

加餐 | 前端与图形学 | 极客时间

winter 2019-05-02



(点击视频观看完整分享内容。)

主要内容

关于前端和图形学，我分成了三个部分来讲解。



- 第一部分是讲前端和图形学有什么样的关系。我们为什么要在前端里引入图形学，这个也是我的一段心路历程。
- 第二部分相对来说是比较实用的，就是图形学的应用场景。如何在前端的日常的工作中，把图形学的知识用进去，为我们的工作和业务创造价值。
- 最后一部分是对图形学基础设施的一些建设，目前还是一个比较初级的阶段，但是对大家来说，应该有些思路还是可以去思考的。

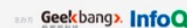
前端和图形学

首先讲第一部分前端和图形学，先讲讲缘起。

缘起

缘起

- 2011 gesture animation scroll
- 2013 flexible design
- 2016- 2017 BindingX
- 2018 ?
- 对齐iOS体验
- 解决适配问题
- 通用交互领域模型



2011 年我做了一个分享，当时 HTML5 正火，我讲了这么一个内容叫做 gesture animation，我是用 HTML5 上的 TouchEvent，去模拟当时非常惊艳的 iOS 的操作风格。

2013 年我又开始讲一个叫做 flexible design 的东西，这是针对当时一个非常火的概念提的。那个概念能从最大的屏幕适配到最小的屏幕的一个技术方案，但是我们从当时的实现来看，这个想法是好的，概念也是高大上的，但是从落地上来看非常困难。考虑到现实情况，我提了一个 flexible design 这样的一个个级概念。

这个就是一个弹性、小范围的适配，我们只把不同的尺寸的安卓机和 iPhone 做适配，最后解决了适配的问题，并提出来了一系列的设计原则。所以在 2013 年，我们主要做得还是解决适配问题。

2016 到 2017，我在各种不同的会议上讲了三场演讲，它们的背后其实都是同一个东西，叫做 BindingX。

我是希望提出一个交互领域的通用模型，我把交互抽象成输入、输出和中间的一个表达式。通过三者之间的关系，来建立针对所有交互的领域模型。

我的三场分别是技术的角度、从架构的角度，和从团队基础设施建设的角度，讲了三次。差不多两年的时间，我一直在研究这个方向。

16 年初的时候，我做了一件事，我让团队的一个小伙伴去找当时所有看起来比较先进的设计，他到网上到处去找，总结了这么一份 PPT。

然后呢，16 年初的时候，我们就对着 PPT 开始研究说，到底哪个东西还是我们现在的基础设施实现不了的。我们用 binding 尽量去实现了。但是还是有一部分在 2016 年是做不到的。

到了 2018 年，我们又做了一次一样的事情，我们把当时的这个 PPT 拿出来，说这个效果还有没有我们做不到的，我们发现整个的效果，我们已经全都能做到了。

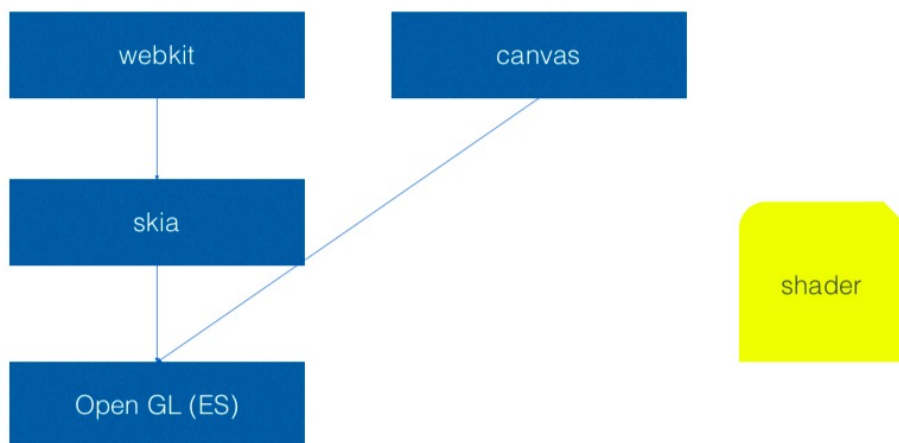
当时我们觉得作为前端，至少从底层能力上来说，我们已经建设得很好了，市面上能看到的先进的交互，我们都是可以做出来的。

不过，我还是做了一些思考，其实还是能找到一些做不出来的效果，比如说一些光和影的效果，还是我们现在做不出来的。

浏览器的图形学

对浏览器来说，图形这一条线下来，它大概会是个这样一个依赖关系。

浏览器中的图形学

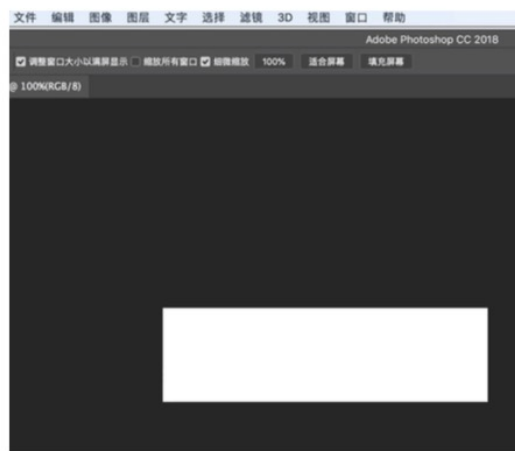


我发现前面觉得自己做不出来的东西，实际上毫无疑问都可以用 OpenGL 的 API 去解决，我觉得它其实除了大家耳熟能详的“做 3D”这一能力之外，是不是还可以用来解决我们在渲染方面的一些问题。

设计稿里的图形学

除了技术的角度，我们也做了另一个角度的分析，我们考察了一下设计师最常用的这个工具，Photoshop。它有一个工具叫滤镜。Photoshop 里能够画出来东西的，都是通过滤镜实现的。

设计稿中的图形学

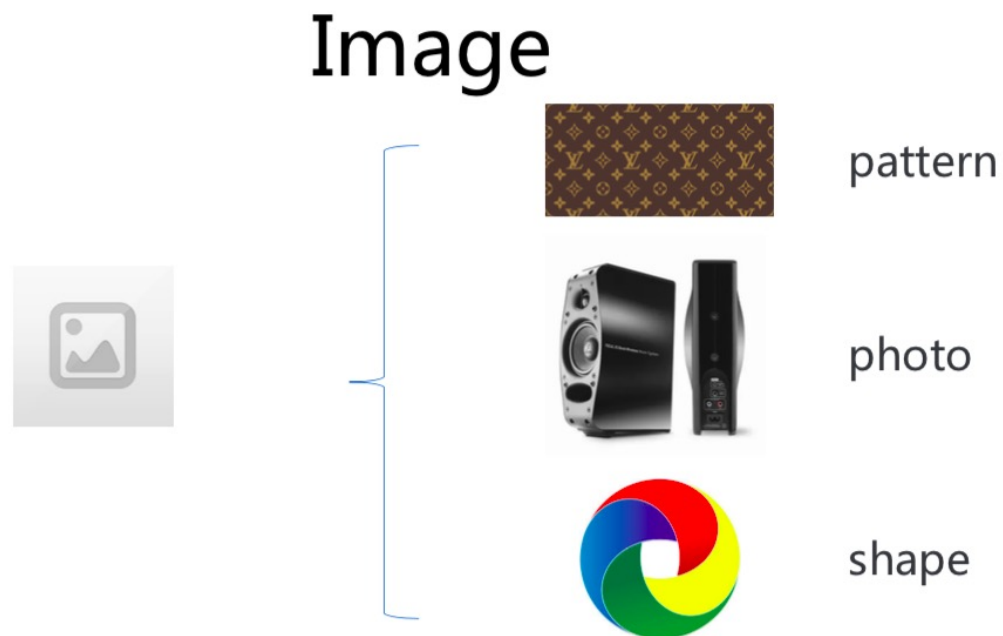


所以，我们做了一些基础渲染型的滤镜，也有一些对图片处理的滤镜，通过对它们的灵活组合，我们可以实现各种各样的图案，比如画个火焰、画个云彩类似的效果。

基于此，我们又做了一些探索。

图形学应用场景

我们把 Photoshop 生成的图片其实分了一些不同的种类。



一种叫做图案，这个它可能是一种重复率比较高的，也可能是不重复，但是它相对来说是一种多用于背景的这样的一种东西。

还有一种，就是 Photo 图片，图片基本上就是拍照拍出来的，比如说这个图里的一个音箱，这个东西你没法去用技术去代替它，就是真实的图像。

还有一种东西叫做形状，比如三角形、圆形、方形，形状已经在浏览器里用了很成熟的技术去实现。

来自设计稿的图形：云雾

那么我们现在重点要去解决的是第一种 pattern。比如我们要实现云雾效果。

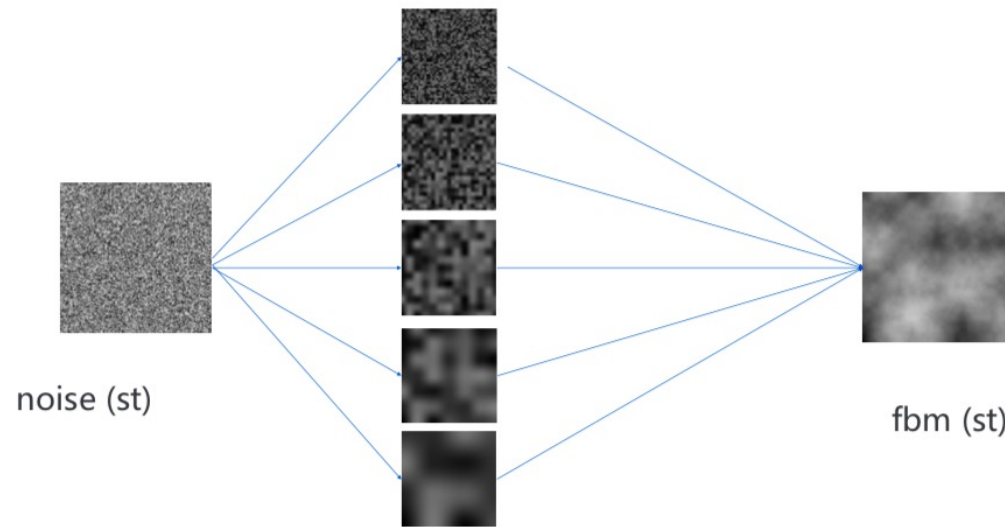
来自设计稿的图形——云雾

```
float noise (in vec2 st) {  
    vec2 i = floor(st);  
    vec2 f = fract(st);  
  
    // Four corners in 2D of a tile  
    float a = random(i);  
    float b = random(i + vec2(1.0, 0.0));  
    float c = random(i + vec2(0.0, 1.0));  
    float d = random(i + vec2(1.0, 1.0));  
  
    vec2 u = f * f * (3.0 - 2.0 * f);  
  
    return mix(a, b, u.x) +  
           (c - a) * u.y * (1.0 - u.x) +  
           (d - b) * u.x * u.y;  
}
```

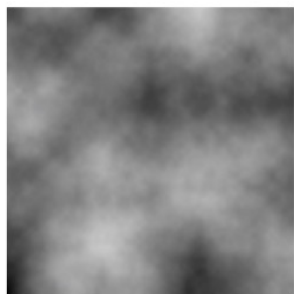
```
#define OCTAVES 6  
float fbm (in vec2 st) {  
    // Initial values  
    float value = 0.0;  
    float amplitude = .5;  
    float frequency = 0.;  
    // Loop of octaves  
    for (int i = 0; i < OCTAVES; i++) {  
        value += amplitude * noise(st);  
        st *= 2.;  
        amplitude *= .5;  
    }  
    return value;  
}
```

首先我们要有一个 noise，小时候看电视这个出雪花就是这样的，那个就是说来自硬件的噪声，当我们把这个噪声做一些处理，放大，放到最大，它就会变成一个模糊的几个块，再放的小一点，就变成几个不动的模糊的块，一直到最后就变成雪花点，但是我们把几张图以一定的比例做一定的叠加，然后就搞定了。

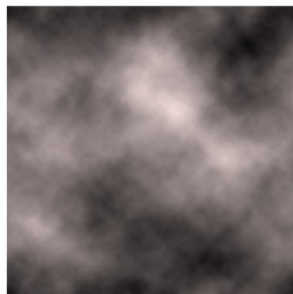
云雾



接下来我们看一下对比图，看我们的结果，我们的这个云彩和 Photoshop 这个云彩渲染出来它的形状基本上是一模一样的。



fbm (st)



photoshop

我们打开了一扇新的大门，我们仔细研究发现很多内容都是可以用 shader 去做的，如果这个想法再深入一点，我们不需要用图片了，可以直接用代码去渲染了。

我们可能未来会给设计师提供很多这样的平台、工具，让他直接在我们的这个平台上去做操作，代替原来 Photoshop 的步骤，或者我们对 Photoshop 的文件做一定的处理，来生成这种图案，这是我们的一个思路。

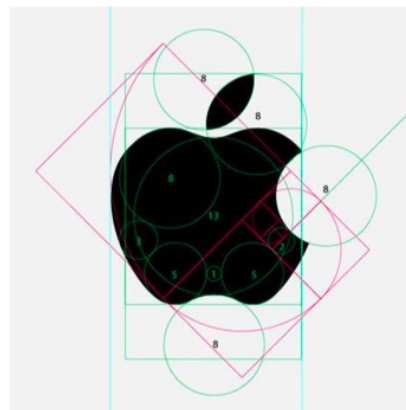
来自数学的图形：几何图形与分形

除了灵感来自 Photoshop 之外，还有些其他的来源，比如几何图形，如果大家看一下这个著名的 Logo，苹果的图标。

这幅图里面有很多的圆、框和螺旋线，它们总能找到一些数学的依据，设计师们做图标的时候都是要讲道理的，不能是凭空手绘的，尤其是这种著名公司的 icon。

来自数学的图形——几何图形

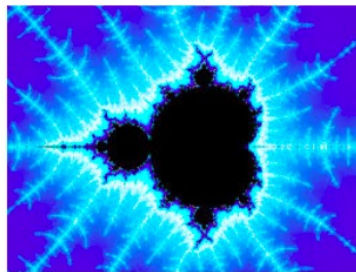
- 几何图形



大家不要去轻视这个简单的几何图形，简单的几何图形也可以产生一些非常好的效果，除了几何图形之外，来自数学的还有一类，非常著名高端，但是实现起来非常简单的，叫做分形。

很多广告片里面，它会用类似这样的图形做这个背景，分形本来是数学里面的一门学科，分形集合，研究分形图的性质，它的特点是每个部分都是大图的一个相似图形，所以说，它可以无限延伸下去。

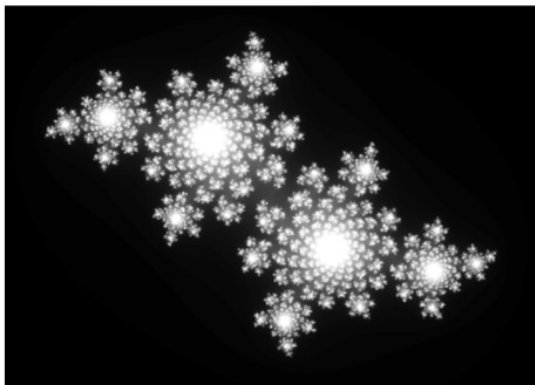
来自数学的图形——分形



分形的代码很简单，就这么多代码，但是它也是画一个像素点的，它可以画成类似雪花这样的东西。

这类的分形图呢，叫 Juila Set，我们作为前端，我们就照着上面的数学公式把它用代码实现就好，所以非常的方便，为什么我要挑出来 Juila Set 来讲，是因为它有个特点，当你用不同的常数的时候的它会产生非常不一样的图形。

Julia Set



```
const int max_iterations = 255;

void mainImage( out vec4 fragColor, in vec2 fragCoord )
{
    vec2 uv = fragCoord.xy - iResolution.xy * 0.5;
    uv *= 2.5 / min( iResolution.x, iResolution.y );

    vec2 c = vec2( -0.8, 0.156 );
    vec2 v = uv;
    float scale = 0.01;

    int count = max_iterations;

    for ( int i = 0 ; i < max_iterations; i++ ) {
        v = c + vec2( v.x * v.x - v.y * v.y, v.x * v.y * 2.0 );
        if ( dot( v, v ) > 4.0 ) {
            count = i;
            break;
        }
    }

    fragColor = vec4( float( count ) * scale );
}
```

比如说我们要做一个后台的这样的系统，给我们的设计师用，你让它自己调一调这个参数，它可以调出不同的图案。

总之，我们看到了很多的可能性，而 Juila Set 只是分形里面的一个集合，而数学里面的各种各样，能画出来奇怪花纹的东西，绝对不只分形一个，这里有非常大的想象空间。

来自物理的图形：光的衍射

还有一些来自物理的一些灵感，尤其是这种光晕效果，这个光晕效果也是 Photoshop 里面提供的，用这个光晕效果也可以做很多的设计。

来自物理的图形——光的衍射



// <http://www.pouet.net/prod.php?which=57245>

```
#define t iTime
#define r iResolution.xy

void mainImage( out vec4 fragColor, in vec2 fragCoord ){
    vec3 c;
    float l,z=t;
    for(int i=0;i<1;i++){
        vec2 uv,p=fragCoord.xy/r;
        uv=p;
        p-=.5;
        p.x*=r.x/r.y;
        z+=.07;
        l=length(p);
        uv+=p/l*(sin(z)+1.)*abs(sin(l*9.-z*2.));
        c[i]=.01/length(abs(mod(uv,1.)-.5));
    }
    fragColor=vec4(c/l,t);
}
```

GMTC
全球大数据技术大会

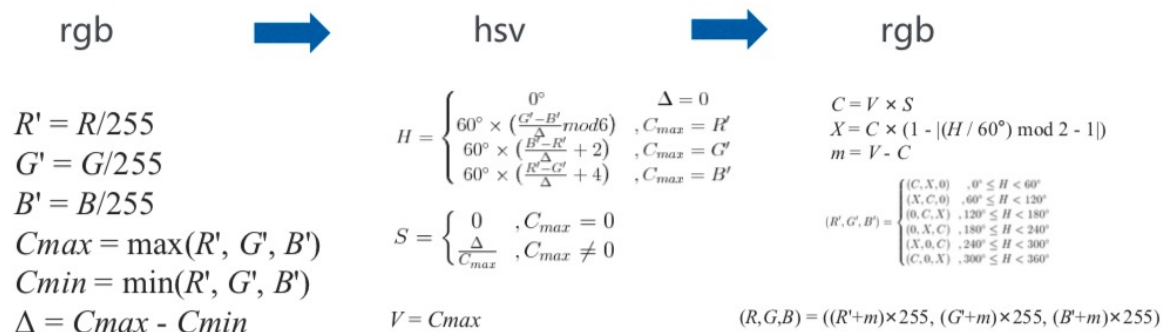
Geekbang InfoQ
极客学院

这个代码也惊人的简单，我们也不需要把这个东西搞的特别清楚，你知道理解我们可以有这些灵感的来源，就足够了。

相变

相变又是来自于一个新的领域的知识，就是我们可以对图片做一个处理，大家看到这个小人有帽子，作为一个有追求的前端程序员，我们可以用代码去改变它帽子的颜色。这里面涉及一个颜色的知识，叫做 hsv。

相变



一般来说，大家只把 hsv 当作一种写颜色的方式，但我认为它是一个比 rgb 语义更好的颜色表述的方式，hsv 是用了一个色相和明度，和纯度这样的概念，我们如果想改变一张图的色相，我们就只需要去改变它的色相。

我们把蓝色的色相变到绿色的色相就 OK 了，这里面有一个很复杂的公式，只是写起来有点吓人，其实都是加加减减就好了，我们在 hsv 完成一个相变，我们再把它转回 rgb，这样就实现了我们的这个色相变化的效果。

我们可以把这一点利用到很多场景上，比如说人民币，你从 5 元到 100 元钱，虽然大家觉得它差异很大，但它其实就是一个色相的改变。（我在前面的文章中讲到了同一种鸟颜色的转变，也是如此。）

绿幕

绿幕



GMTC
全球大数据技术大会

赞助商 **Geekbang** **InfoQ**
极客公园 技术大会

绿幕是电影的技术。如果大家到拍摄现场看拍电影，你会发现他们经常弄一个绿幕在上面，我们也用了类似的技术，我们也管它叫做绿幕。

3D 图形

3D图形

- threejs & babylonjs



GMTC
全球大数据技术大会

Geekbang
InfoQ

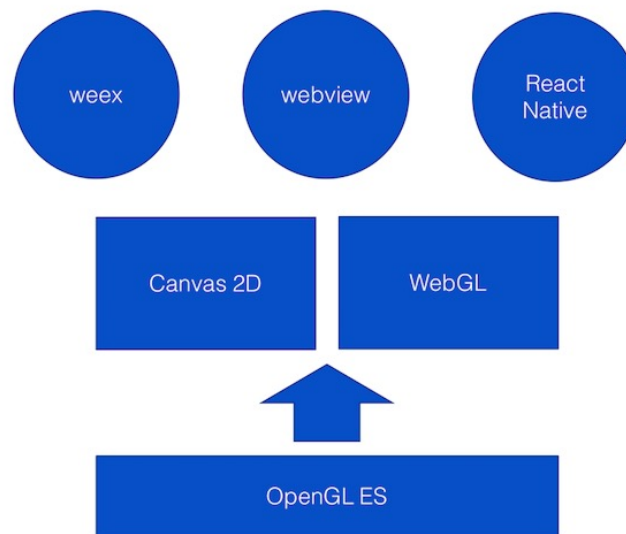
最后提一下 3D 图形，因为这个是业内非常成熟的东西，我就不仔细讲了。就是我们的 ThreeJS 和 BabylonJS 提供的 3D 的效果。3D 的领域，现在是个红海竞争，你写一个引擎基本上跟别的引擎差不多，你有的功能别人都有，除非你与一个很厉害的实验室合作做了一些特殊的优化，但是，我觉得对于工程团队来说，这个代价有点高。

图形学的基础设施

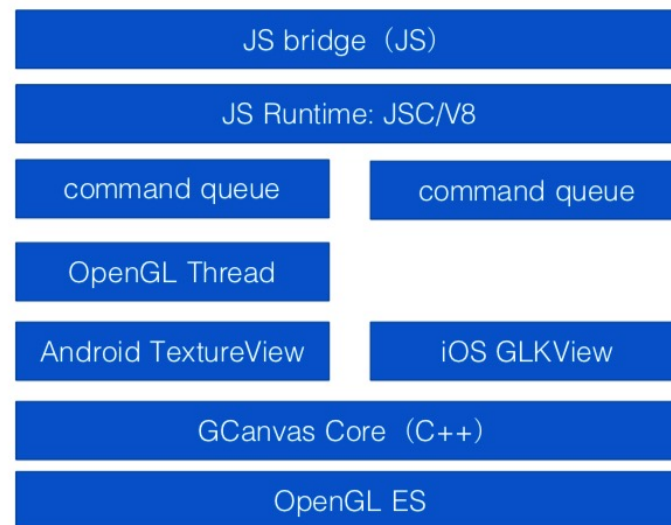
最后讲一下图形学的基础设施，我们做图形的事情，还有些比较现实的问题。

基础设施：GCanvas

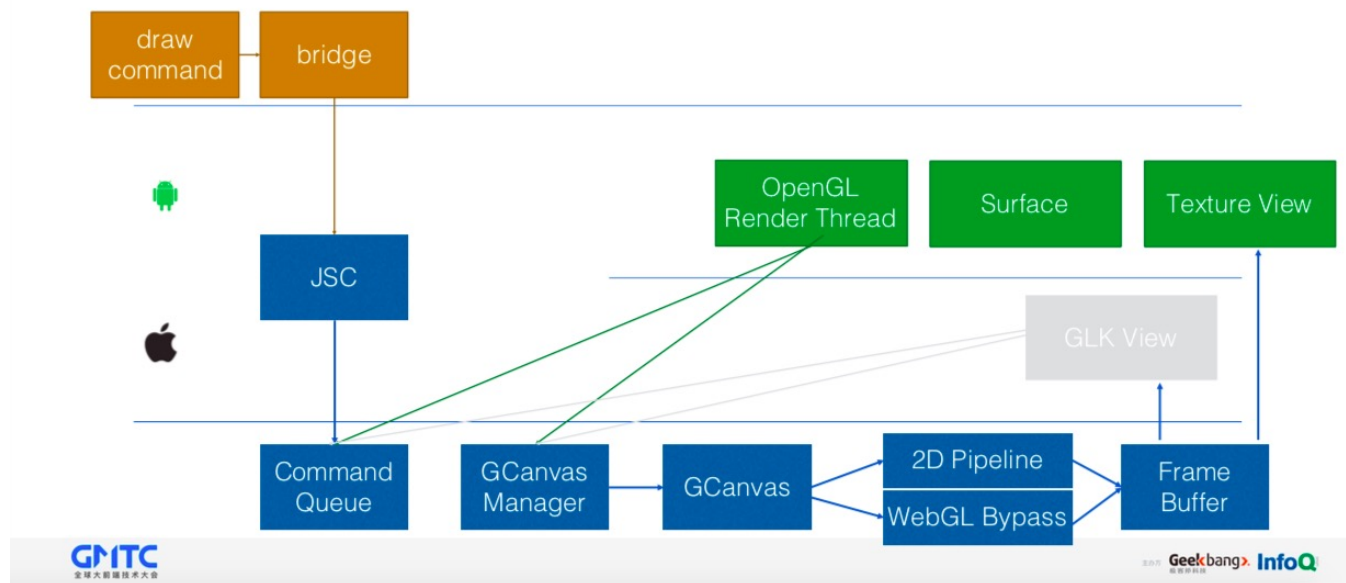
基础设施——GCanvas



基础设施——GCanvas



基础设施——GCanvas



比如说阿里巴巴现在已经不用 web view，基本上淘宝里面的页面已经是百分之百 weex 化了，可能就有一两个页面不是，我们面对的一个非常现实的问题，就是我们在用 weex 技术，而它里面是没有 Canvas 的，如果大家没有用 weex，用了 React Native 其实也要面临一样的问题。

没有 Canvas 怎么办？其实还是很简单的道理，做一个，所以我们做了一个叫做 GCanvas 的东西。

基础设施：G3D

基础设施——G3D



业界还有一个东西就是 G3D，它与 ThreeJS 一类，没有什么本质的区别。

首先底层它会有些管理的能力，它也可以交互，我们也做了什么点选，拖拽、顶点变形这样的能力。值得一提的是，我们做了 PBR，也就是光线追踪，相对来说，PBR 也是一个比较高等级的引擎了。

如果你对今天的内容有什么样的想法，你可以给我留言，我们一起分享。



重学前端

每天 10 分钟，重构你的前端知识体系

winter 程劭非
前手机淘宝前端负责人



新版升级：点击「👤请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。



bd2star

精选留言(7)

•



Frojan

应用场景还是有限，在大公司可能有机会接触对应的项目或基础设施建设，在小一点的公司则很难接触到。作为加分项尚可，作为主攻方向可能比较吃亏。如果要在node和图形学做选择的话可能node全栈更好找工作。作为一个喜欢图形学的前端，希望图形学在前端的应用能越来越多，越来越丰富。将来有一天也会有更多

前端图形学工程师的岗位哈哈，winter老师您说会有这么一天吗？☺

□

8

- Sentry

最近出于好奇，clone了github上chrome的源码仓库，发现竟然有12G多，貌似比linux内核的源码还多。个人特别想探索一下浏览器源码，但面对如此庞大的代码，不禁望而生畏，也不知从何下手。请问老师，浏览器内核源码该如何去研究，skia渲染引擎是最先进吗，svg，canvas，WebGL该如何选择，怎样深入地学习和掌握它们呢？望老师拨冗答疑，谢谢！

□

1

- 麦冬

请问下老师，为什么flexible布局方案不再维护了呢？这个方案本身存在问题吗？

□

2019-05-05

-



許敲敲

有的工业软件，3d建模的也可以用浏览器来实现了，用的是ThreeJS WebGL，老师了解这些嘛？怎么看待这个发展前景

□

2019-05-03

-



□

2019-05-03

-



□

2019-05-03

