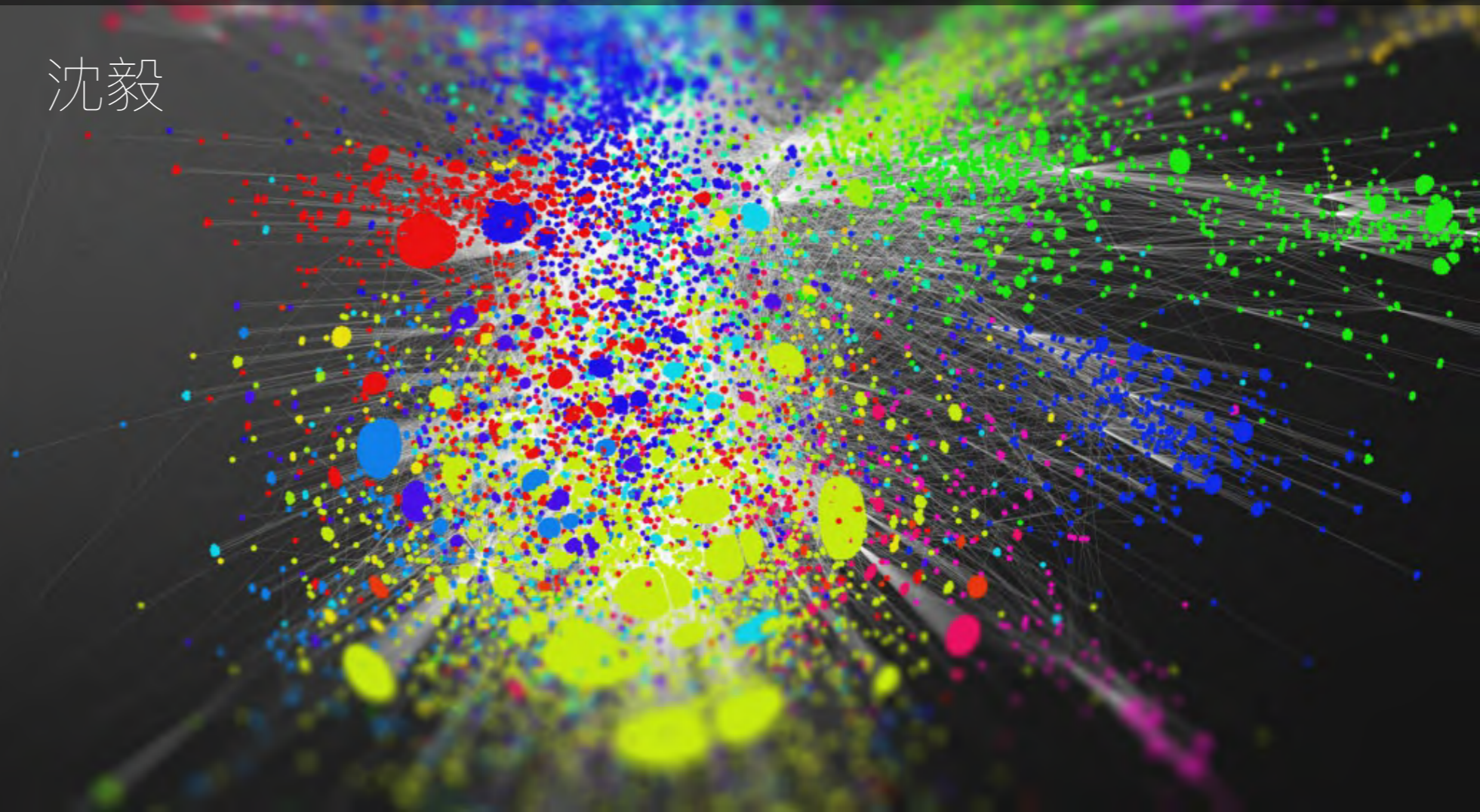


# WEBGL 在数据可视化中的实践

沈毅





# ABOUT ME

- **WebGL** 2011
- **Canvas** 2013
- **ECharts** 2014
- **qtek** 2013



# 大纲

- ECharts 简介
- 为什么选择 Canvas 及 Canvas 的限制
- WebGL 与 ECharts 的结合
  - 三维空间的可视化
  - 利用 WebGL 加速力引导布局
  - 在前端实现高品质的渲染



# ECHARTS 是什么

- 拥有 17k star 的开源前端可视化库
- 声明式的编程接口
- 丰富的可视化类型和交互方式
- 大数据量展现的能力
- 吸引眼球的动画和特效



# 为什么选择 CANVAS?

- 更灵活的性能优化
- 像素操作的能力
- 能够和 WebGL 更好的结合



# 越来越复杂的需求

- 我要三维图表
- 我要显示几十万的数据
- 我要一秒内能够完成关系图布局
- 我要大屏上酷炫的特效



# 越来越力不从心的 CANVAS

- 画路径还是矢量的方式
- 只能“软渲染”三维图形





新世界的大门



# WEBGL 能够带来什么

- 三维场景的绘制
- 二维绘制的性能提升
- GPU 通用计算(GPGPU)
- 更加酷炫的效果



能力越大

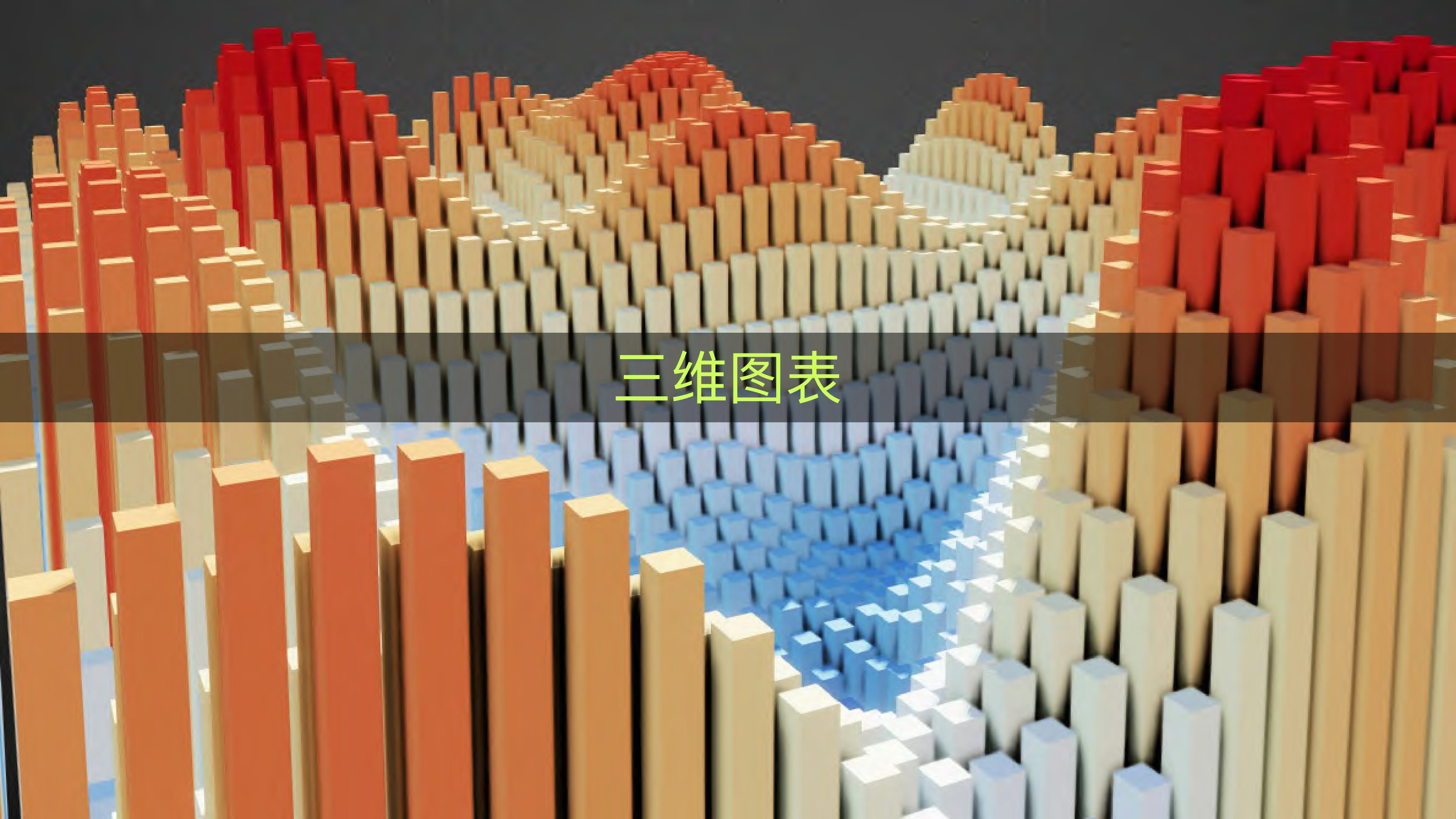
坑越大



# WEBGL 的使用和坑

- 三维图表的绘制
- 利用 WebGL 加速力引导布局
- 前端实现高品质的渲染



A 3D bar chart visualization with a color gradient from red to blue. The bars are arranged in a grid, with the height of each bar representing a data value. The chart is overlaid with a semi-transparent dark grey banner containing the text '三维图表' in a light green font.

# 三维图表



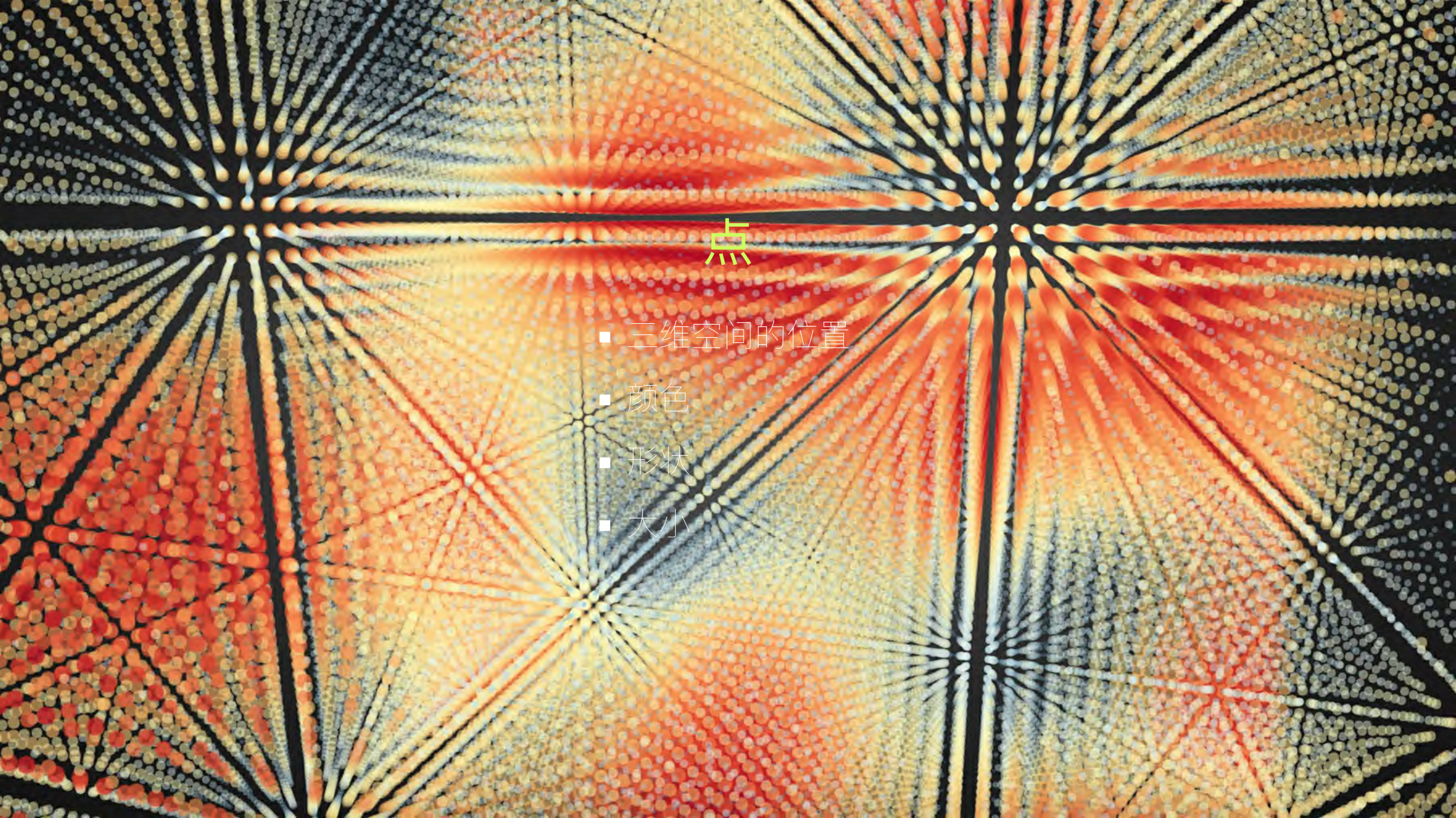
总结起来就是

点·线·面



画好三维的点线面





## 点

- 三维空间的位置
- 颜色
- 形状
- 大小



## JAVASCRIPT

```
gl.drawArrays(gl.POINTS, 0, 100);
```

## VERTEX

```
attribute float size;
```

```
gl_PointSize = size;
```

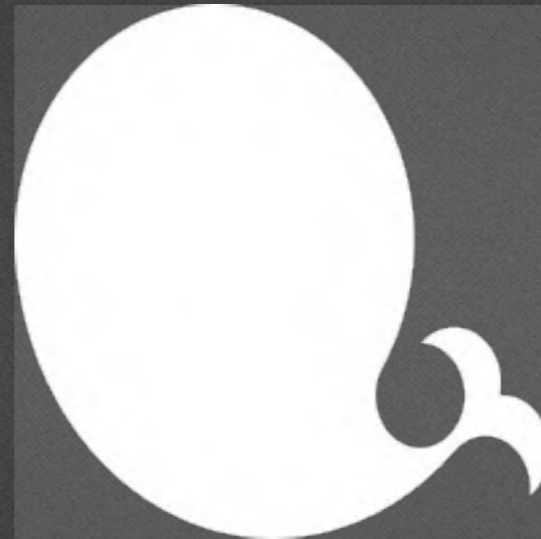
## FRAGMENT

```
gl_FragColor = vec4(1.0);
```



# 不同形状的点

- 把形状用白色填充到 Canvas 上作为纹理



```
gl_FragColor = color * texture2D(sprite, gl_PointCoord);
```



# 描边?

- 画轮廓线
- 单独再创建一张描边的纹理
- 单纹理中描边和填充用颜色区分
- Signed Distance Field



# SIGNED DISTANCE FIELD (SDF)

- 存储到最近的图像边缘的距离
- Shader 中根据这个距离填色

```
float d = texture2D(sprite, gl_PointCoord).r;  
// Antialias  
gl_FragColor.a *= smoothstep(0.5 - smoothing, 0.5 + smoothing, d);
```



# 优势

- 存储空间小，放大后也有清晰的边缘
- 开销小
- 能实现外发光，投影



A globe of the Earth is shown against a dark background. Overlaid on the globe is a dense network of glowing blue lines. These lines represent various types of lines, such as flight paths, network connections, or geographical features. The lines are most concentrated in the upper right quadrant of the globe, where they form a complex web. The globe itself is rendered in a dark, almost black color, with some lighter areas representing landmasses and oceans.

# 线

- 折线
- 飞线
- 轮廓线
- 网格线



# 原生态画线

```
gl.lineWidth(5);  
gl.drawArrays(gl.LINES, 0, 100);
```

- gl.LINES
- gl.LINE\_STRIP
- gl.LINE\_LOOP



但是

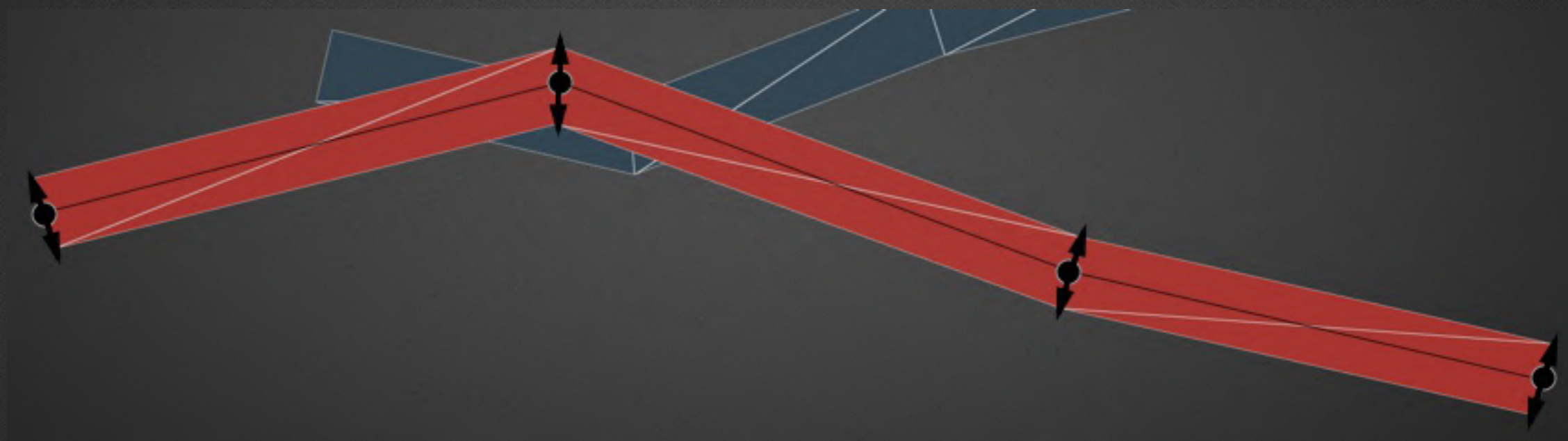


# 原生画线方法的各种坑

- 不同的显卡驱动下画线的效果会有细微区别
- 无法控制 `lineJoin` 和 `lineCap`
- 有最大线宽的限制，而且 Windows 下最大只有 1



# 三角化线段





# 实现屏幕空间固定宽度

```
vec2 dirA = normalize(currScreen - prevScreen);  
vec2 dirB = normalize(nextScreen - currScreen);  
vec2 tanget = normalize(dirA + dirB);  
  
len *= 1.0 / max(dot(tanget, dirA), 0.5);  
offset = tanget;  
  
offset = vec2(-offset.y, offset.x) * len;  
currScreen += offset;
```



# 面

- 三角面
- 程序生成



# GEO3D

- 将 GeoJSON 转成 Mesh
- Triangulation
- Extrude

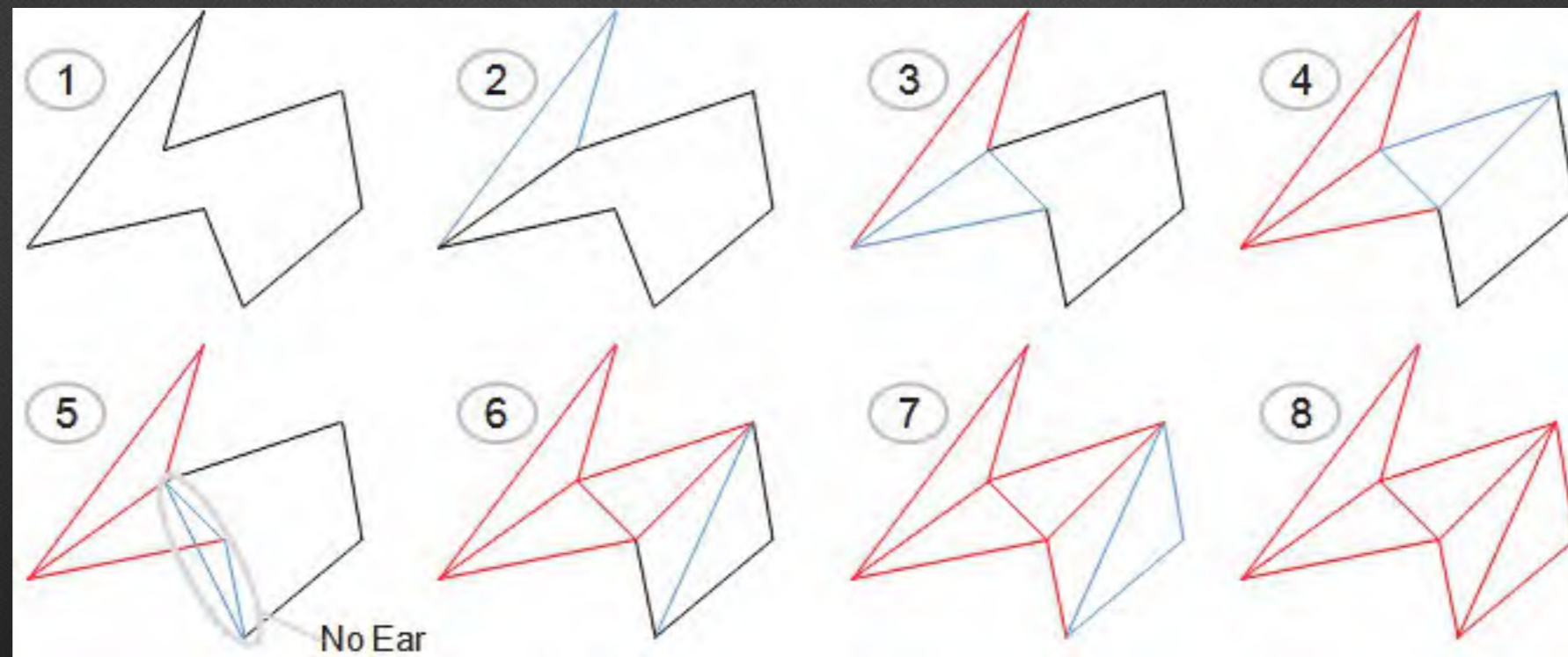






# TRIANGULATION - EAR CLIPPING

- 实现简单
- 可以利用空间哈希优化
- 使用链表存储顶点





# GPU 的通用计算

WebGL 中实现力引导布局



# 力引导布局介绍

- 用于关系图的布局
- 节点与节点之间模拟斥力，边模拟弹簧的引力
- 每次迭代  $O(n^2)$ , 需要上百次迭代才能结束

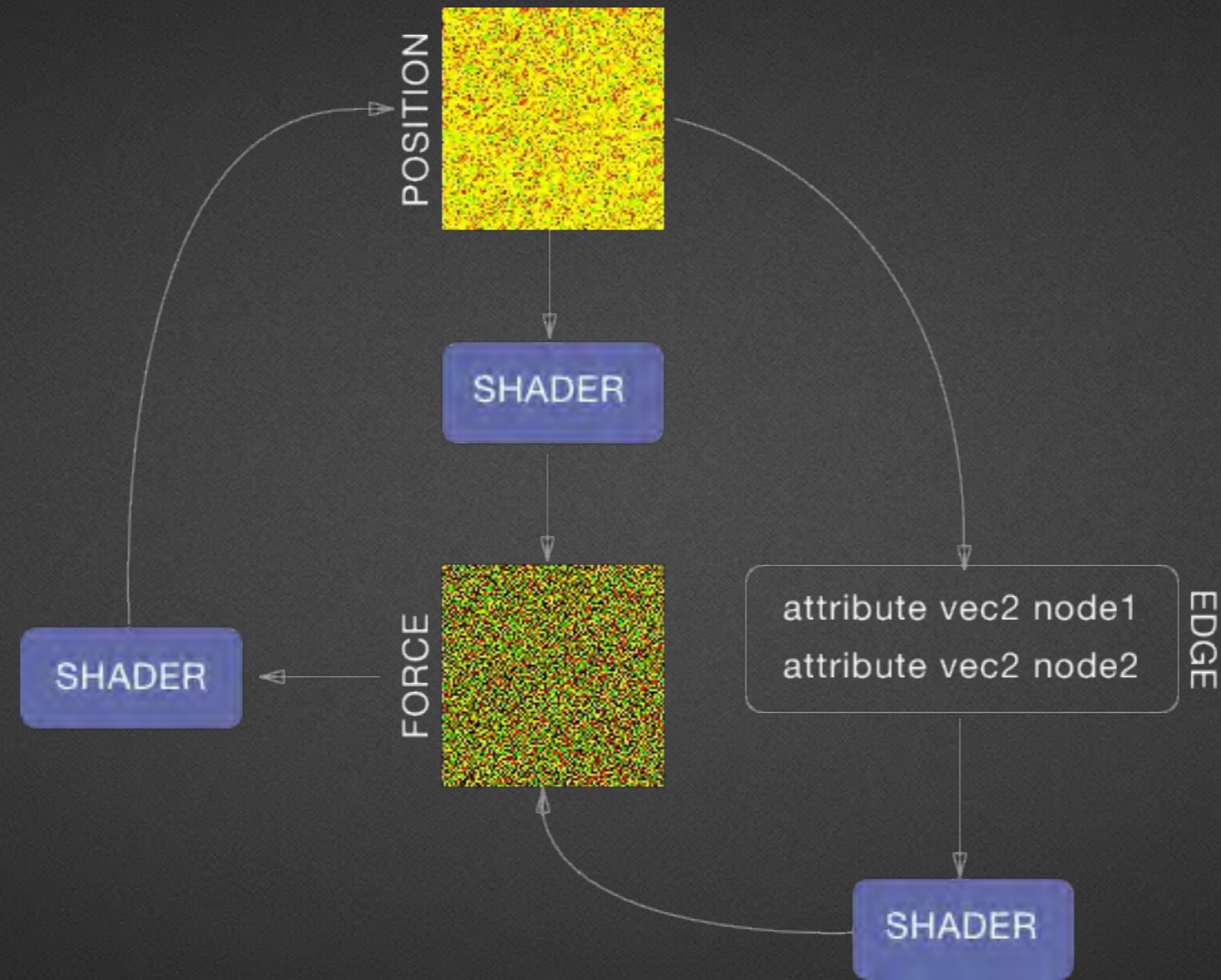


# 力引导布局的性能优化

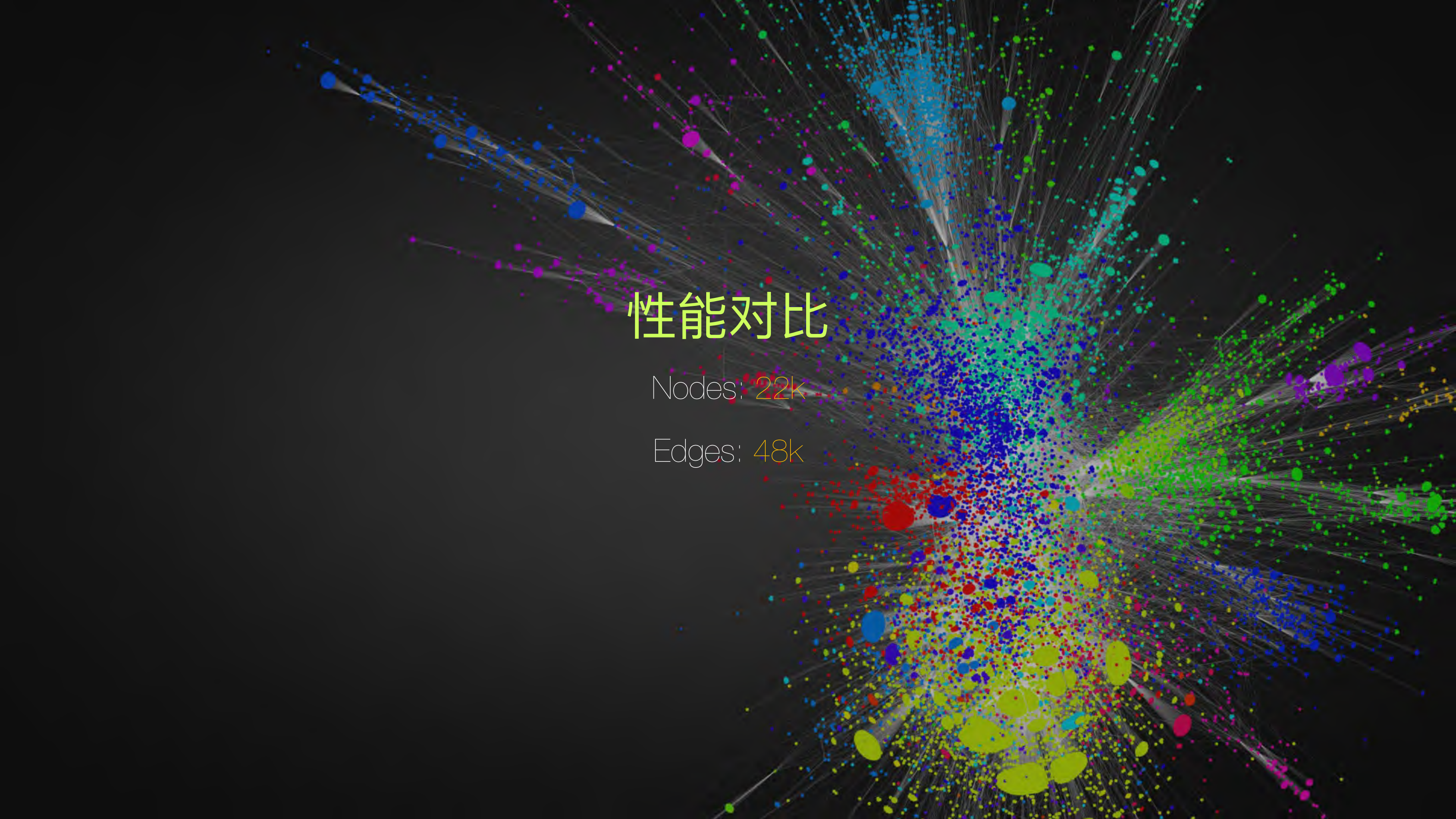
- Barnes Hut Simulation
- 多线程? Web Worker
- SIMD?



# WEBGL 中实现力引导布局







# 性能对比

Nodes: 22k

Edges: 48k



# 原论文

CPU without Barnes Hut: **~41000 ms**

CPU with Barnes Hut: **~400 ms**



Macbook 13 2012

CPU without Barnes Hut: **~28000 ms**

CPU with Barnes Hut: **~1000ms**

GPU: **~260ms**



GTX1070, i7

CPU without Barnes Hut: **~12000 ms**

CPU with Barnes Hut: **~300ms**

GPU: **~2ms**



A close-up, low-angle shot of a high-end NVIDIA GTX 1080 graphics card. The card is dark grey or black with a prominent cooling fan visible in the lower right corner. The text "GTX 1080" is embossed on the top edge of the card. The lighting is dramatic, highlighting the metallic textures and sharp angles of the card's design. The background is dark, making the card stand out.

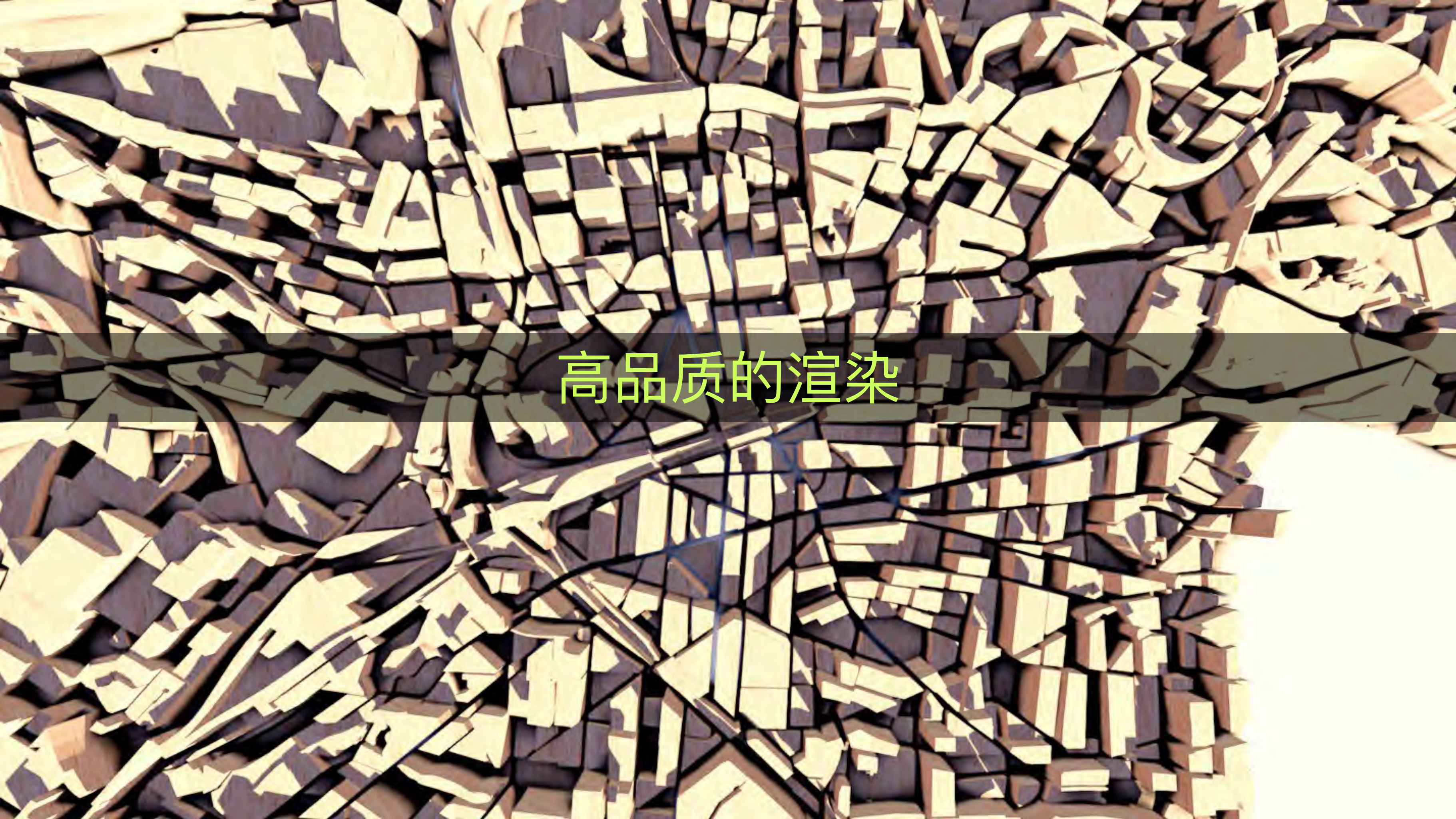
GPU 加速在高端显卡中的性能提升十分显著



# 限制

- 需要浏览器支持 WebGL
- 需要浮点纹理扩展
- 数据量特别大的时候容易造成整个系统阻塞

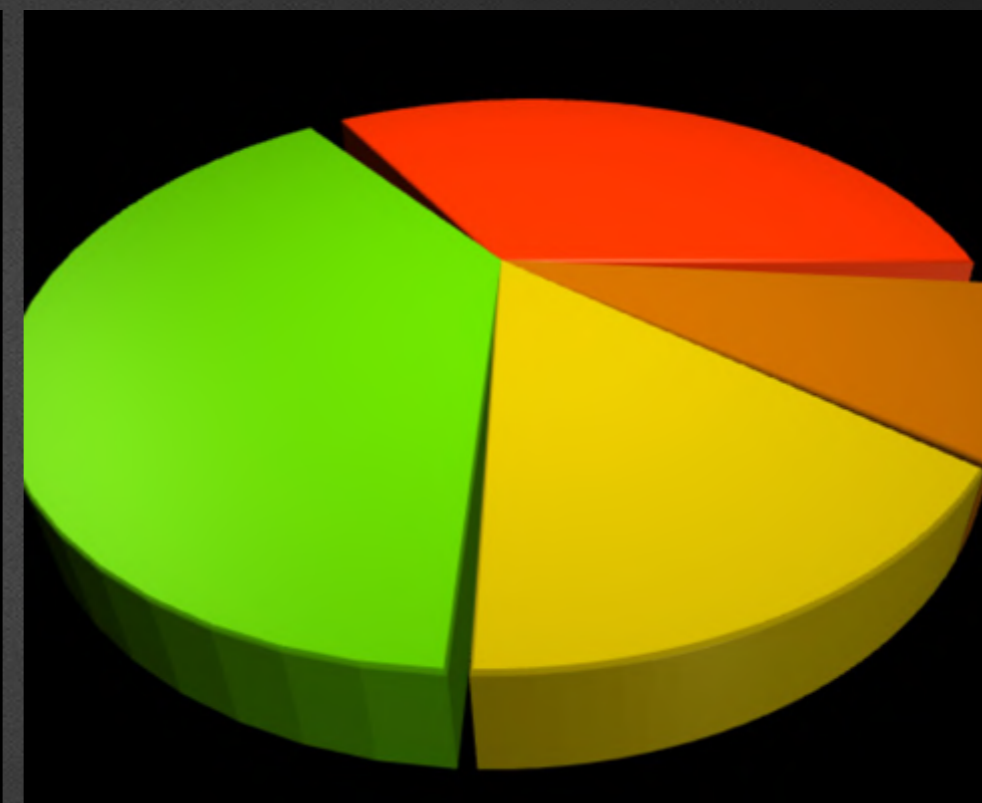
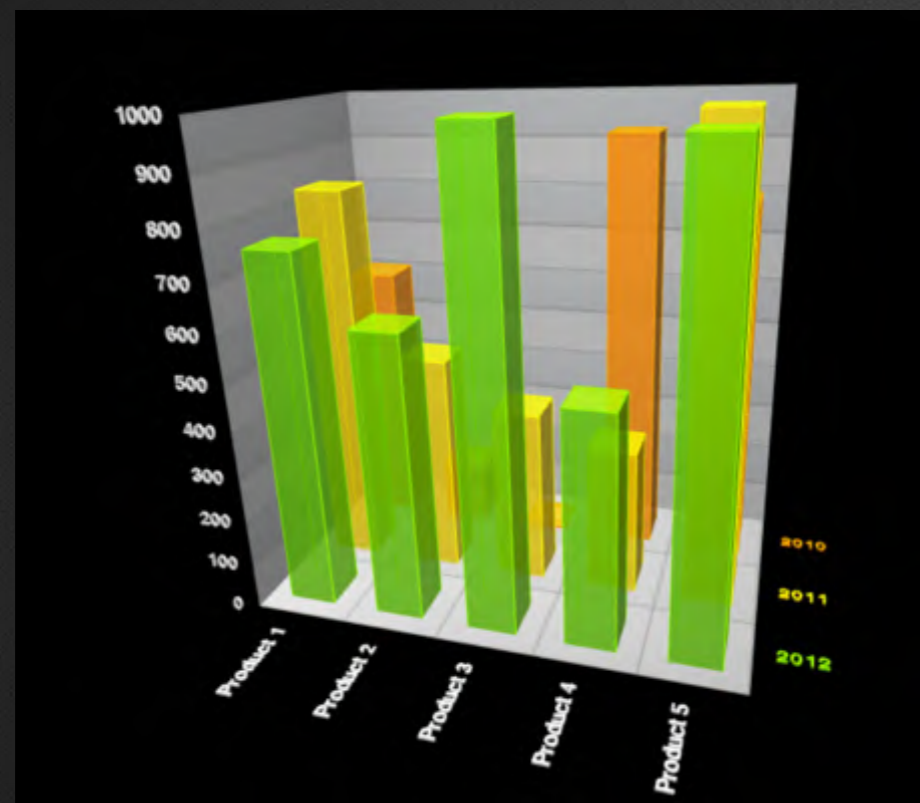


The image displays a high-quality 3D architectural rendering of a city street. The scene is composed of numerous rectangular buildings of varying heights and orientations, creating a complex, textured urban environment. The buildings are rendered in shades of light beige and tan, with dark shadows cast between them, emphasizing their three-dimensional form. A semi-transparent dark grey horizontal bar is positioned across the middle of the image, serving as a background for the text. The overall lighting is bright and directional, suggesting a sunny day, which creates strong highlights and deep shadows, contributing to the realism of the scene.

高品质的渲染



# 不要过时的三维效果



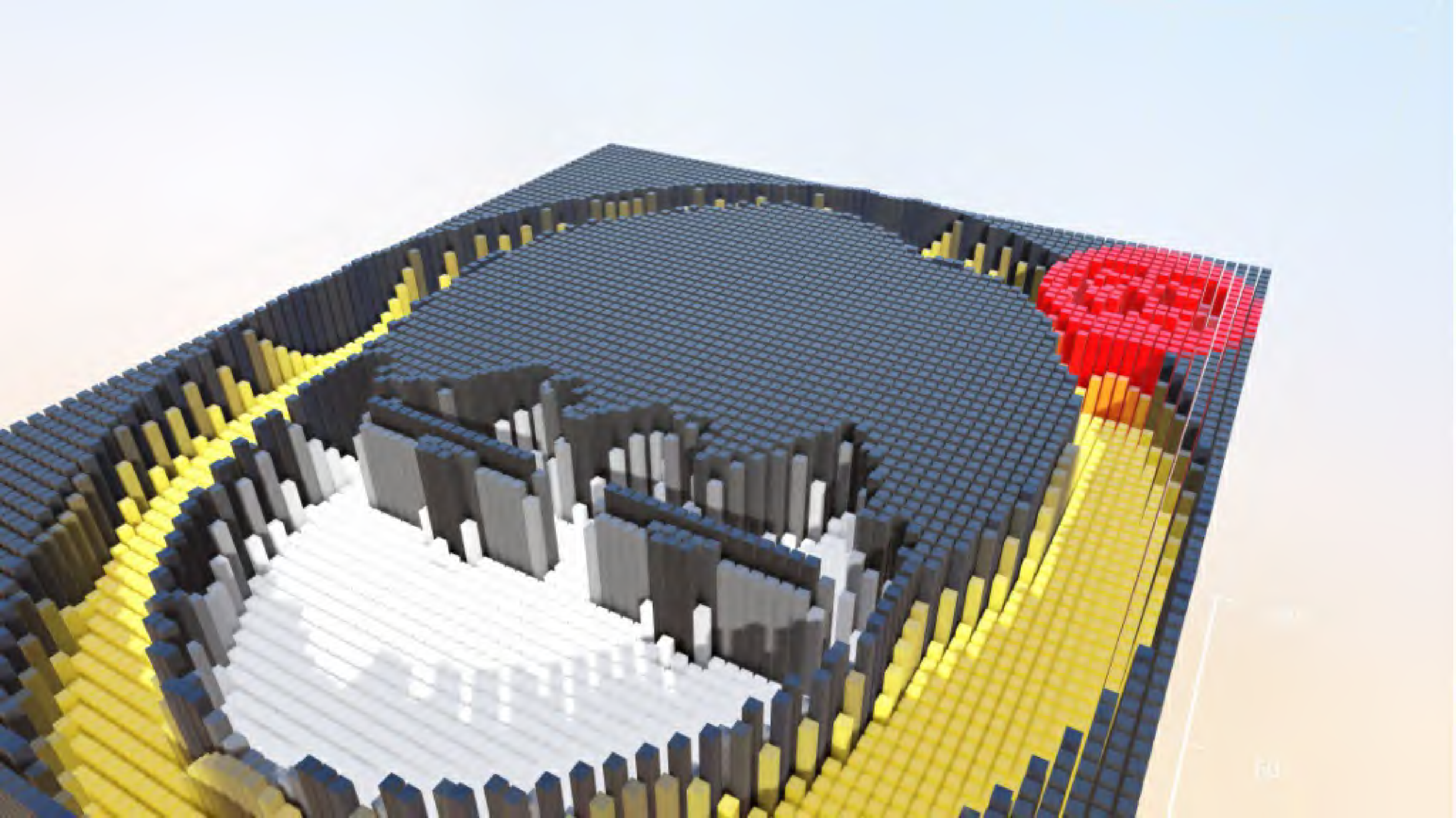


目标

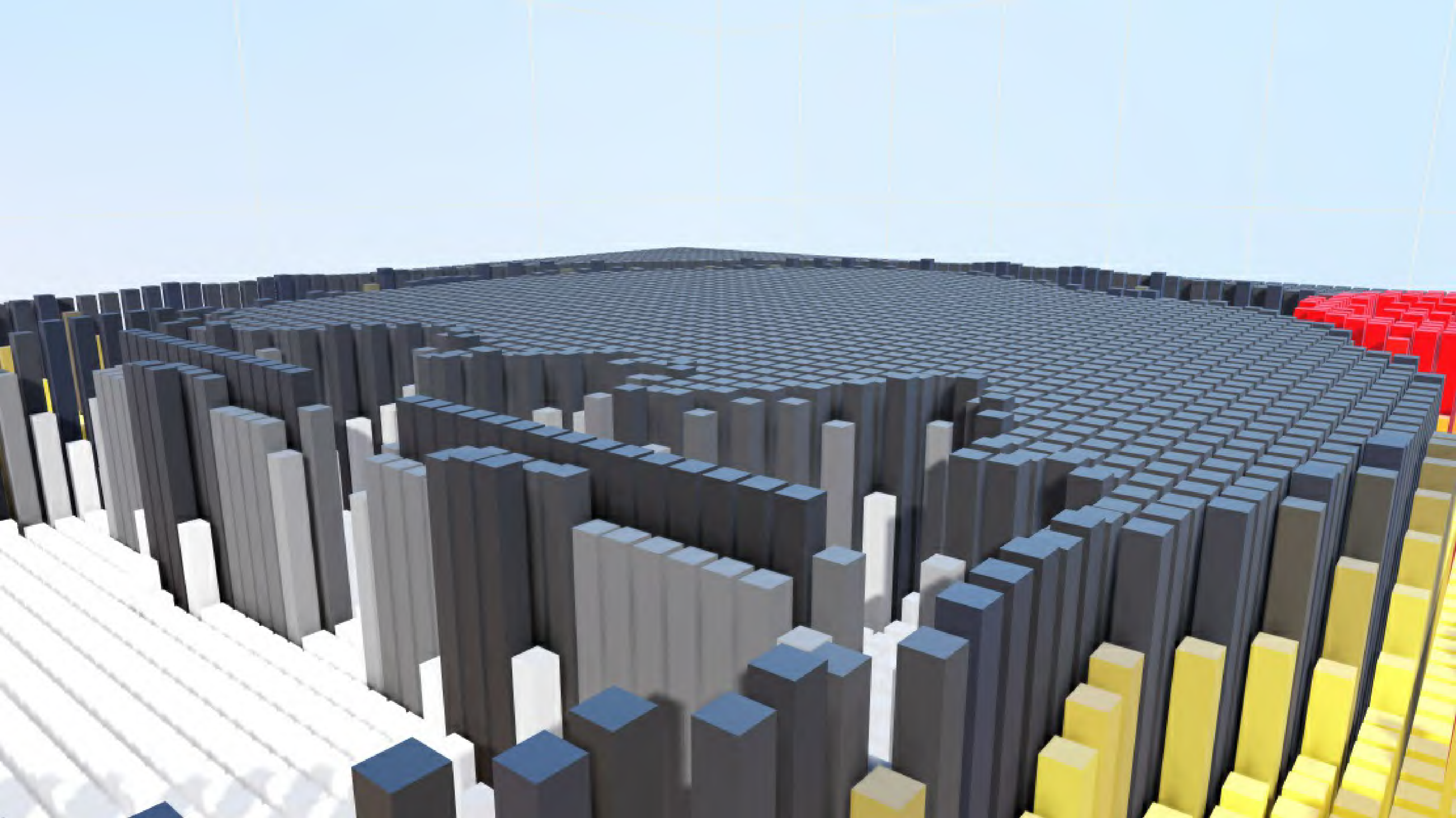
随手截一张就是桌面背景

PPT, 媒体写作

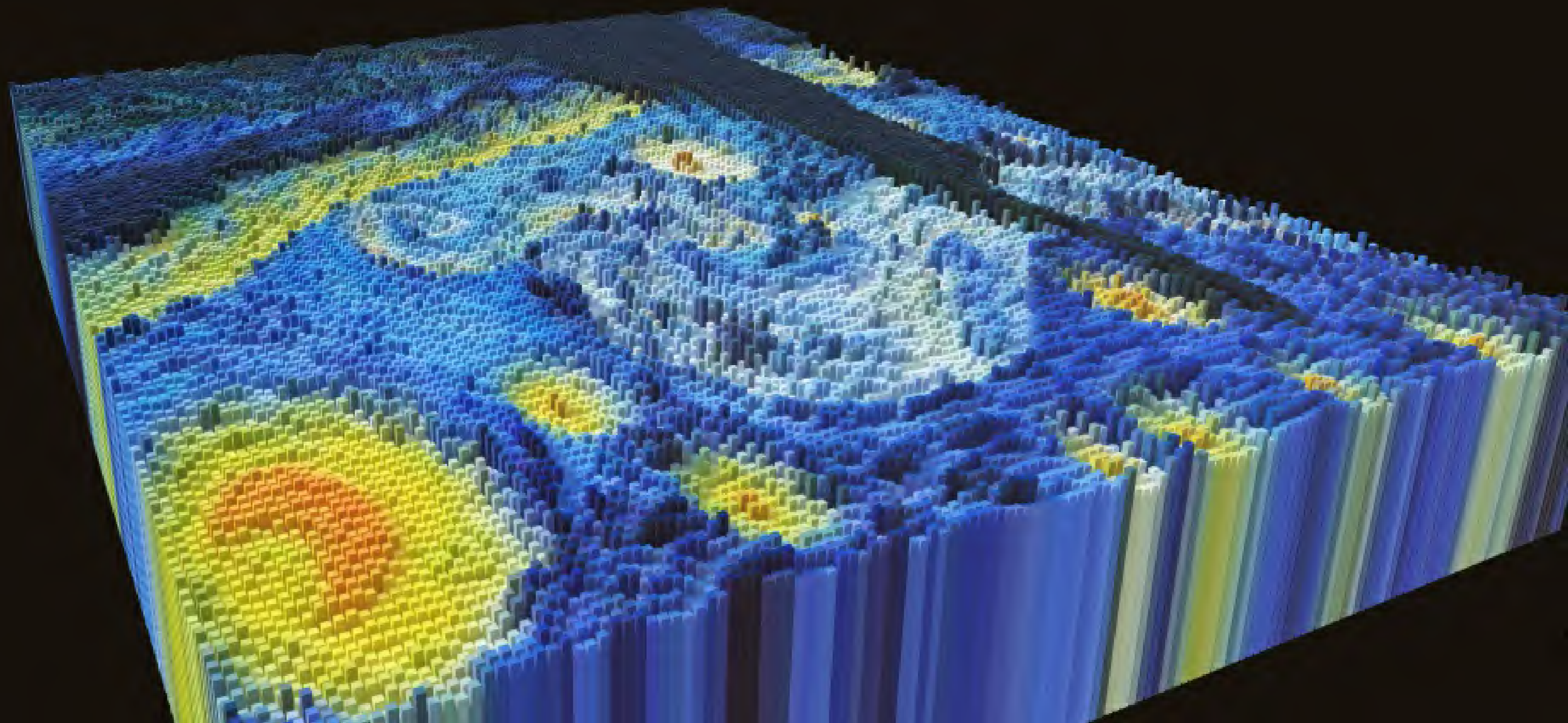




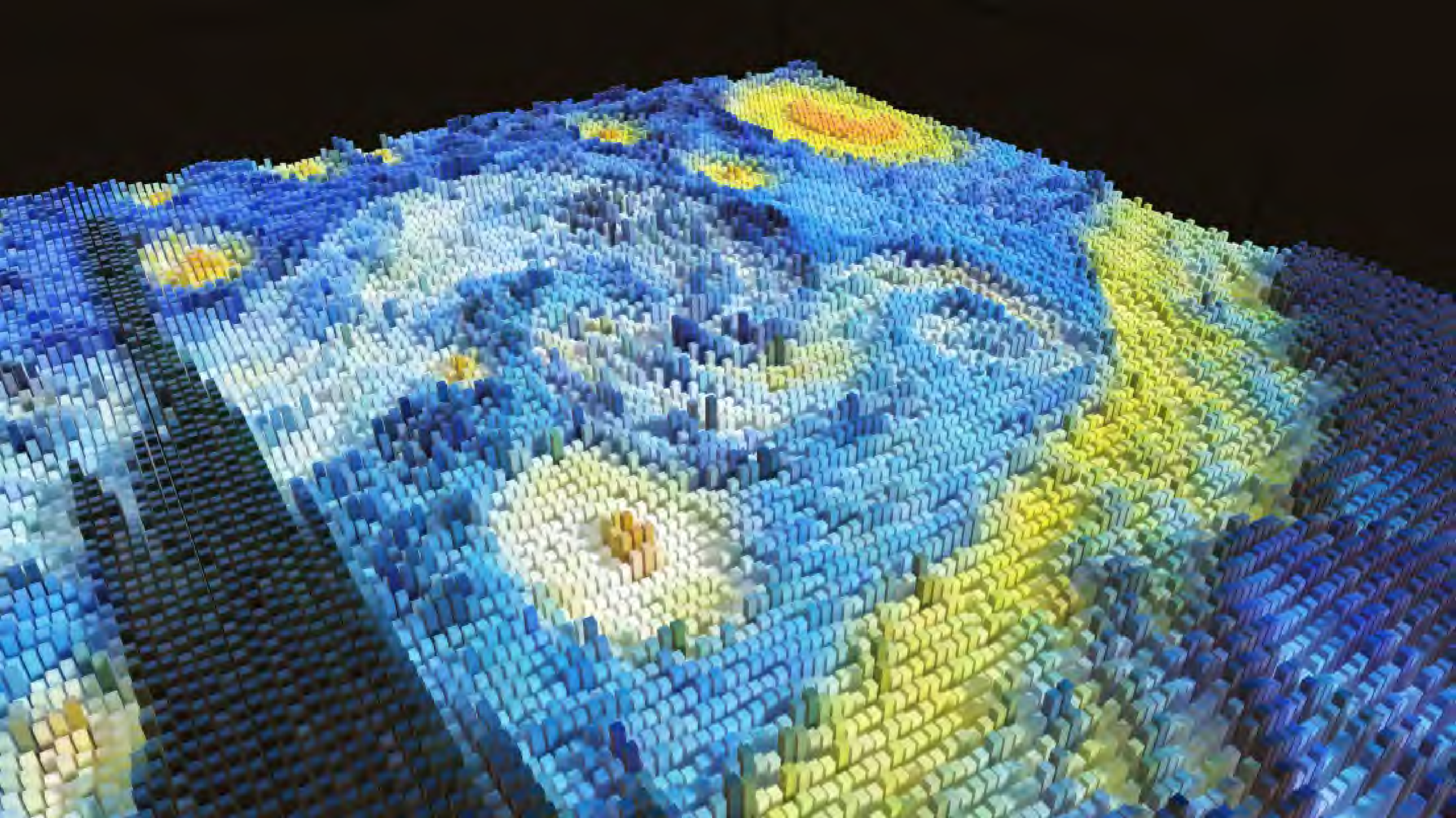










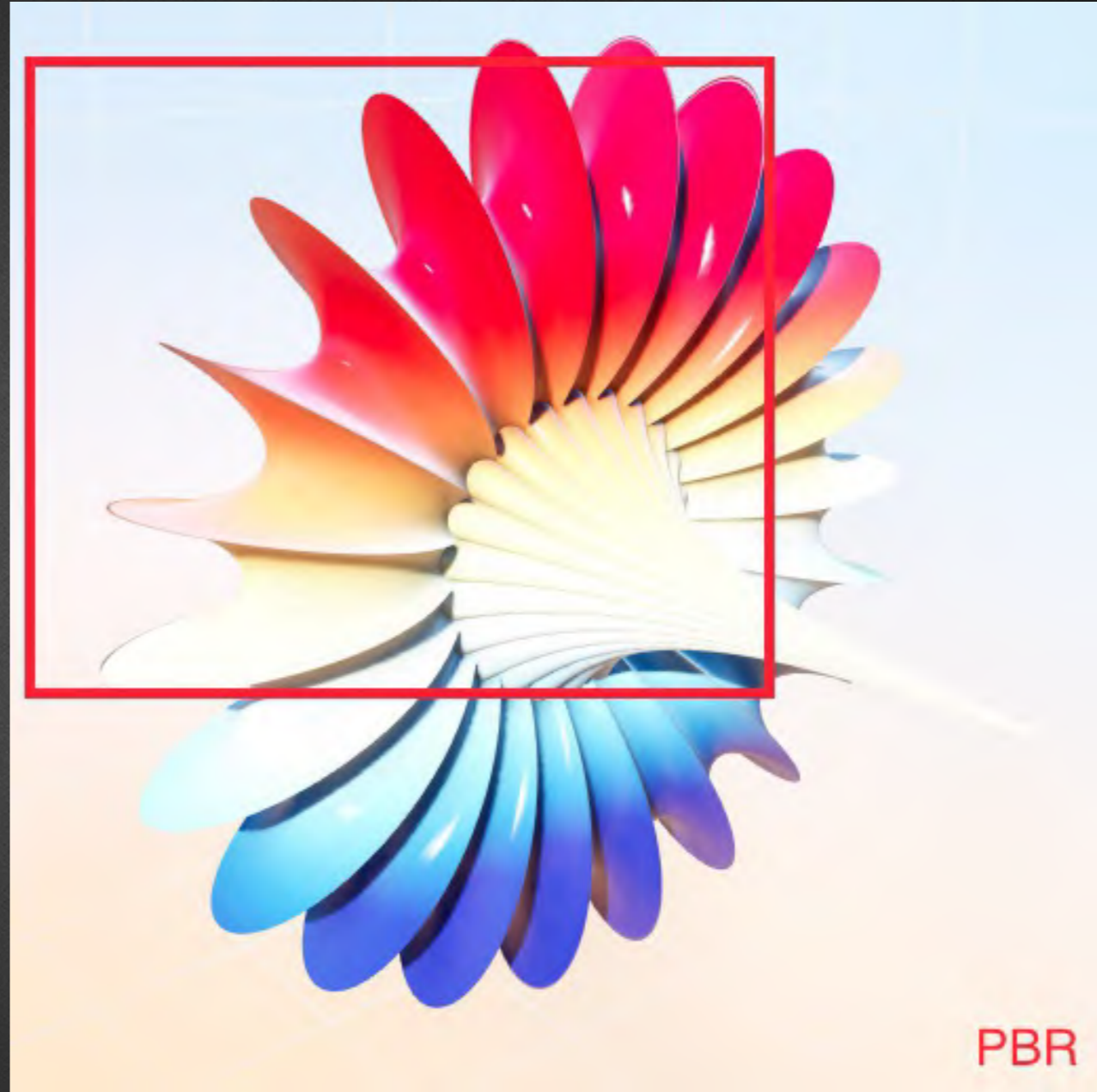
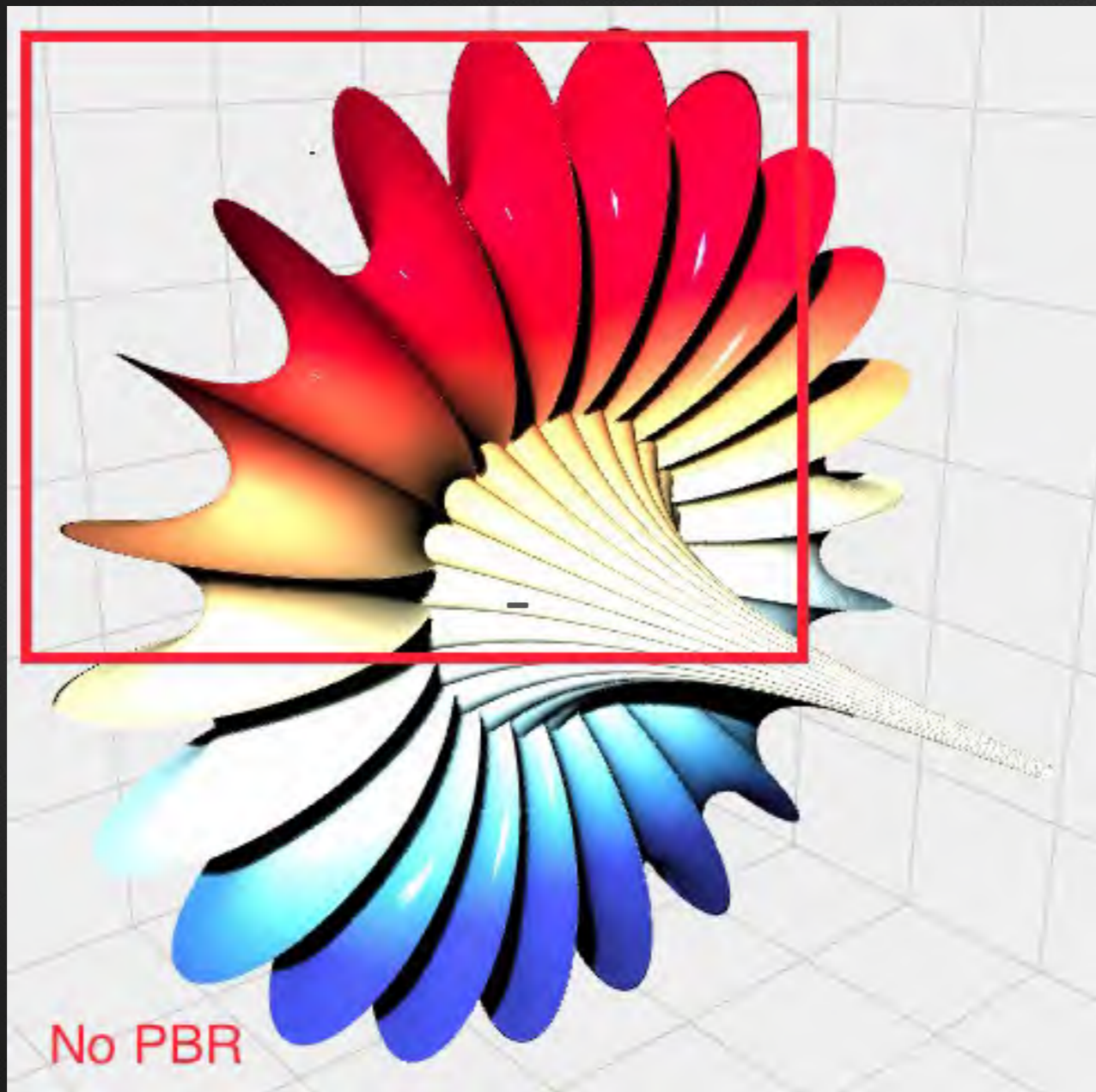




# 几个对画质提升比较大的技术



# 基于物理的渲染 (PBR)



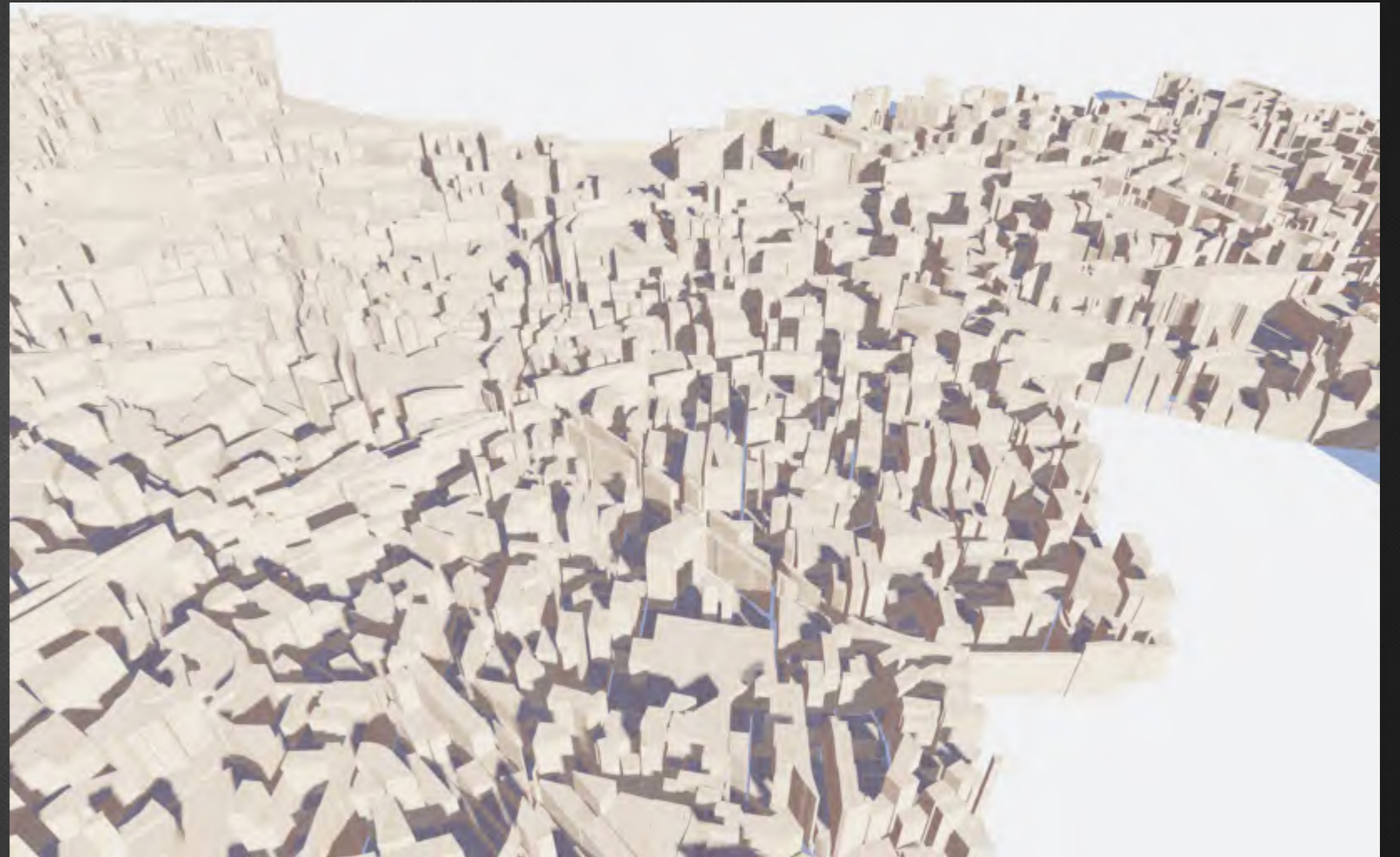
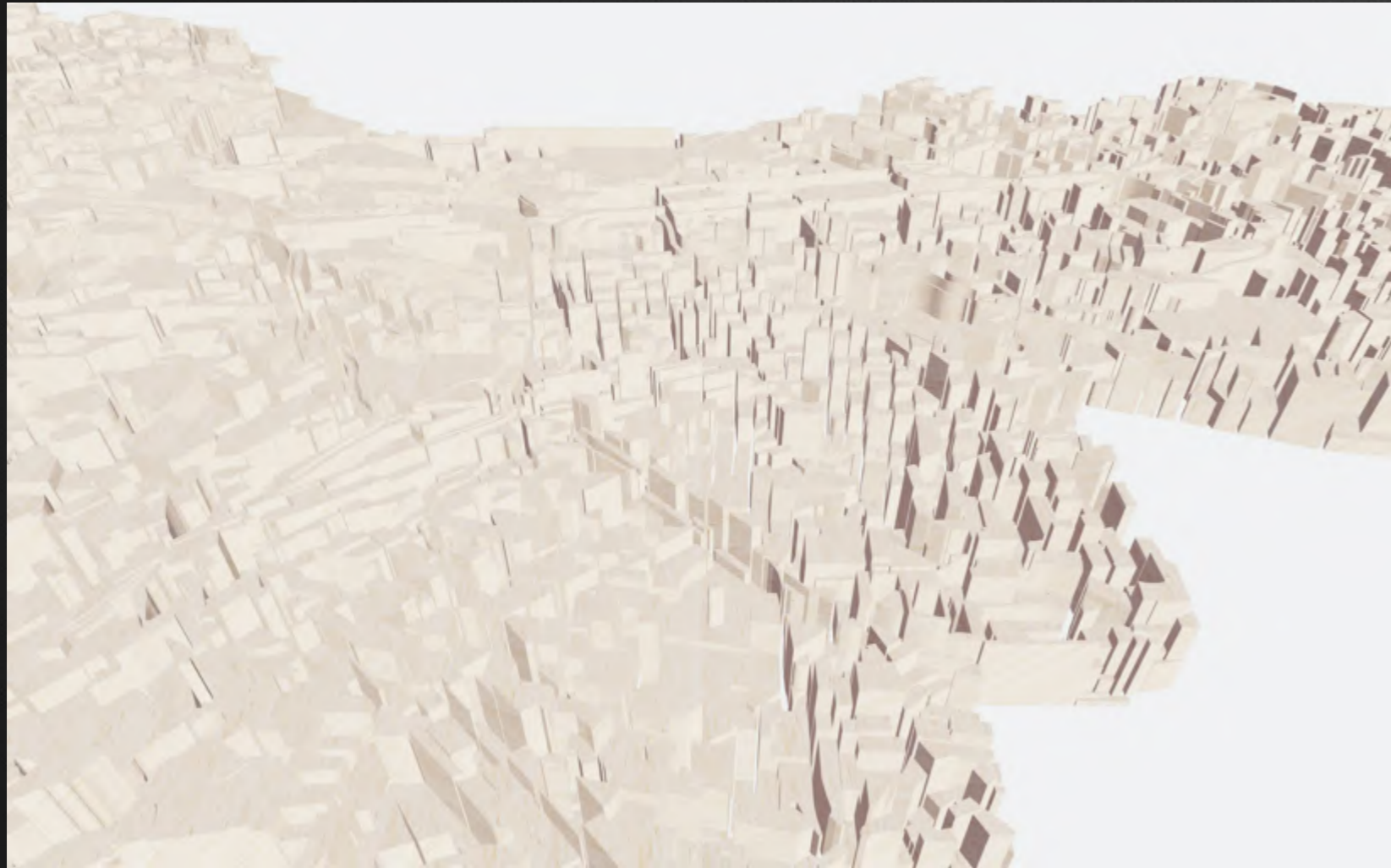


- HDR 的环境光照贴图
- 对环境光照的积分预计算 (prefilter)
- 能量守恒的光照公式
- 经验模型 直观的公式参数。



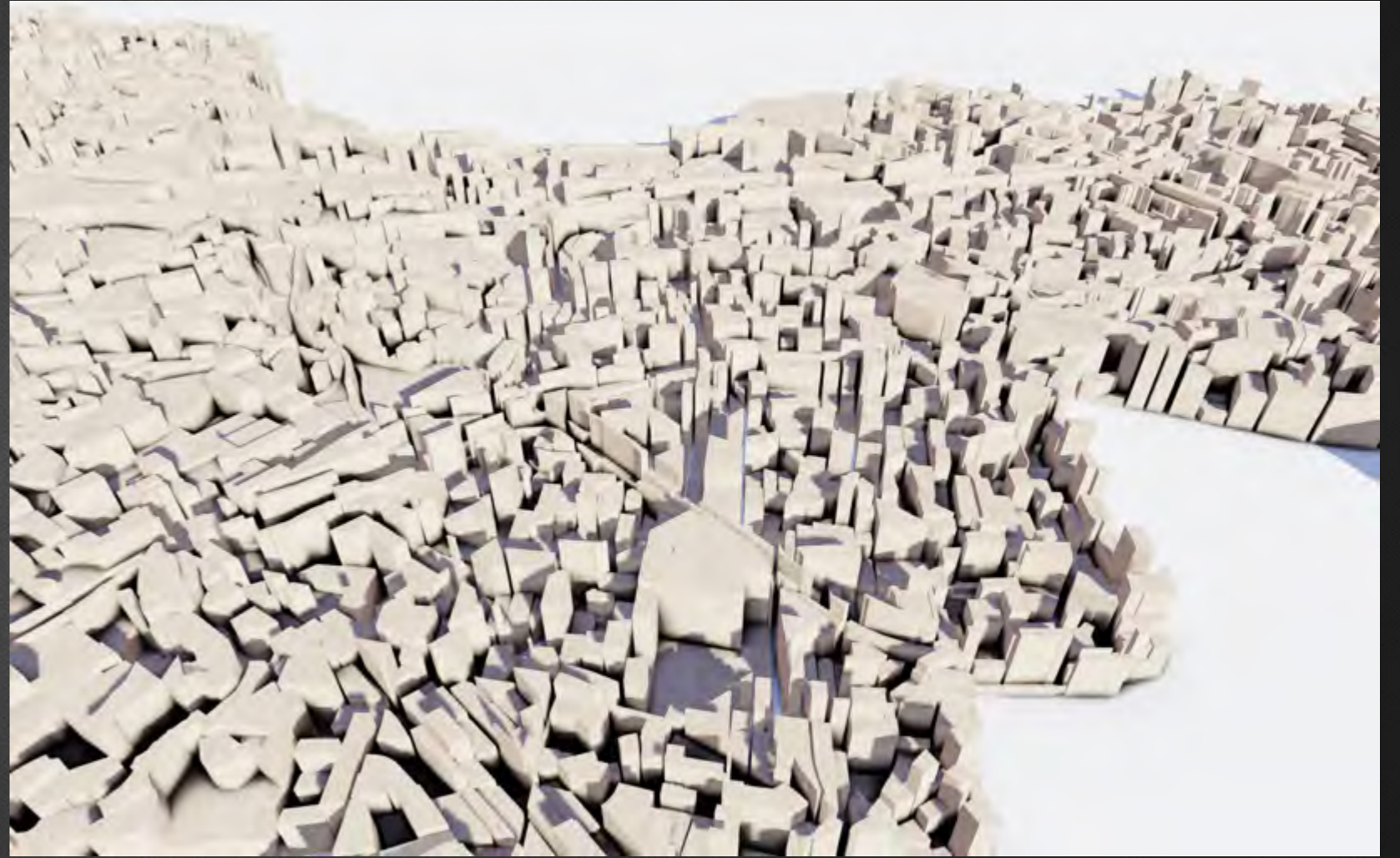
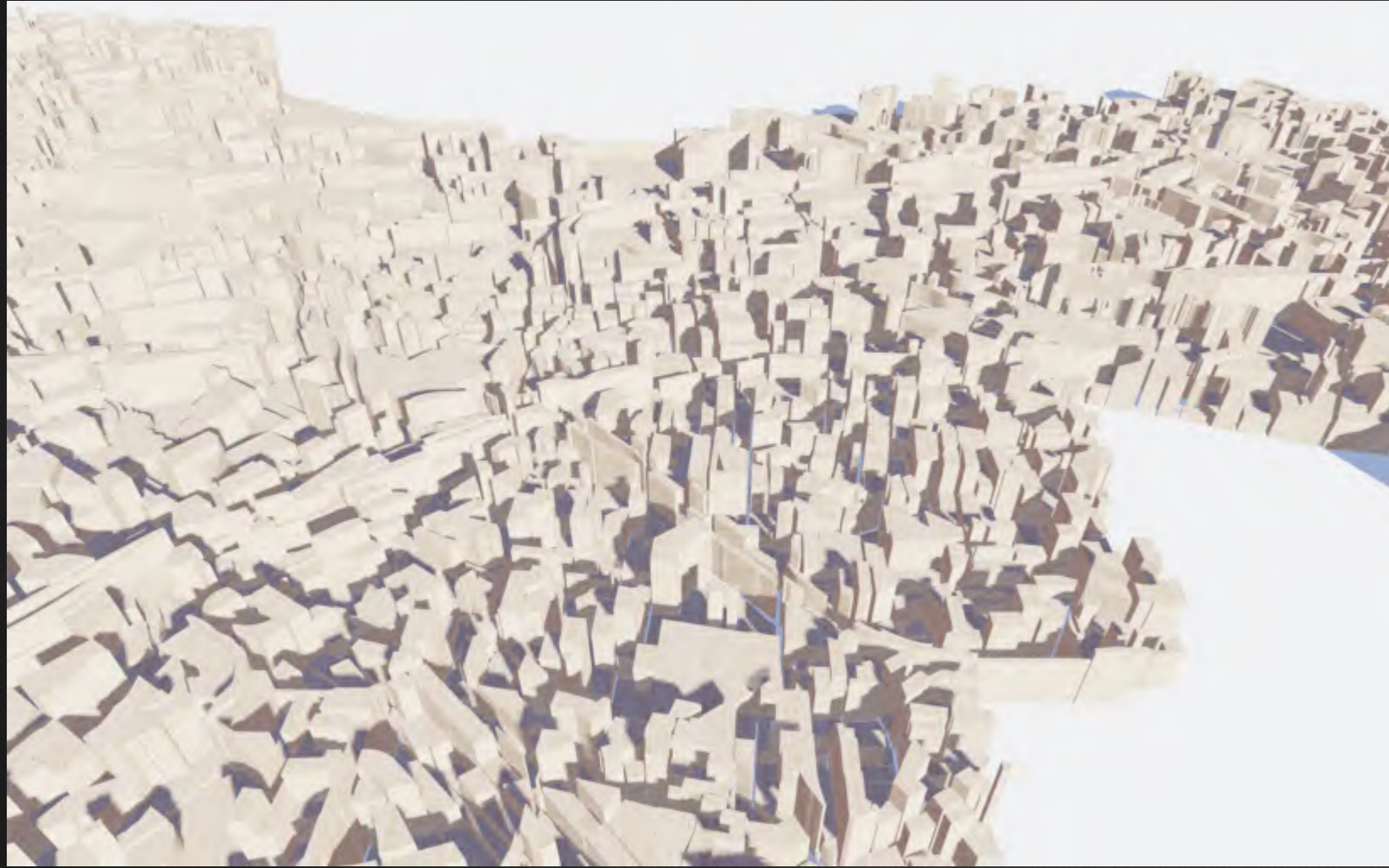


# 软阴影





# SSAO - 环境光遮蔽







景深



- American Airlines
- Air China
- China Southern Airlines
- Delta Air Lines
- Ryanair
- China Eastern Airlines
- easyJet
- United Airlines
- US Airways
- Southwest Airlines

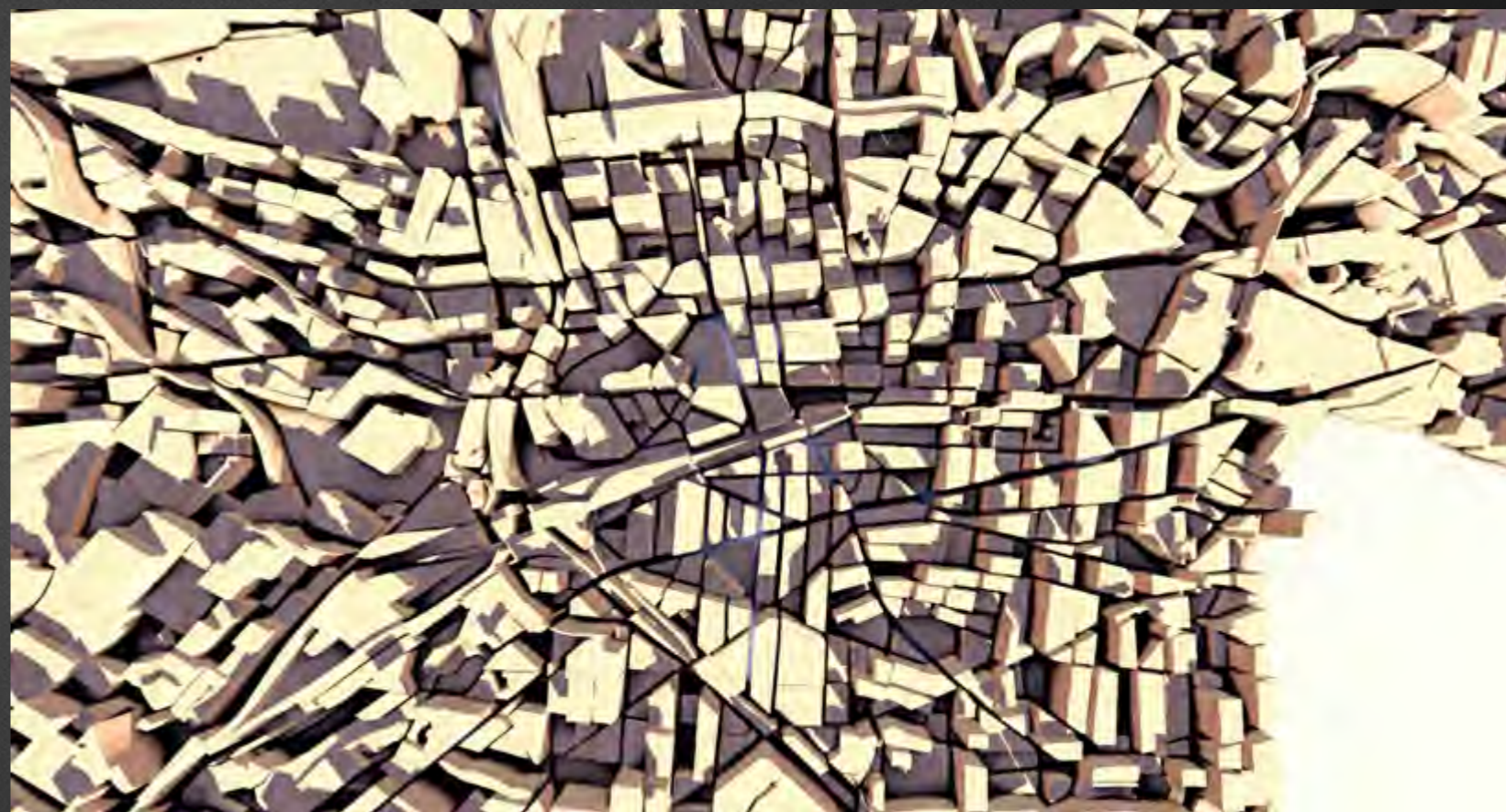
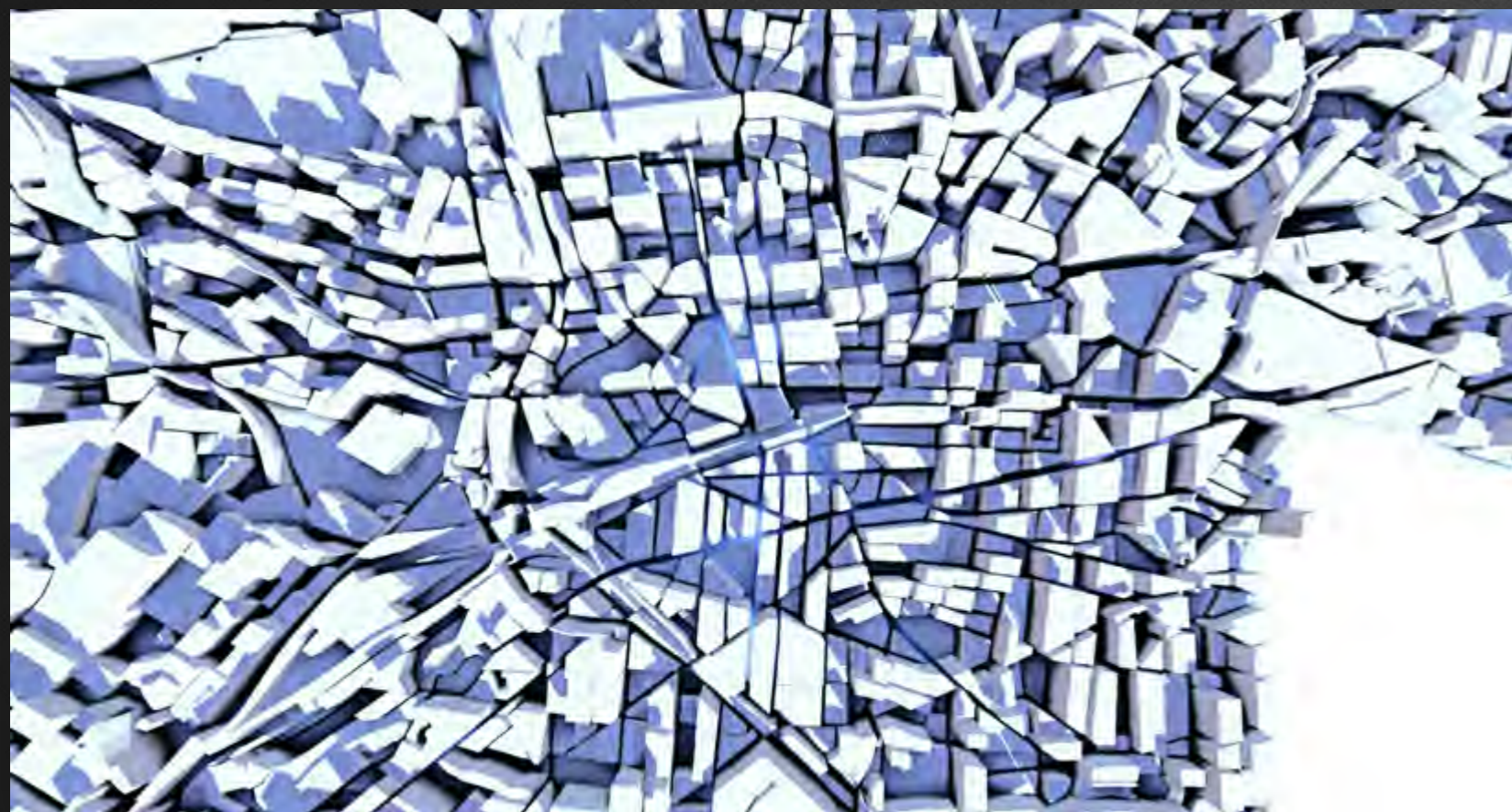


# 散景 BOKEH



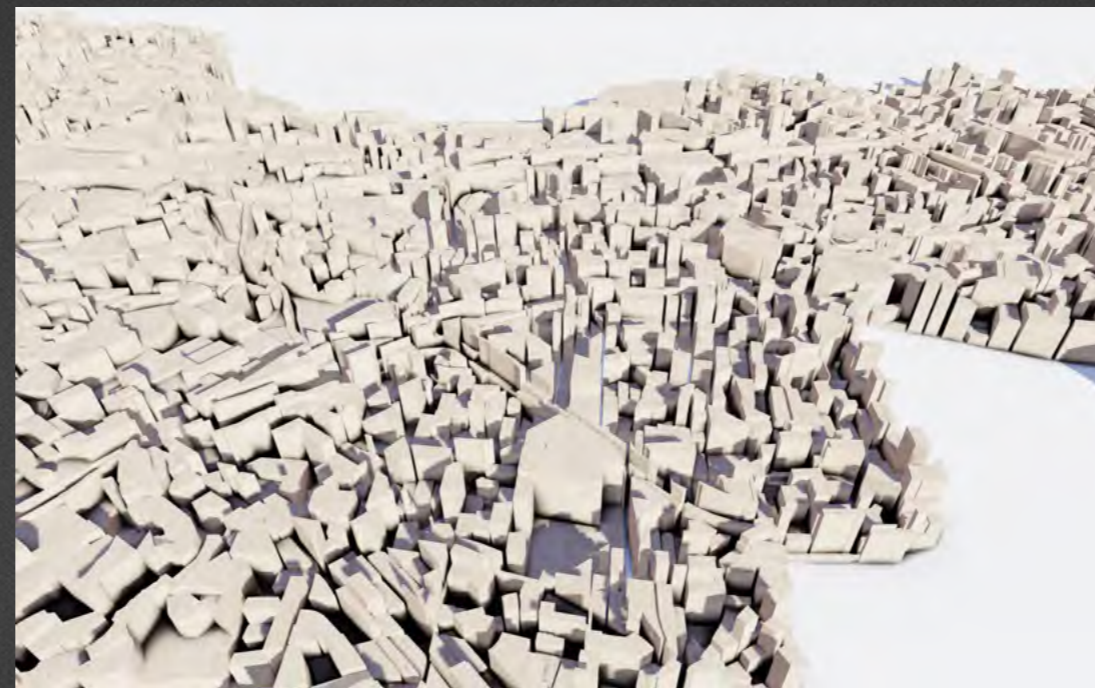
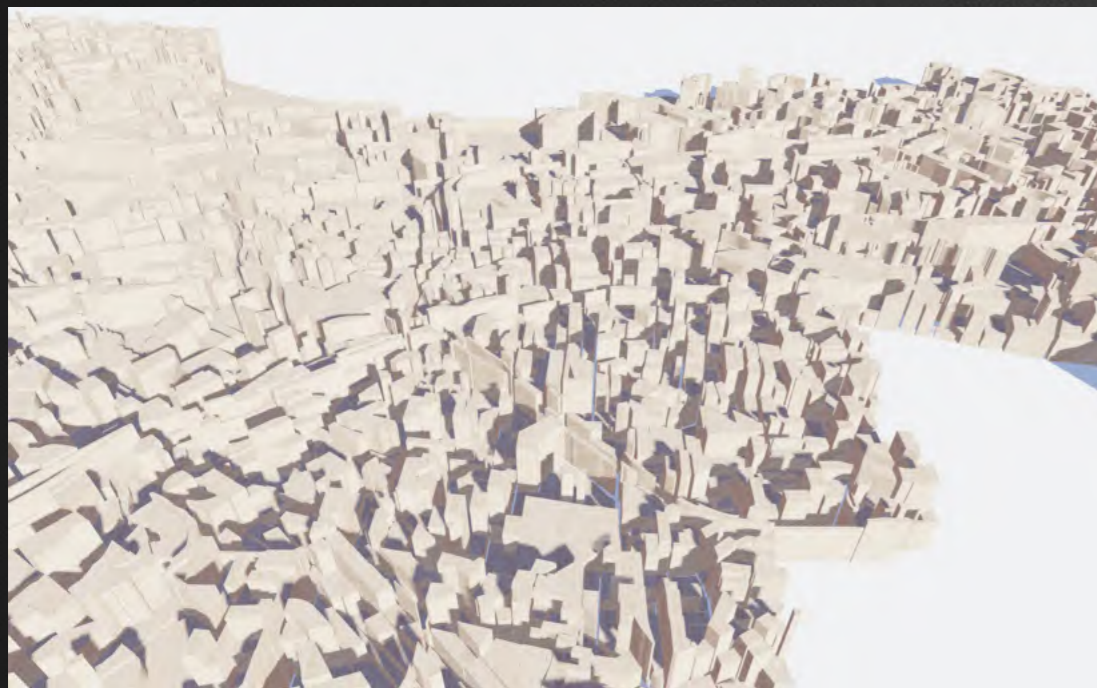
# 调色

ACES Tone Mapping + Color Grading





# 在有限的电脑配置内实现“无限”的画质



采样！ 采样！ 采样！



# 渐进式增强

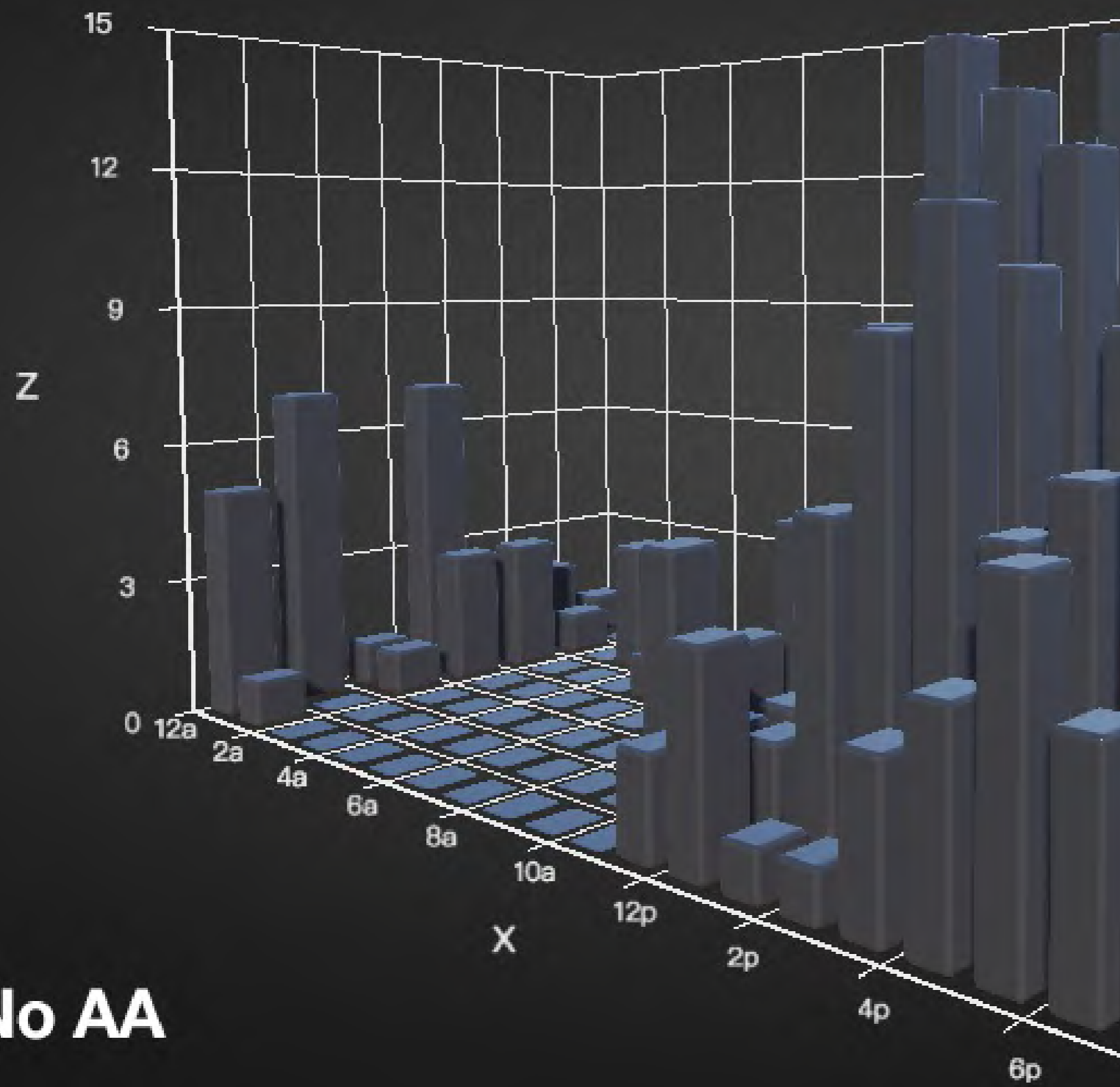
- 将采样分布到多帧中
- 交互的时候能够立刻反馈
- 停止交互后渐进增强画面



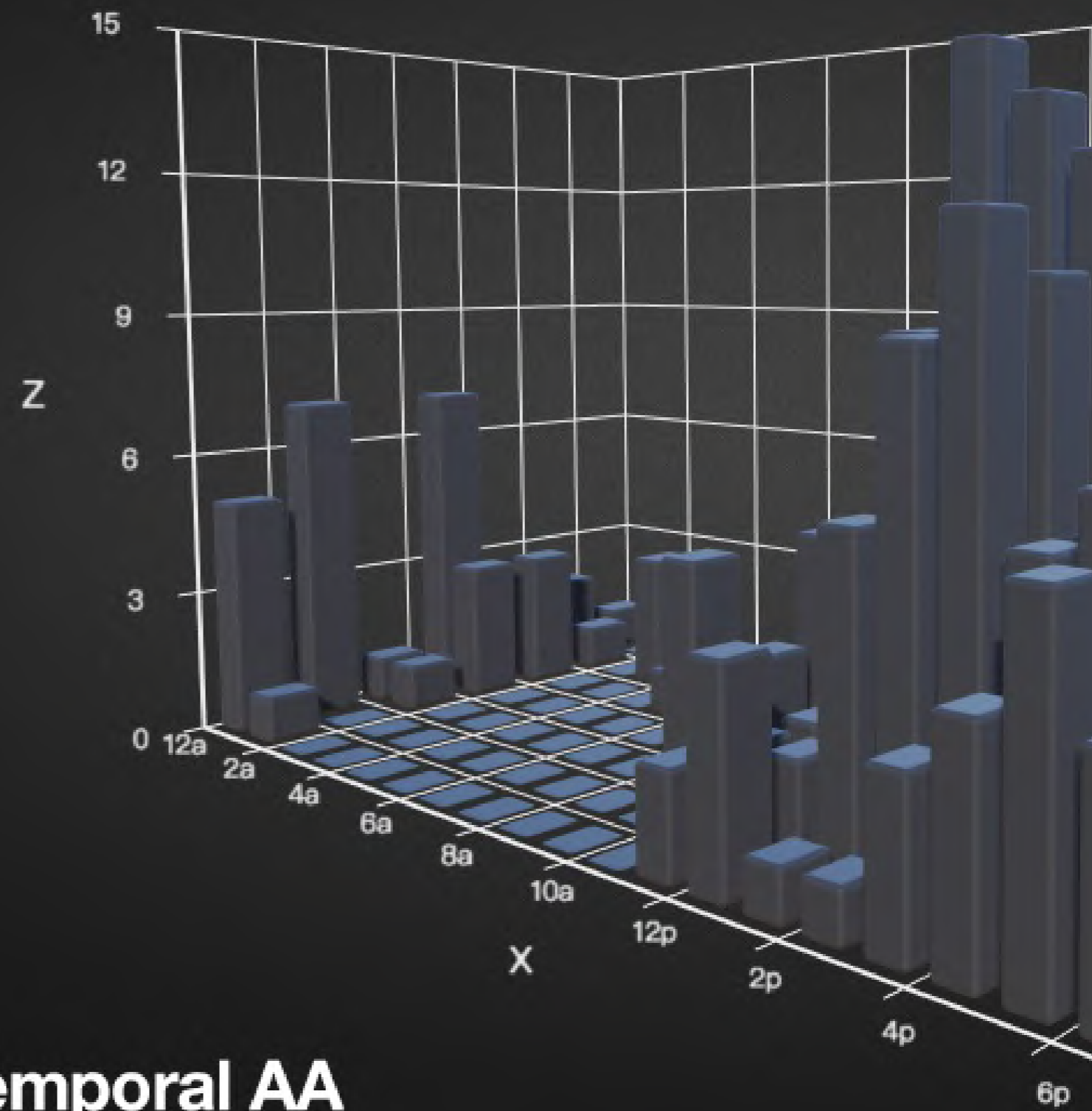
# 抗锯齿

- MSAA (不支持离线的 FrameBuffer)
- SSAA (慢)
- FXAA (效果差强人意)
- **Temporal AA**





**No AA**



**Temporal AA**



# TEMPORAL METHODS 无法解决的

- 动态的画面
- 几何信息缺失
- 精度不够 - Bias



# MORE

- GPGPU 中实现 Barnes Hut



# 总结

- 三维图表的绘制
  - 点线面
- GPU 通用计算
- 优化画质的方法
  - Temporal Methods



THANKS





