# RAYDATA可视化之探索与实践

王 波

光启元研发部总经理

# 对标阿里 P7，带你打破前端职业天花板

【前端实战训练营】3 个月带你进大厂 | 7月7日开营

扫码获取详细大纲
并咨询课程详情

- 8 大模块教学
- 9 大实战项目
- 15 周全程直播授课
- 大厂助教 1v1 答疑
- 简历直推一线互联网公司

# 王　波

高级软件工程师

光启元研发部总经理

## INDIVIDUAL RESUME

在互联网、计算机图形学、BIM、GIS、人工智能等领域有着丰富的实践经验和技术积累。

制定公司研发战略目标，探索前沿技术，指导攻克技术难点，负责研发部技术研发、技术管理、技术规划和人才培养等工作。

带领研发团队研发了 RAYDATA 系列产品，使其在数据可视化、跨平台渲染、云渲染等技术领域实现了商业落地。

# 目 录

CONTENTS

# RAYDATA

智慧城市 | 智慧政务 | 智慧能源

智慧建筑 | 智慧园区 | 智慧文旅

智慧金融 | 智慧医疗



# RAYDATA

## 定制服务

建模、GUI 设计、
图表设计、场景编辑、视效、
动效、数据对接等

## 产品工具

桌面端、网页端、移动端；
资产商城；资产编辑器等

# 目 录

CONTENTS

# 游戏引擎

为什么
选择游戏引擎?

01 _
3D场景

02 _
渲染效果

03 _
视觉交互

04 _
图形化编辑器

# 游戏引擎


实时3D互动内容创作
和运营平台

UNREAL
ENGINE

实时3D创作工具

# 游戏引擎：Unity

**Unity 构建 WebGL 项目原理：**

Unity 运行时代码（用 C 和 C++ 编写）通过Emscripten 编译器工具链交叉编译为 WebAssembly。

.NET 游戏代码（C# 脚本）需要先通过 IL2CPP 技术转换为 C++ 代码，然后再通过 Emscripten 编译为 WebAssembly。

## 常见的应用场景：
## 通过2D控件控制3D对象

```
unityInstance.SendMessage(objectName, methodName, value);
```

```
unityInstance.SendMessage('Building', 'PopupWindow', 'Name');
```

# 游戏引擎：Unity

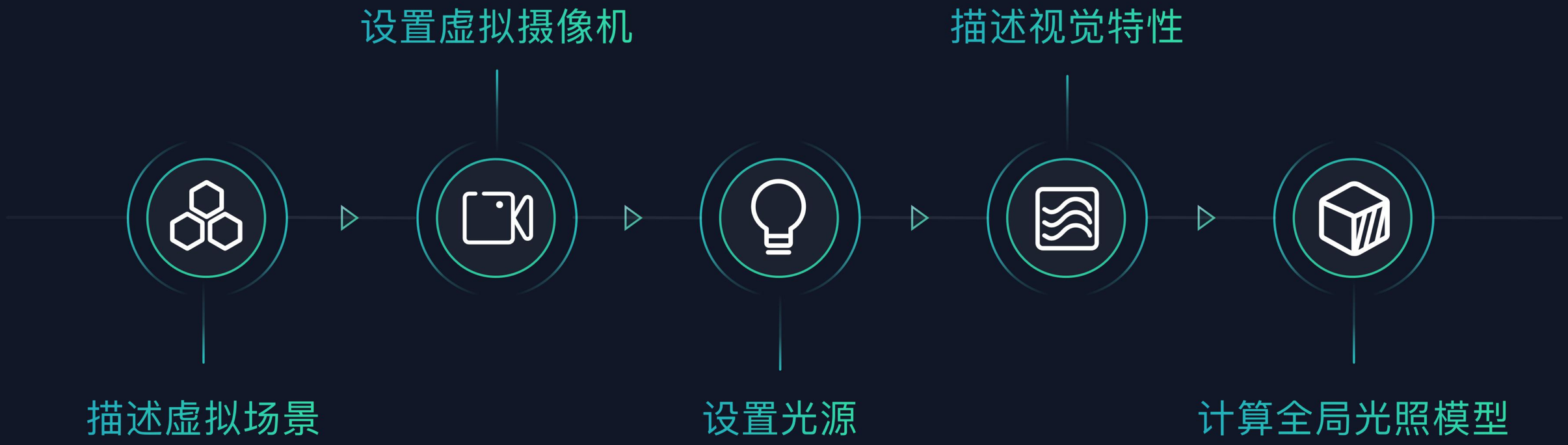
单线程

网络

# 目 录

CONTENTS

# 3D渲染



# 渲染管线


工具阶段
（脱机）

资产调节阶段
（脱机）

应用程序阶段
（CPU）

几何阶段
（GPU）

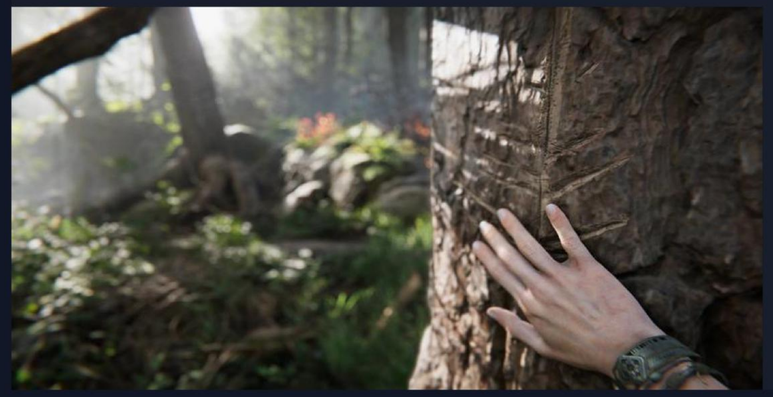
光栅化阶段
（GPU）

# 渲染管线

RENDER
PIPELINES

## 通用渲染管线（URP）

简化的、基于物理的光照和材质、范围更广、使用单通道前向渲染



## 高清渲染管线（HDRP）

基于物理的光照和材质、支持前向渲染和延迟渲染、画质逼真

# 光照

## 局部光照模型

Local Illumination Model

## 全局光照模型

Global Illumination Model

## Cookie

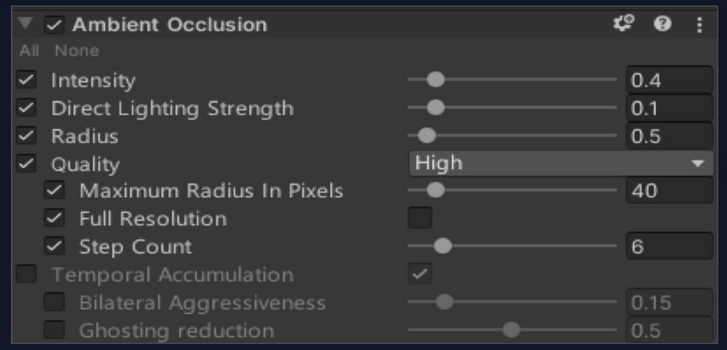
灯光投射的RGB纹理

模拟复杂照明效果的有效方式

# 光照

## 环境光遮蔽

Ambient Occlusion, AO




## 延迟渲染

Deferred Rendering

主要的光照计算是

在屏幕空间进行的而非观察空间

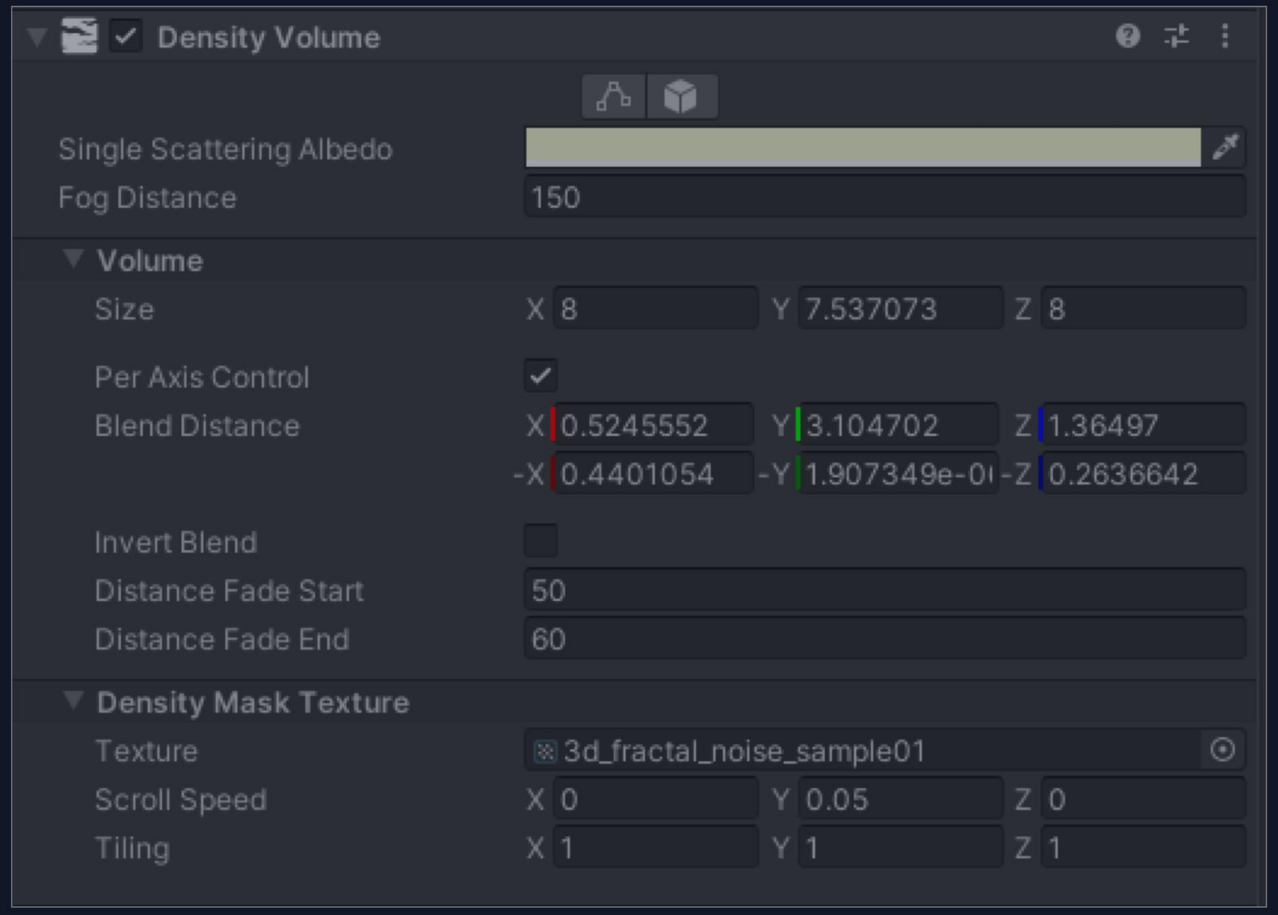
# 效果

## 密度体积
## Density Volume

密度体积是

一种表示为定向包围盒的加法雾效体积

可以在场景中模拟雾气等效果



# 效果

## 全屏后处理效果
## Full-Screen Post Effect

应用在已渲染的三维场景上

以增加真实感或做出特殊的风格

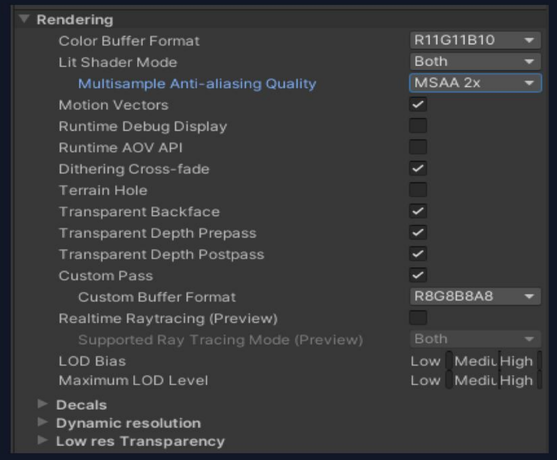
| |
|---|
| 动态模糊<br>Motion Blur |
| 景深模糊<br>Depth-of-Field Blur |

# 其他

01 _

## 多重采样抗锯齿

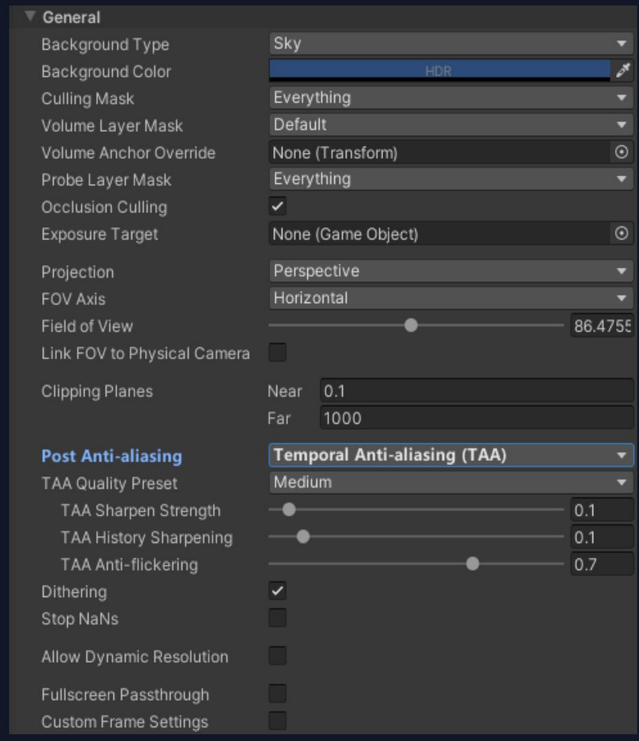
Multisample
Antialiasing, MSAA




02 _

## 临时性抗锯齿

Temporal
Antialiasing, TAA



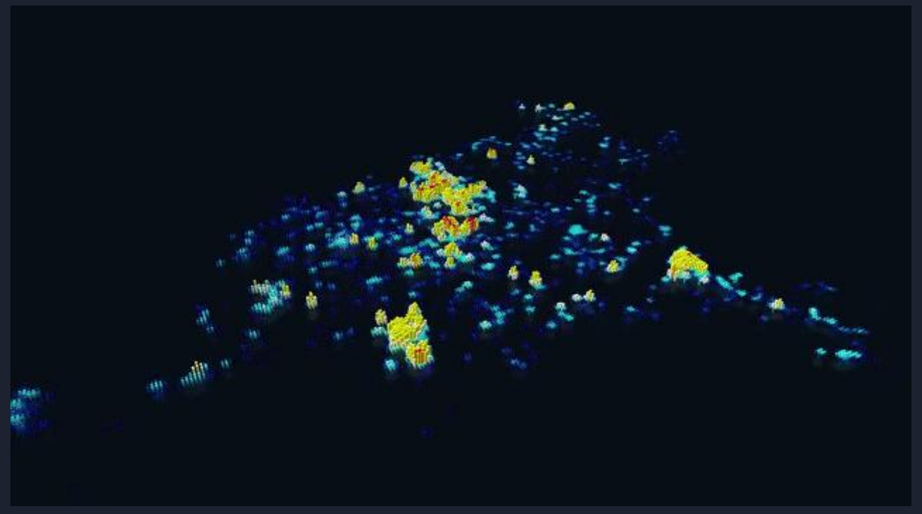

ShangHai
07:00 AM

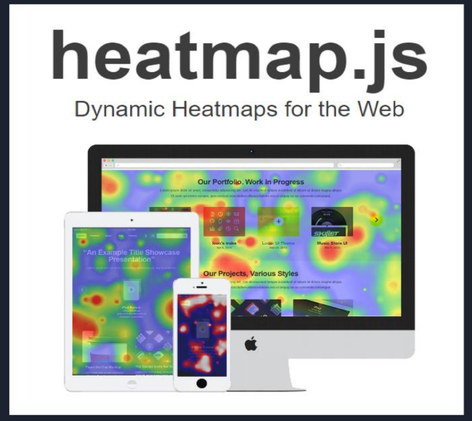
# 目 录

CONTENTS

# 应用：热力图

通过粒子系统或叠加
2D 图层的方式






常见的
JavaScript库

# 应用：热力图

## 腾讯位置服务
## 区域实时人口热力图接口

BIG DATA
VISUALIZATION

```
{
 "status":0,
 "data":[{
  "time":"2021-01-01 12:00:00",
  "points":[
    {"lat":39.9018,"lng":116.4101,"weight":10},
    {"lat":31.2261,"lng":121.4745,"weight":15}
  ...]
},{
  "time":"2021-01-01 12:00:00",
  "points":[
    {"lat":23.1217,"lng":113.2598,"weight":35},
    {"lat":22.5375,"lng":114.0611,"weight":80}
   ...]
  }
 ...]
}
```

# 应用：热力图

## GCJ-02 > WGS-84 > Mercator

```
// Mercator Projection
public static Point MercatorProjection(double lng, double lat)
{
    double x = lng * 20037508.3427892 / 180;
    double y = Math.Log(Math.Tan((90 + lat) * Math.PI / 360)) / (Math.PI / 180);
    y = y * 20037508.3427892 / 180;
    return new Point(x, y);
}
```

# 应用：热力图

通过C#
以位图方式
绘制热力图

01 _
笔刷混色

```
private ColorBlend GetColorBlend(int intensity)
{
    // 初始化 ColorBlend 实例
    ColorBlend colorBlend = new ColorBlend(3);

    // 设置渐变位置，其中全局变量 BrushStop 指的是笔刷变化点位置
    // 一般，值越小越靠近调色板上的红色，这个值需根据实际预警阈值进行调节
    colorBlend.Positions = new float[3] { 0, BrushStop, 1 };

    // 设置渐变颜色
    colorBlend.Colors = new Color[3]
    {
        Color.FromArgb(0, Color.White),
        // intensity 指的是热力点的中心浓度，相当于热力点的权重值
        Color.FromArgb(intensity, Color.Black),
        Color.FromArgb(intensity, Color.Black)
    };
    return colorBlend;
}
```

02 _
加载调色板

```
private int[] LoadPalette(string paletteFile)
{
    int[] palette = new int[256];

    // 使用 Bitmap 类读取调色板文件
    // 调色板文件是 256*1 pixels 的带透明通道的图片，从左至右是彩虹渐变色
    Bitmap paletteImage = (Bitmap)Bitmap.FromFile(paletteFile);

    // 依次获取调色板每个像素的 ARGB 值
    for (int i = 0; i < palette.Length - 1; i++)
        palette[i] = paletteImage.GetPixel(i, 0).ToArgb();
    // 最后一个颜色需要设置为透明，以保证没有热力点的区域保持原状
    palette[palette.Length - 1] = 0;

    return palette;
}
```

03 _
绘制热力点

```
private void DrawHeatPoint(Graphics surface, HeatPoint heatPoint)
{
    // 根据热力点位置计算椭圆参数
    int x = heatPoint.point.X - Radius;
    int y = heatPoint.point.Y - Radius;
    int len = 2 * Radius;

    // 根据以上计算结果绘制椭圆路径
    var ellipsePath = new GraphicsPath();
    ellipsePath.AddEllipse(x, y, len, len);

    // 构造一个 PathGradientBrush 并对其着色
    PathGradientBrush brush = new PathGradientBrush(ellipsePath);
    ColorBlend colorBlend = GetColorBlend(heatPoint.value);
    brush.InterpolationColors = colorBlend;

    // 使用 brush 在 surface 上填充椭圆
    surface.FillEllipse(brush, x, y, len, len);
}
```

# 应用：热力图

通过C#
以位图方式
绘制热力图

04 _
绘制灰度图

```
public void Draw(ref Bitmap bitmap)
{
    // 通过位图创建一个 Graphics 绘图对象 surface
    Graphics surface = Graphics.FromImage(bitmap);

    // 在 surface 上绘制热力点
    foreach (HeatPoint heatPoint in _heatPoints)
        DrawHeatPoint(surface, heatPoint);

    surface.Dispose();
}
```

05 _
着色

```
public void Colorize(ref Bitmap bitmap, string paletteFile)
{
    // 加载调色板
    int[] palette = LoadPalette(paletteFile);

    // 遍历灰度图的每个像素
    for (int y = 0; y < bitmap.Height; y++)
    {
        for (int x = 0; x < bitmap.Width; x++)
            // 以每个像素 Alpha 通道值（高 8 位）取反后的结果作为索引值，
            // 从调色板中取出相应颜色对输出位图的对应点进行着色
            bitmap.SetPixel(x, y, Color.FromArgb(
                palette[(byte)~(((uint)(bitmap.GetPixel(x, y).ToArgb())) >> 24)]));
    }
}
```

# 应用：热力图

## 通过C#以位图方式绘制热力图

创建一个空白位图，

实例化热力图对象，添加热力点数据

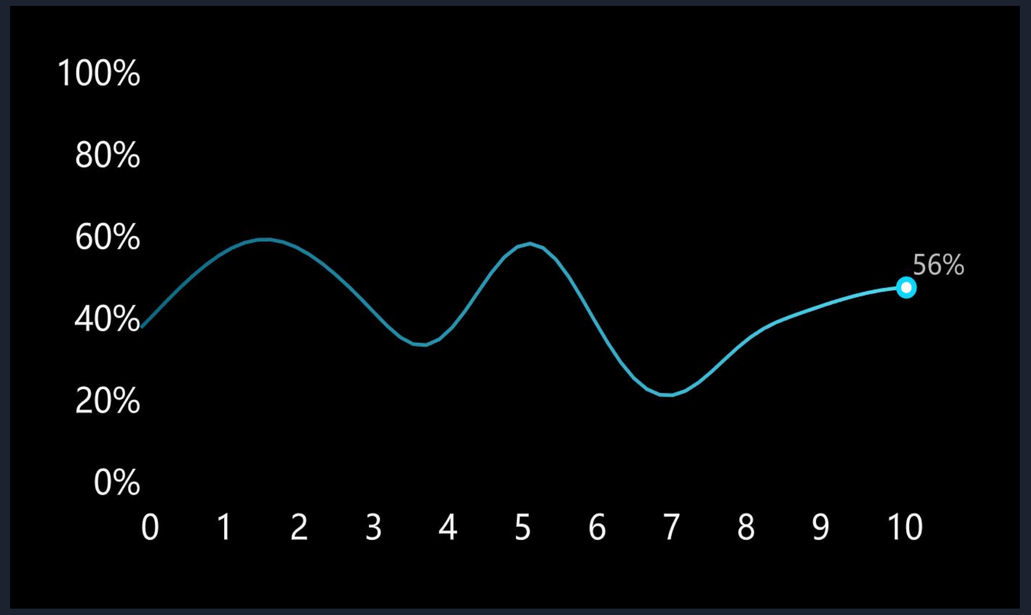
调用 Draw() 和 Colorize()

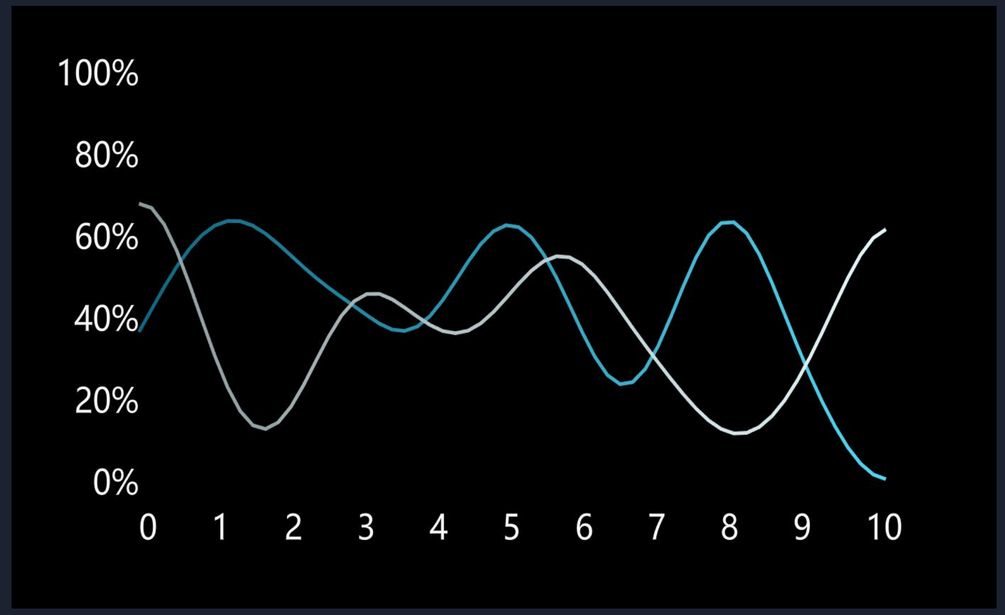
方法绘制热力图

```
//...
// 创建一个空白位图
Bitmap bitmap = new Bitmap(width, height, PixelFormat.Format32bppArgb);
// 实例化热力图对象
RayHeatMap heatMap = new RayHeatMap();
// 设置笔刷变化点位置
heatMap.BrushStop = brush;
// 设置热力点半径
heatMap.Radius = radius;

// 添加热力点数据
for (int i = 0; i < lst1.Count; i++)
{
    int y = (int)((lst1.Max() - lst1[i]) / scale);
    int x = (int)((lst2[i] - lst2.Min()) / scale);
    int weight = (int)((double)lst3[i] / lst3.Max() * 255);
    heatMap.AddPoint(new Point(x, y), weight);
}

// 调用 Draw() 和 Colorize() 方法绘制热力图
heatMap.Draw(ref bitmap);
heatMap.Colorize(ref bitmap, palettePath);
//...
```

# 应用：随机曲线

通过Perlin Simplex Noise(柏林单形噪声)算法模拟一些曲线图表




准备数据 ▷ 建立采样空间 ▷ 计算噪声值

# 应用：随机曲线

## 定义一个宏，目的是向下取整

C++实现方法

```
1    #define FASTFLOOR(x) ( ((x)>0) ? ((int)x) : (((int)x)-1) )
```

# 应用：随机曲线

## 定义一个置换表
## 保留为静态数据

C++实现方法

```
unsigned char perm[512] = {151,160,137,91,90,15,
  131,13,201,95,96,53,194,233,7,225,140,36,103,30,69,142,8,99,37,240,21,10,23,
  190, 6,148,247,120,234,75,0,26,197,62,94,252,219,203,117,35,11,32,57,177,33,
  88,237,149,56,87,174,20,125,136,171,168, 68,175,74,165,71,134,139,48,27,166,
  77,146,158,231,83,111,229,122,60,211,133,230,220,105,92,41,55,46,245,40,244,
  102,143,54, 65,25,63,161, 1,216,80,73,209,76,132,187,208, 89,18,169,200,196,
  135,130,116,188,159,86,164,100,109,198,173,186, 3,64,52,217,226,250,124,123,
  5,202,38,147,118,126,255,82,85,212,207,206,59,227,47,16,58,17,182,189,28,42,
  223,183,170,213,119,248,152, 2,44,154,163, 70,221,153,101,155,167, 43,172,9,
  129,22,39,253, 19,98,108,110,79,113,224,232,178,185, 112,104,218,246,97,228,
  251,34,242,193,238,210,144,12,191,179,162,241, 81,51,145,235,249,14,239,107,
  49,192,214, 31,181,199,106,157,184, 84,204,176,115,121,50,45,127, 4,150,254,
  138,236,205,93,222,114,67,29,24,72,243,141,128,195,78,66,215,61,156,180,
  151,160,137,91,90,15,
  131,13,201,95,96,53,194,233,7,225,140,36,103,30,69,142,8,99,37,240,21,10,23,
  190, 6,148,247,120,234,75,0,26,197,62,94,252,219,203,117,35,11,32,57,177,33,
  88,237,149,56,87,174,20,125,136,171,168, 68,175,74,165,71,134,139,48,27,166,
  77,146,158,231,83,111,229,122,60,211,133,230,220,105,92,41,55,46,245,40,244,
  102,143,54, 65,25,63,161, 1,216,80,73,209,76,132,187,208, 89,18,169,200,196,
  135,130,116,188,159,86,164,100,109,198,173,186, 3,64,52,217,226,250,124,123,
  5,202,38,147,118,126,255,82,85,212,207,206,59,227,47,16,58,17,182,189,28,42,
  223,183,170,213,119,248,152, 2,44,154,163, 70,221,153,101,155,167, 43,172,9,
  129,22,39,253, 19,98,108,110,79,113,224,232,178,185, 112,104,218,246,97,228,
  251,34,242,193,238,210,144,12,191,179,162,241, 81,51,145,235,249,14,239,107,
  49,192,214, 31,181,199,106,157,184, 84,204,176,115,121,50,45,127, 4,150,254,
  138,236,205,93,222,114,67,29,24,72,243,141,128,195,78,66,215,61,156,180
};
```

# 应用：随机曲线

## 计算梯度向量
## 与方向向量的点积

C++实现方法

```
float Grad( int hash, float x ) {
    // 取 hash 值低 4 位
    int h = hash & 15;

    // 取 h 值低 3 位，将梯度值 grad 限制在 1.0 至 8.0 之间
    float grad = 1.0f + (h & 7);

    // 对梯度值 grad 随机取反
    if (h&8) grad = -grad;

    // 返回梯度值 grad 与距离 x 的乘积
    return ( grad * x );
}
```

# 应用：随机曲线

将噪声值
归一化至[–1,1]

C++实现方法

```cpp
float SimplexNoise(float x) {
    // 分别计算两个临近整数坐标 i0 和 i1
    int i0 = FASTFLOOR(x);
    int i1 = i0 + 1;

    // 分别计算指向 x 的向量 x0 和 x1，相当于距离
    float x0 = x - i0;
    float x1 = x0 - 1.0f;

    // 定义两个噪声值 n0 和 n1
    // 插值函数为 (1 - x^2)^4
    float n0, n1;

    float t0 = 1.0f - x0*x0;
    t0 *= t0;
    n0 = t0 * t0 * Grad(perm[i0 & 0xff], x0);

    float t1 = 1.0f - x1*x1;
    t1 *= t1;
    n1 = t1 * t1 * Grad(perm[i1 & 0xff], x1);

    // 观察 n0 + n1 曲线或对其求导后发现
    // 当 x 为 1/2 时，二者叠加后的噪声值最大
    // 此时，各自插值函数值为 (3/4)^4，Grad 值为 4
    // 因此，叠加后的噪声最大值为 8*(3/4)^4 = 2.53125
    // 若归一化至 [-1,1]，需除以 2.53125，即乘以 0.395
    return 0.395f * (n0 + n1);
}
```

# 应用：GIS

ArcGIS、Cesium（开源）和 SuperMap GIS

# GIS：SuperMap Scene SDK

01 _

SuperMap GIS 与 Unity 游戏引擎深度集成的开发平台。

02 _

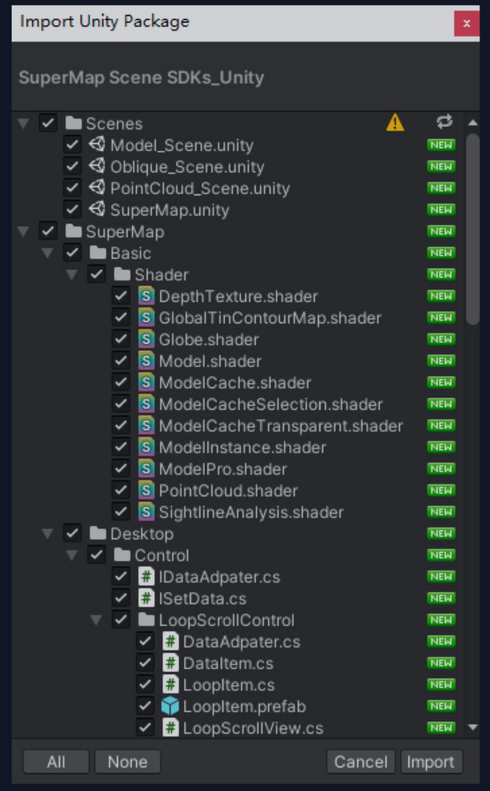
支持影像、地形、倾斜摄影模型、人工模型、BIM 等多种海量 GIS 空间数据的本地/在线浏览查询、空间分析等多种功能。

03 _

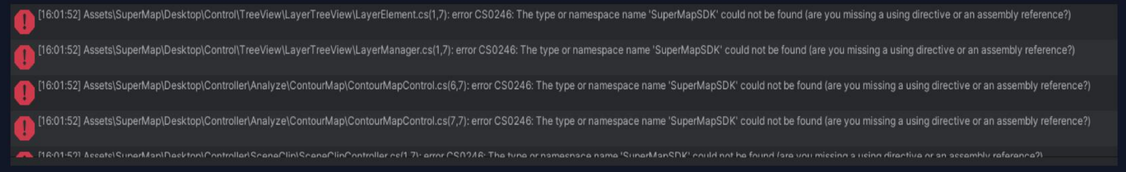
SDK 支持的数据类型有：三维影像缓存文件（*.sci3d）、三维地形缓存文件（*.sct）和三维切片缓存（*.scp）

# GIS：SuperMap Scene SDK

新建项目
并导入



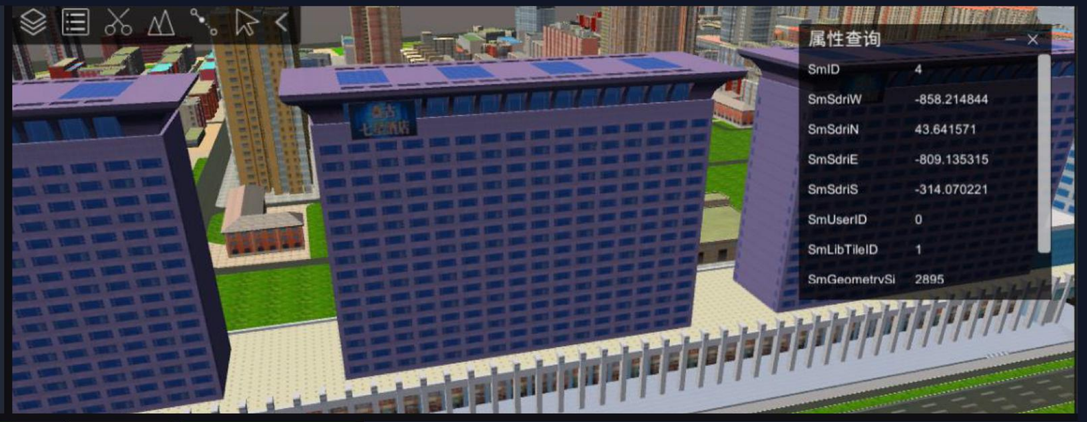

出现类型
或命名空间错误

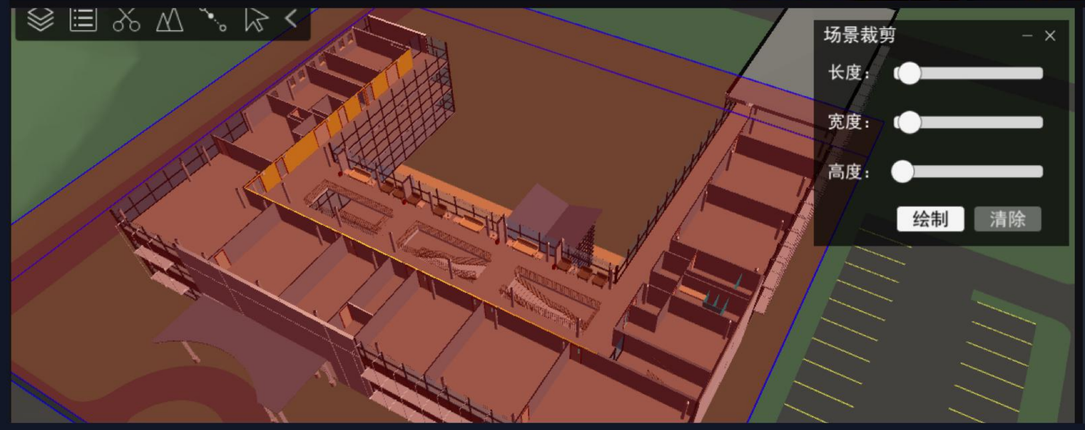


# GIS：可视化常用功能

BIG
DATA
VISUALIZATION

属性查询



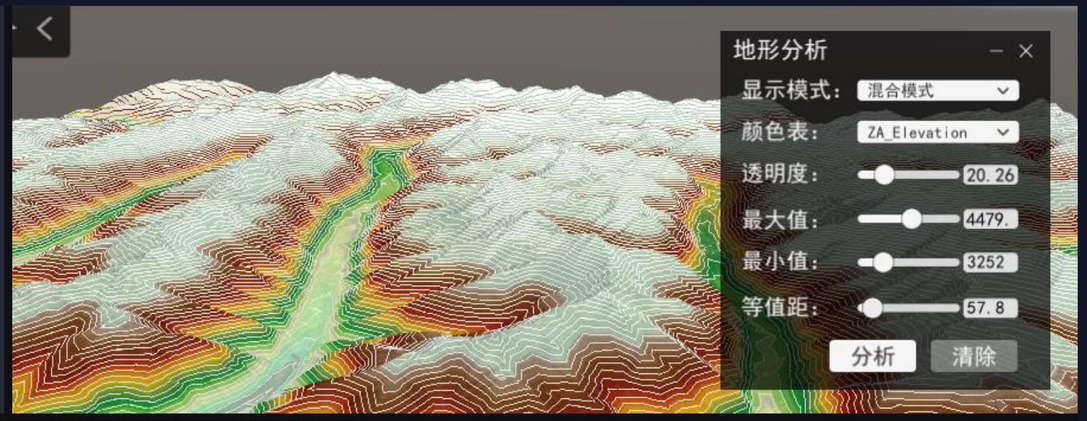




场景裁剪

# GIS：可视化常用功能

BIG
DATA
VISUALIZATION

## 等值线分析





## 通视分析

# 目 录

CONTENTS

# 展望：云渲染

随时随地

多用户

大场景

# 展望：数字孪生

## 01_ 准确性

- 如何提高基于 PBR 的渲染真实度
- 如何提高画面质量
- 如何有效兼容不同平台

……

## 02_ 时效性

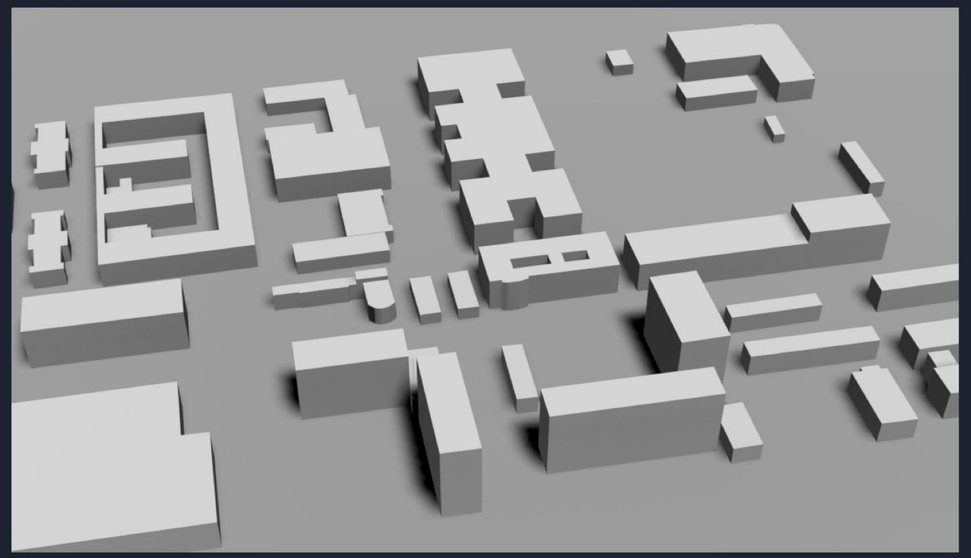
- 如何提高场景资源更新频率
- 如何提高业务数据刷新频率
- 如何提高渲染帧率

……

# 展望：AI与可视化

在深度学习的目标检测和图像分割方向与可视化做了一些探索和尝试。


– 图像识别截图 –



– 程序建模截图 –

GMTC
全球大前端技术大会

# THANKS

InfoQ

GMTC
全球大前端技术大会

# THANKS

InfoQ