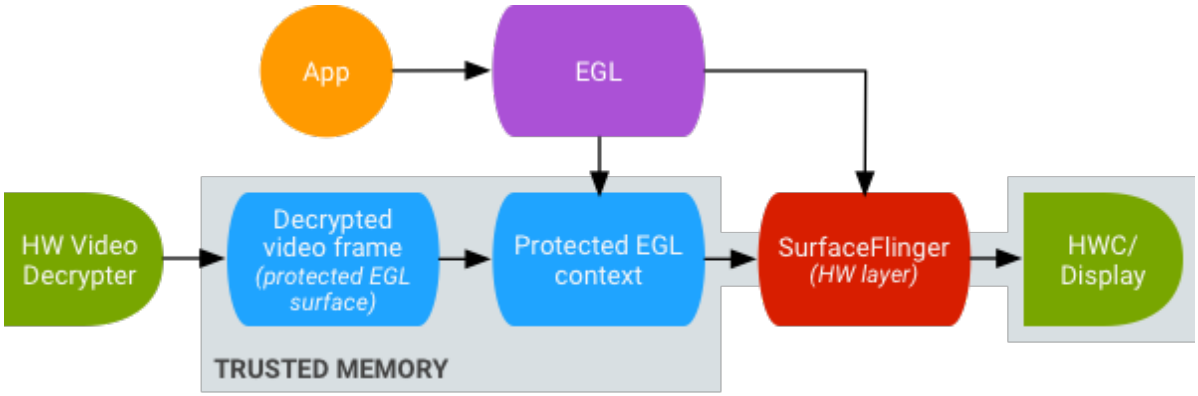

图 **2.**安全纹理视频播放

使用以下两个扩展程序实现支持：

- **EGL** 扩展程序 (`EGL_EXT_protected_content` (https://www.khronos.org/registry/egl/extensions/EXT/EGL_EXT_protected_content.txt))。允许创建受保护的 GL 上下文和 Surface，它们都可以对受保护的内容进行操作。
- **GLES** 扩展程序 (`GL_EXT_protected_textures` (https://www.khronos.org/registry/gles/extensions/EXT/EXT_protected_textures.txt))。允许将纹理标记为受保护，以便它们可以用作帧缓冲区纹理附件。

Android 7.0 还会更新 SurfaceTexture 和 ACodec (`libstagefright.so`)，这样一来，即使窗口 Surface 没有加入窗口编写器（例如 SurfaceFlinger）的队列，也允许发送受保护的内容，并且提供受保护的视频 Surface 以在受保护的上下文中使用。该操作通过在受保护的上下文（由 ACodec 验证）中创建的 Surface 上设置正确的受保护消耗方位 (`GRALLOC_USAGE_PROTECTED`) 来完成。

这些更改可使以下对象受益：应用开发者，他们可以创建应用来执行增强的视频效果，或在 GL 中（例如在 VR 中）使用受保护的内容来应用视频纹理；最终用户，他们可以在 GL 环境中（例如在 VR 中）查看高价值的视频内容（如电影和电视节目）；以及原始设备制造商 (OEM)，他们可通过增加设备功能（例如在 VR 中观看高清电影）提高产品销量。新的 EGL 和 GLES 扩展程序可以由系统芯片 (SoC) 提供商和其他供应商使用，目前是在 Nexus 6P 中使用的 Qualcomm MSM8994 SoC 芯片组上实现。

安全纹理视频播放为在 OpenGL ES 环境中进行强大的 DRM 实现奠定了基础。如果没有诸如 Widevine 级别 1 这样的强大 DRM 实现，许多内容提供商将不允许在 OpenGL ES 环境中渲染其高价值内容，从而阻止重要的 VR 用例（例如在 VR 中观看受 DRM 保护的内容）。

AOSP 包括用于安全纹理视频播放的框架代码；驱动程序支持取决于供应商。设备实现人员必须实现 `EGL_EXT_protected_content` 和 `GL_EXT_protected_textures extensions`。使用您自

己的编解码器库（替代 libstagefright）时，请注意，只要消耗方用法位包含 **GRALLOC_USAGE_PROTECTED**，**/frameworks/av/media/libstagefright/SurfaceUtils.cpp** 中的更改便会允许将标记为 **GRALLOC_USAGE_PROTECTED** 的缓冲区发送到 ANativeWindow，即便 ANativeWindow 没有直接加入窗口编写器的队列也可以。有关实现扩展程序的详细文档，请参见 Khronos 注册表（EGL_EXT_protected_content (https://www.khronos.org/registry/egl/extensions/EXT/EGL_EXT_protected_content.txt)、GL_EXT_protected_textures (https://www.khronos.org/registry/gles/extensions/EXT/EXT_protected_textures.txt)）。

设备实现人员可能还需要进行硬件更改，才能确保映射到 GPU 的受保护内存仍受保护，且不可由未受保护的代码读取。

上次更新日期：九月 13, 2017