Emb 的 canvas 设计思路是包，也可以说封装，一旦涉及到绘制性能，大批量绘制，canvas 就会变得很鸡肋，随便放几张图一般只有个位数帧率。

要把 Canvas. DrawBitmap 的性能瓶颈说清，先要解释一下实现 canvas 硬件加速的设计思路，GDI 和 DX 的 software 模式不在此范围。

论硬件加速核心实现思路：
FMX 所支持硬件 api 的结构，主要是 3 种，DX,GL,GLES，这些硬件 api，凡是涉及到画图片，都以图片句柄作为实例来绘制，也就是说图片是常驻于内存 or 显存中，最底层的 api 在绘制时，只需要告诉硬件，这张图片的句柄（也就是常用的 index，id 这类东西），这时候，sim 指令流被分发给并发 gpu 单元执行绘制。

Bitmap 在硬件加速中的作用：
在 FMX 中，所有最底层的硬件加速，都是基于 Tbitmap 进行的封装。大致可以理解为，tbitmap 被载入时，是放在内存中的，它只是光栅化的数列块，并没有实际作用。而当我们使用硬件加速绘制时，通过 TTextureBitmap 之类的接口，tbitmap 的光栅化数列块，就被复制到了某个地方，这个地方它可以是显存，内存，情况不定。这时，我们就有了硬件加速的实例化句柄。

实现加速绘制：
FMX 的 Canvas 加速机制是以封装成傻瓜包为主，绕过了许多专业图形学常识，比如绘制批次，深度状态，zbuff，顶点 shader，像素 shader，几何 shader，纹理正反面，纹理光栅化，许多许多，都被绕过了。Fmx 所封装的所有的纹理，都是在 bitmap 上所做的扩展，它不是图形学，它是封装，为了易用性而牺牲标准图形程序编程范式。

理解和使用 FMX 硬件加速：
我给 fmx 硬件加速总结的要点只有两处，
第一是 bitmap 必须被 texture 实例化，而不是每次绘制再去实例化一次，因为每次绘制，做实例化，如 gpu 单元中的 DoDrawBitmap 方法，下图

```
procedure TCanvasGpu.DoDrawBitmap(const ABitmap: TBitmap; const SrcRect, DstRect: TRectF; const AOpacity: Single;
  const HighSpeed: Boolean);
var
  B: TBitmapCtx;
  Bmp: TBitmap;
begin
  if FContext <> nil then
  begin
    Bmp := ABitmap;
    if Bmp.HandleAllocated then
    begin
      B := TBitmapCtx(Bmp.Handle);
      FCanvasHelper.TexRect(TransformBounds(CornersF(DstRect)), CornersF(SrcRect.Left, SrcRect.Top, SrcRect.Width,
        SrcRect.Height), B.PaintingTexture, PrepareColor(FModulateColor, AOpacity));
    end;
  end;
end;
```

这个调用 B.PaintingTexture 假如每次画图片，都要经过一次 copy 内存到实例的操作，将会非常慢，所以我们需要让这个 B 参数，能够常驻在实例化所需要的地方（内存 or 显存）。

第二是你要记住，凡是移动平台，甚至桌面平台，FMX 默认的底层，都是开了硬件加速的，不要怀疑自己的程序没有硬件加速。如果绘制缓慢，先检查 bitmap 是否被实例化过了，或则是否主动设置了 software 画图模式。

深入到 Android 和苹果手机去了解 Canvas 的实例化加速：
首先，安卓和苹果，在经过 Bitmap 实例化时，都使用了 TTextureBitmap 作为扩展 Bitmap 的接口，在 TTextureBitmap 中，我们看到了 Shader 的封装：将实例化句柄作为传递给 shader。EMB 的 shader 并没有写成明文，而是将明文裁短，以二进制方式来书写，我想 emb 的用心是不想让 app 开发流去了解 shader 层面的技术了。

另一方面，在 TContext3D（所有的加速接口都会通过它和底层 GLES 交互）中，凡事涉及到了矩阵操作，均是自己实现，不依赖 GL api 操作矩阵。

由此，我么可以确定手机上的 Canvas 是基于 GLES2.0 的框架。
个人建议，Shader 是个超级大坑，没几年图形学沉淀，最好远离它，这无关你的数学功底，主要是它太专业，脱机生产和工作的实际了，所以我建议大家远离它。

最后补充：
我本来想发点 Demo 上来，后来一想，觉得不妥，因为混混会粘贴拷贝，四处招慌撞骗，反而降低了专业门槛的层次。所以我就改写文档，把核心思路给说了，让勤快的人自己动手动脑去学习研究。

by2015-6-26，qq600585