理解光栅库 Z.MemoryRaster 的结构

Z.MemoryRaster 库是个庞大的体系,先说说 TMRaster,光栅库主体的下层体系,它们都集成在 Z.MemoryRaster 库的内部.

- **TRasterSerialized**,光栅库序列化容器,文件 IO 容器,TMRaster 被空置时,RGB/BGRA 数据将使用文件 IO 来暂存,从而释放出更多的内存
- **TSequenceMemoryRaster**,光栅库序列帧支持,例如烟花,爆炸,都是一种序列帧,序列帧素材可以从淘宝这类网站廉价购买,基本不需要自己做. Animate,Toon Boon Harmony,TVPaint,使用这类工具也可以自己制作序列帧素材.
- **AGG**,这是一个综合的大库,pas 版本移植自 C 库,在 TMRaster 中只使用了反走样绘图功能.
- **TRasterVertex**,顶点渲染器,这是一个基于顶点的软渲器,在 Z.DrawEngine 如果使用软光栅作为输出目标,那么 TRasterVertex 就是主要使用的接口.
- **TFontRaster**,光栅字体支持,这是压缩后的光栅格式字体,同时也能存储和载入 TMRaster 自定义字体.
- **TMorphomatics**,形态学支持,这是将 RGB/BGRA 光栅分解成不同的浮点结构,然后进行形态学计算,包括 20 来个滤波器,符号支持,以及形态学中必备的腐蚀塌陷开闭这类操作,同时也可用多种方法转换二值化结构.
- **TMorphologyBinaryzation**,形态学二值化支持,TMRaster 光栅必须通过形态学才能二值化,当二值化以后,可进行腐蚀塌陷开闭这类操作,二值化方法在这里有完整支持体系,其操作思路与使用 OpenCV 类似.
- **TMorphologySegmentation**,形态学分割器,分割和抠图不同,分割是产生关键数据,抠图是数据+处理.
- **TMorphologyRCLines**,纵列线段识别,只能识别绝对直线和竖线,并形成数据结构

TMRaster 文件格式支持体系,均为 Z 系自主实现格式

- **(标头$4D42)Bmp 格式**:只能支持 24bit/32bit 两种格式的 bmp,支持方式为原生内置,不依赖于外部库,不支持 64k,256 色,16 色,灰位等格式.
- **(标头$8D42) 32 位 Bmp 被 Zlib 压缩后格式**:这并不是 png 分通道压缩,而是直接对已经编码的 bmp 进行压缩,这种格式可以保留 RGB+Alpha 通道,支持方式为原生内置,不依赖于外部库,效率优于 Png,压缩比不如 Png.
- **(标头$8D43) 32 位 Bmp 被 Deflate 压缩后格式**:这并不是 png 分通道压缩,而是直接对已经编码的 bmp 进行压缩,这种格式可以保留 RGB+Alpha 通道,支持方式为原生内置,不依赖于外部库
- **(标头$8D44) 32 位 Bmp 被 BRRC 压缩后格式**:这并不是 png 分通道压缩,而是直接对已经编码的 bmp 进行压缩,这种格式可以保留 RGB+Alpha 通道,支持方式为原生内置,不依赖于外部库

- **(标头$D8FF/$8DFF+子标头$F7FF)JLS 格式**:JLS 是一直准备移除的格式,因为 JLS 有几个问题暂时无法解决,一是算法效率很低,其次是图像尺寸必须可以被 4 整除,否则像素将会错位.JLS 主要做为外部库来支持,外部库以 Z.JLS.*.pas 开头
- **(标头$D8FF/$8DFF)Jpeg 格式**:图像如果不要求 Alpha 通道和 100%保真,那么 Jpeg 就是主力存储格式,JPEG 支持像素兼容排列(CMYK,YcbCr,RGB,BGR,Gray,可兼容 ps),带 Alpha 通道的非兼容像素排列(YcbCrA:RGB 带 A,GrayA:灰度位带 A),JPEG 主要做为外部库来支持,外部库以 JPEGZ.MemoryRaster.JPEG.*.pas 开头

JPEG 像素转换兼容(可以 Load 的像素排列格式):

1. 8 位:单像素占用 1Byte
2. 16 位: 单像素占用 2Byte
3. 32 位:单像素占用 4Byte
4. Triplets 格式:3 像素格式,通常这种格式为 BGR 或则是 RGB
5. JFIF 格式:JIF 和 Jpeg 的兼容像素排列格式,主要为 Y-Cb-Cr 与 BGR/RGB 的互转
6. CMYK 格式:CMYK 是 Photoshop 提出了一种像素排列格式,具体细节自己查资料,这里的 CMYK 可以支持与 BGR/RGB 的互转
7. YCCK,CIE,ITU CIE,CIELab,YUV 格式:这几个色彩格式只能被 Load,JPEG 提供正转换,不支持逆转换,无法在保存时使用这些格式.

- **(标头$8D46)YUV 格式**:YUV 大约 4 个临近像素共享一个 UV 颜色,因此体积大幅度降低,这里的 YUV 是非公共格式,并不支持外部软件例如 ps,除此之外,YUV 格式可以有压缩和非压缩两种模型.原生内置支持.
- **(标头$8D47)half YUV 格式**:YUV 格式的 1/2 尺度,同样可以支持压缩和非压缩,非标格式. 原生内置支持.
- **(标头$8D48) Quart YUV 格式**:YUV 格式的 1/4 尺度,同样可以支持压缩和非压缩,非标格式. 原生内置支持.
- **(标头$8D50)灰度格式**:直接从 color buffer 编码.原生内置支持.
- **(标头$8D51)256 色格式**:直接从 color buffer 编码.原生内置支持.
- **(标头$8D52)64k 色格式**:直接从 color buffer 编码.原生内置支持.
- **(标头$89504E97,简单来说标头就是$89+PNG)PNG 格式**:作为外部 Png 库来支持(Z.Raster.PNG.pas),PNG 格式压缩会检测 RGBA 的使用状态,从而选择一种最佳的无损压缩方式,表现在使用时 Save 会非常慢,Load 则是正常性能.

TMRaster 的外部文件格式: 当引用过 Z.DrawEngine.FMX.pas 这类接口库以后,TMRaster 在读取图片格式时会优先检查内部格式是否支持该图片,如果不支持,那么就使用 FMX 的图片解码器来读取图片,例如在 Win 系统,FMX 会 使用内置的 jpeg 库,而在 android,ios 这类系统会系统操作系统自带的 api 来读取图片,通常来说,内置格式永远都是最优的选择,它不光是可以支持 fpc,还可以支持非 fmx 框架应用.

FFMPEG 所有视频帧的编码解码都以 TMRaster 作为数据源

TMRaster 作为像素容器用于存储解码和编码

DrawEngine 对 TMRaster 的相互引用体系

DrawEngine 是独立的渲染器,通过接口来工作,理论上它可以接口任何图形系统

TMRaster 可以直接在内部使用 DrawEngine,这时候,DrawEngine 会直接以软渲方式输出到当前的 TMRaster 中来.

例如在 FFMPEG 支持体系中,所有的视频帧的光栅都以 TMRaster 进行解码或编码,在解码时可以通过 DrawEngine 在光栅中画需要的内容,然后再来输出到屏幕.编码同样,可以通过 DrawEngine 来画内容,然后再编码成 mp4,mkv.

在 AI 和 CV 体系中所有的图像均以 TMRaster 作为数据源

因为 CV 不光是外部 C++那边的算法,还会涉及到许多形态学,像素分割这类功能,因此 TMRaster 也是一种 AI 的二次计算体系.

2024-1

By.qq600585