



3.2 UE场景运行渲染与交互



本章目录

UE场景：从对象状态到画面与交互

第三章 三维环境建模与仿真

3.1 三维仿真基础与工具选型

3.2 UE场景运行渲染与交互

3.3 多模态传感器数据生成

3.4 分布式仿真架构与数据接口

3.5 场景与载具资源开发

3.6 例程体系与综合实践

3.7 习题案例分析与课程设计

本节 5 个教学单元

3.2.1 框架与渲染流程

教材对应：第3.1.4.1节

3.2.2 三维资产管理

教材对应：第3.1.4.2节

3.2.3 光照渲染系统

教材对应：第3.1.4.3节

3.2.4 物理与动画系统

教材对应：第3.1.4.4节

3.2.5 仿真交互框架

教材对应：第3.1.4.5节





3.2 UE场景运行渲染与交互



3.2.1 框架与渲染流程：三层结构

UE用三层结构把对象状态变成屏幕图像

仿真交互框架：用Actor、控制器、事件与Tick决定这一帧各对象的状态。

底层引擎模块：虚幻引擎（UE）管理模型、材质、动画与物理资源，并用渲染管线把三维场景实时转换为二维图像。

平台适配接口：渲染硬件接口（RHI）把同一套渲染命令转换为DirectX、Vulkan、Metal等图形应用程序接口（API）的调用。

交互框架先定状态，引擎据此渲染，RHI再把命令交给显卡。



3.2.1 框架与渲染流程：场景对象

先认识关卡、Actor和组件

关卡 (Level) : 组织一组场景对象及其摆放，例如地面、建筑、灯光和箱体。

实体 (Actor) : 可放入关卡的对象实例。箱体Actor可以移动和旋转，也可以执行自己的逻辑。

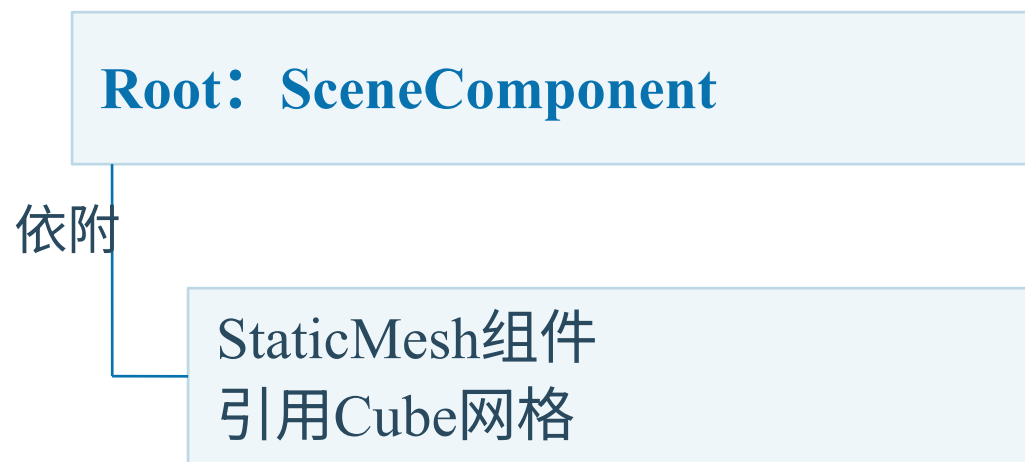
组件 (Component) : Actor拥有的功能模块，例如静态网格、相机或碰撞形状；各组件职责不同。

本例对应： 箱体Actor通过StaticMesh组件引用Cube网格资产；改变实例位置，不会改写网格文件。

关卡组织实体，Actor组合组件，组件提供具体能力。

3.2.1 框架与渲染流程：组件拥有与依附 根组件提供Actor的空间变换

盒体Actor：拥有组件



根组件： SceneComponent提供位置、旋转和尺度；Actor的变换取自根组件。

子组件： 相对父组件保存变换；父组件移动时，子组件随依附关系更新。

一般组件： ActorComponent也可只保存逻辑，例如状态管理，不必有空间位置。

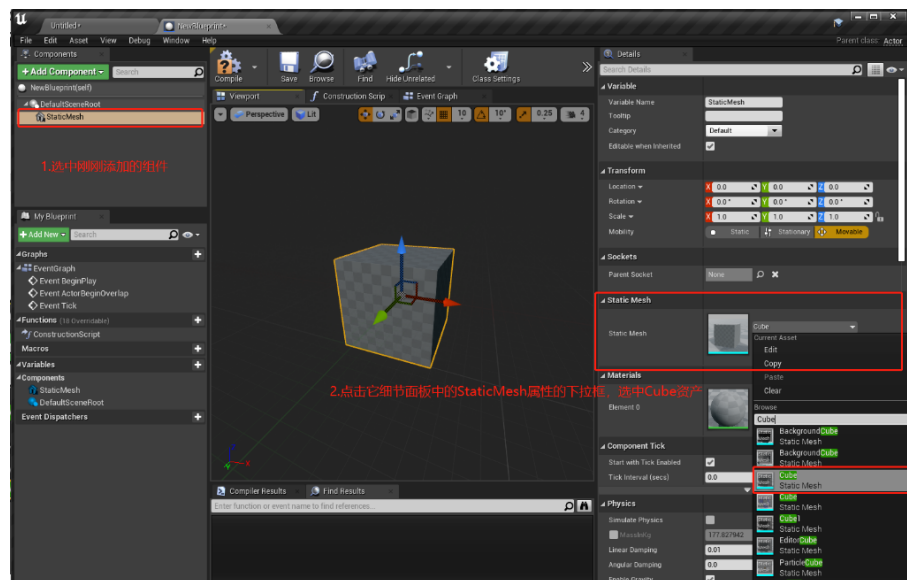
有空间变换的组件可建立父子关系；逻辑组件可以没有位置。

3.2 UE场景运行渲染与交互

3.2.1 框架与渲染流程：编辑器中的对象

真实箱体：组件引用Cube网格资产

图：UE4.27.2中箱体蓝图的组件配置



组件树：DefaultSceneRoot→StaticMesh



属性：Static Mesh字段选择Cube；
Transform配置组件变换。

所选网格 →



启动：编译保存蓝图，拖入关卡后点击Play，观察实例运行。

选中组件改变实例配置，选择网格资产决定显示的形状。

3.2.1 框架与渲染流程：对象与交换信息

用无人机区分对象与交换的信息

对象或信息	无人机示例	承担的作用
Level	试飞场景	组织地形、建筑、光照和多个实体
Actor	一架无人机	作为可放置、可移动和可识别的场景实体
Component	网格、碰撞体、相机	分别提供外观、接触边界和观测入口
交互信息	位置、姿态、控制输入	让外部模型或蓝图读取、更新和验证实体行为

位置、姿态和控制输入是信息，不是新的对象层级。

3.2.1 框架与渲染流程：事件与Tick

事件写入状态，Tick按时间更新

BeginPlay： 对象开始参与运行时进行初始化，例如设置初始角度和目标速度。

输入事件： 收到新指令后更新速度变量；指令未变化时，无需每帧重复建立同一配置。

Tick： 按启用状态和调度执行更新；DeltaSeconds表示本次更新使用的时间间隔，单位为秒。

时间作用： 速度乘时间得到位移，角速度乘时间得到角增量；固定每帧增量则会随帧率改变速度。

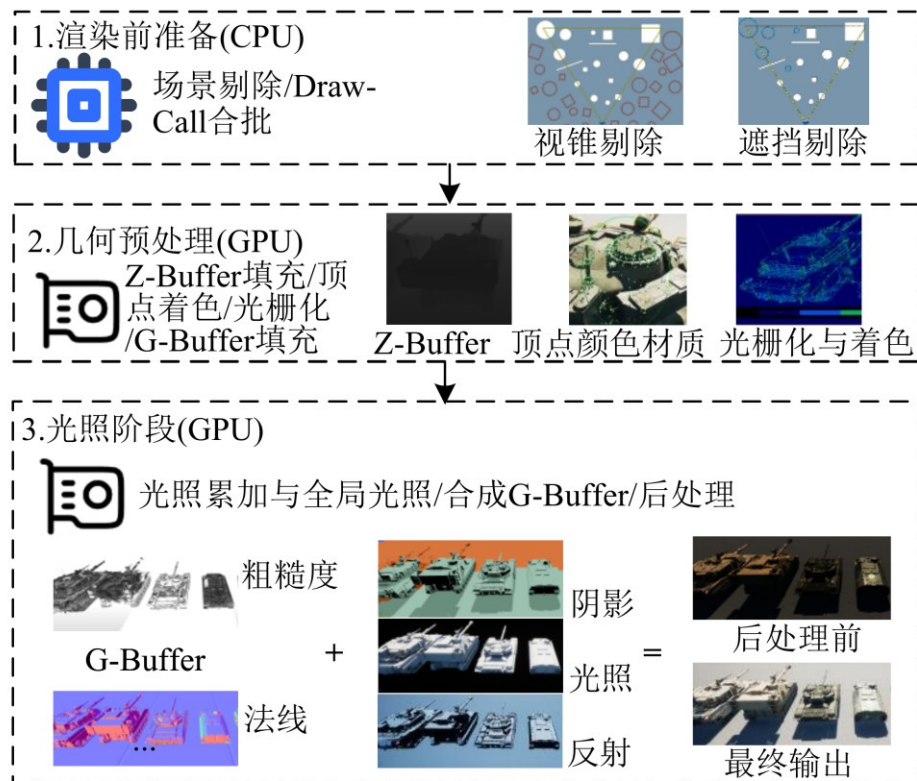
一步算例： 角速度 $\omega = 90^\circ/\text{s}$ 、帧间隔 $\Delta t = 1/60 \text{ s}$ 时，本帧转 $\Delta\theta = \omega \Delta t = 1.5^\circ$ ；帧间隔加倍则每帧转 3° ，每秒仍是 90° 。

触发时保存新的目标；持续变化时使用本次时间间隔。

3.2 UE场景运行渲染与交互

3.2.1 框架与渲染流程：渲染管线总图

每帧先由CPU准备，再由GPU生成图像



中央处理器（CPU）：更新对象状态，筛选绘制内容，提交绘制任务。

图形处理器（GPU）：处理几何与表面，计算光照及后处理，输出图像。

阅读顺序：渲染前准备→几何预处理→光照阶段，后面逐段放大。

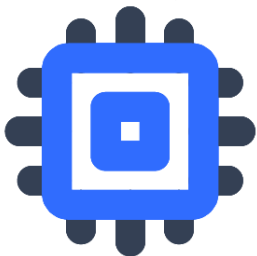
图3.4 渲染管线流程

3.2.1 框架与渲染流程：剔除

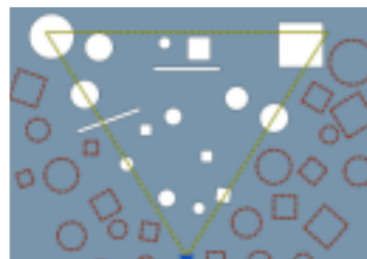
距离、视锥与遮挡测试减少不必绘制的对象

剔除：通过距离、视锥和遮挡测试，排除不必绘制的对象。

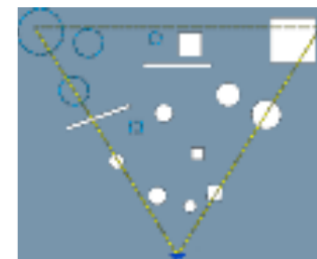
1.渲染前准备(CPU)



场景剔除/Draw-
Call合批



视锥剔除



遮挡剔除

遮挡测试：依方法由CPU或GPU执行；视野外对象仍可更新运动状态。

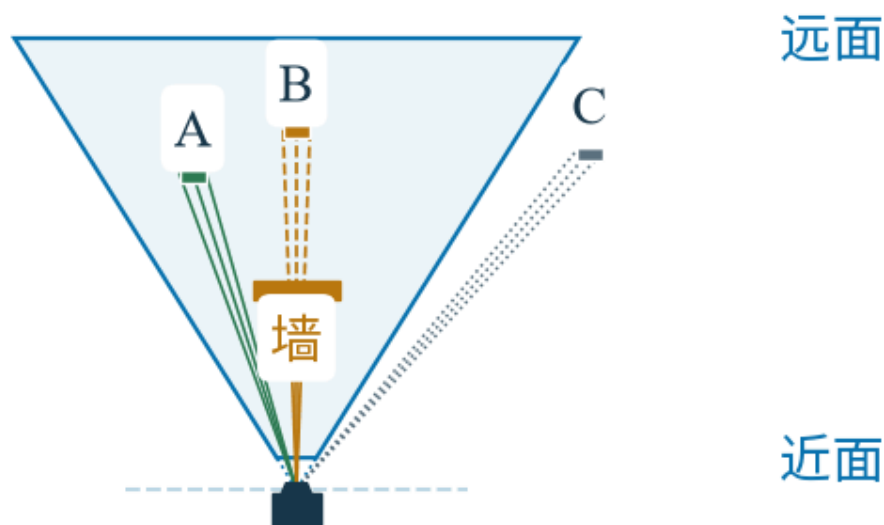
图3.4（局部）可见性准备与对象筛选

先排除不可见对象，绘制预算才能留给视野中的表面。

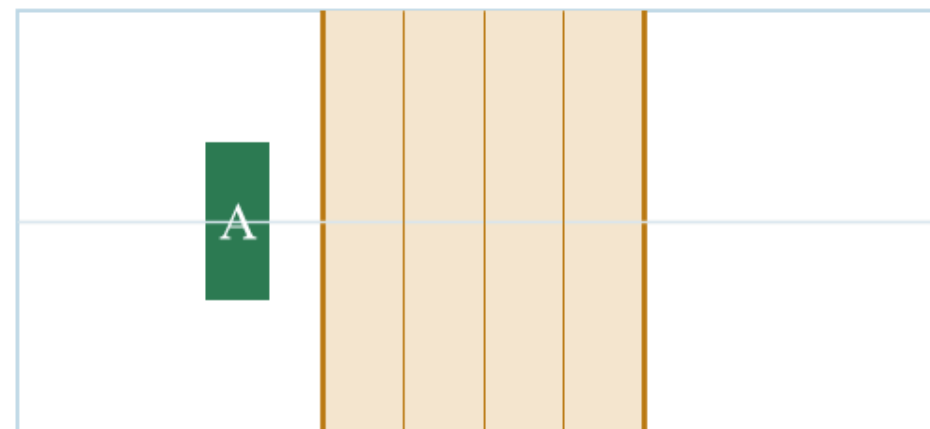
3.2.1 框架与渲染流程：剔除

相机移动后，哪些物体进入画面？

俯视（水平截面）



相机图像



■ A 可见

■ B 被挡

■ C 视锥外

图：左为视锥水平截面，右为同一相机图像；区分视锥外、被挡与可见

3.2.1 框架与渲染流程：绘制调用与合批

合批减少绘制调用，PSO减少状态切换

绘制调用 (Draw Call)： CPU每提交一次绘制命令，都要交代顶点缓冲、索引缓冲和材质状态，调用越多CPU负担越重。

合批与实例化： 网格和材质相同的对象合成一次提交。假设剔除后剩120个对象，只用到4种“网格+材质”组合，理想情况下调用次数可从约120次降到约4次。

管道状态对象 (PSO)： 预先编译并缓存着色器绑定、输入装配、混合模式等整套状态，运行时直接取用，减少状态切换开销。

合批条件： 材质不同或需逐个排序的透明物体难以合批，实际次数取决于场景组织。

剔除减少要画的对象，合批和PSO减少提交与切换的开销。

3.2 UE场景运行渲染与交互

3.2.1 框架与渲染流程：光栅化与深度

几何处理得到像素对应的表面信息

光栅化：把投影后的三角形铺到像素上，并插值出各像素的法线与纹理坐标。



图3.4（局部）几何预处理（GPU）

深度缓冲（Z-Buffer）：存每个像素最近表面的深度编码，用于判断遮挡。

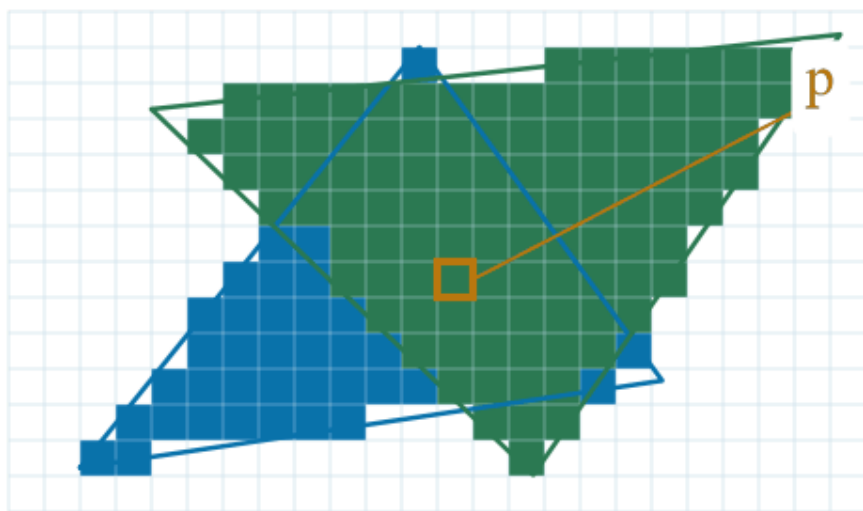
提前深度测试（Early-Z）：着色前先比深度，更远的表面直接丢弃，不算材质。

深度用于可见性判断，法线和材质参数供光照计算。

3.2.1 框架与渲染流程：光栅化与深度

投到同一像素的表面，谁挡住谁？

逐像素绘制结果



■ A ■ B

像素 p: 先 A, 后 B

A 已存 0.8576

B 待绘 0.7737

0

1

$0.7737 < 0.8576$ B 通过, 写入 B

■ p: B d = 0.7737

不透明; 深度 d: 近 0, 远 1

图: 不透明表面逐像素比较深度, 保留更近表面的颜色

3.2.1 框架与渲染流程：延迟渲染

延迟渲染先存表面，再计算受光



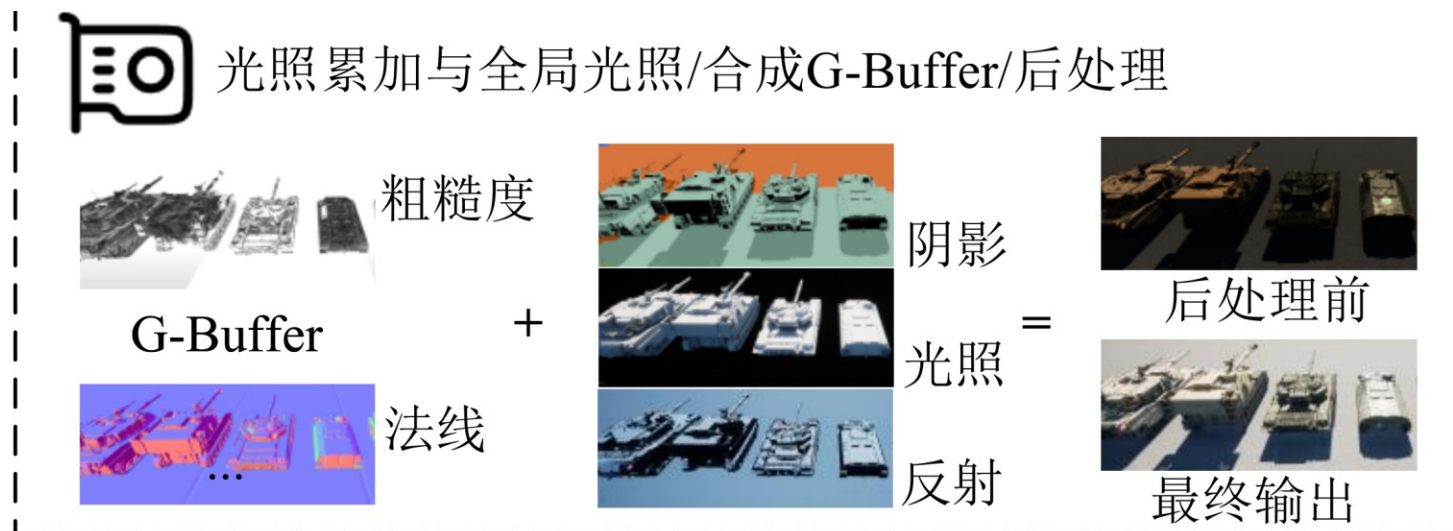
几何缓冲（G-Buffer）：延迟渲染保存法线、基础颜色、粗糙度等属性的一组缓冲；深度通常另有缓冲。

延迟的好处：光照只对屏幕上的可见像素逐个光源计算，多光源场景仍高效；前向渲染则在绘制每个表面时直接计算光照。

表面缓冲保存中间量；最终颜色还取决于光照与后处理。

3.2.1 框架与渲染流程：光照与后处理

光照合成后，后处理改变最终像素



表面数据： G-Buffer提供粗糙度、法线等属性。

光照合成： 依据表面属性、光源和阴影计算光照，再累加光照与反射结果。

后处理： 曝光、色调映射等改变最终像素，几何位置不变。

图3.4（局部）光照阶段（GPU）

固定几何仍可改变亮暗；曝光和后处理也会改变图像。

3.2.1 框架与渲染流程：帧耗时算例

用每帧耗时定位优化方向

帧率与帧时：帧率 f 指每秒帧数（FPS），每帧可用时间 $T = 1000/f$ ms；30FPS约33.3 ms，60FPS约16.7 ms。

读取指标：在UE编辑器中运行关卡后按~键打开控制台，输入stat unit，读取Frame（整帧）、Game（游戏线程）、Draw（渲染线程）和GPU的毫秒数。

题设算例：Game约9 ms、Draw约6 ms、GPU约28 ms。三段在流水线中处理不同的帧，整帧不是43 ms，而是接近最慢的GPU（约28 ms，约36FPS），应先查GPU工作。

帧率上限：t.MaxFPS 30只把上限设为30FPS，算不完时仍会更低；它也不规定相机采样或控制频率。

CPU与GPU可流水并行，帧耗时不能简单相加。

3.2.1 框架与渲染流程：定位瓶颈

耗时变长时，先找到具体工作

Game偏高：检查Actor的定期更新（Tick）、重复逻辑或通信处理；一次性工作放入对应事件。

Draw偏高：检查绘制请求与对象组织；合批或实例化需要兼容的资源与材质条件。

GPU偏高：检查像素数量、材质、阴影与后处理；细节层级（LOD）可减少渲染几何成本。

对照：固定相机和场景，再改变一个设置；同时记录Frame耗时与目标边缘是否仍清楚。

一次只改变一个主要条件，用帧时和画面解释收益。



3.2 UE场景运行渲染与交互



本章目录

UE场景：从对象状态到画面与交互

第三章 三维环境建模与仿真

3.1 三维仿真基础与工具选型

3.2 UE场景运行渲染与交互

3.3 多模态传感器数据生成

3.4 分布式仿真架构与数据接口

3.5 场景与载具资源开发

3.6 例程体系与综合实践

3.7 习题案例分析与课程设计

本节 5 个教学单元

3.2.1 框架与渲染流程

教材对应：第3.1.4.1节

3.2.2 三维资产管理

教材对应：第3.1.4.2节

3.2.3 光照渲染系统

教材对应：第3.1.4.3节

3.2.4 物理与动画系统

教材对应：第3.1.4.4节

3.2.5 仿真交互框架

教材对应：第3.1.4.5节



3.2.2 三维资产管理：导入管线

外部模型经导入成为UE资产，再被组件引用

数字内容创作 (DCC) 工具：在Blender、Maya、3ds Max等软件中建模、展开纹理坐标并制作骨骼动画。

导入保留：取决于格式与导入器；材质、贴图连接与动画需逐项核验。

资产与实例：导入后成为.uasset资产，由内容浏览器管理；组件引用资产，同一资产可放出多个实例。

外部模型导入后成为资产，关卡里的实例只引用资产、不复制它。

3.2.2 三维资产管理：几何细节

LOD与Nanite按屏幕占用决定几何细节

LOD：为同一模型准备由精到粗的几套网格，按它在屏幕上占多大切换，远处用更少三角形。

一步算例：画面高度约与距离 d 成反比、像素数约与 d^2 成反比； d 从10 m增到100 m，像素约为原来的1/100。

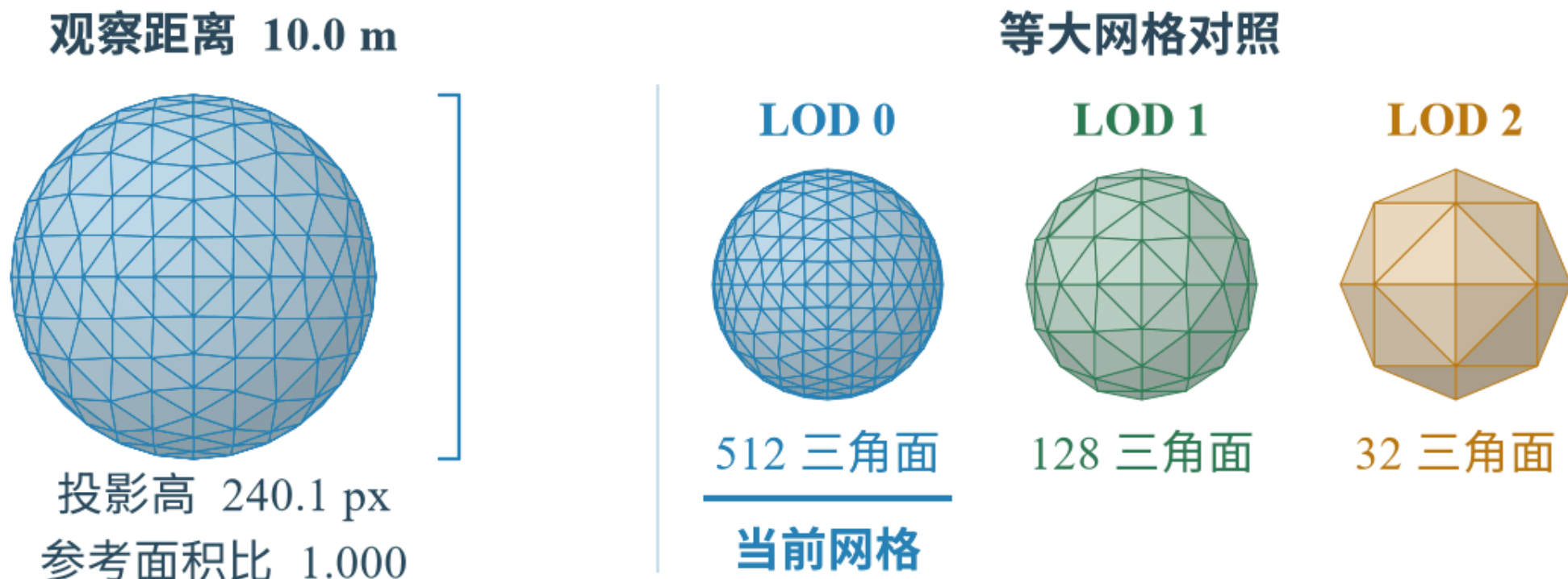
Nanite（虚拟几何体）：UE5按屏幕像素需求自动筛选细节，可减少手工制作LOD的工作，适合高细节扫描模型。

检查方法：固定相机路径观察LOD切换时门窗边缘，再比较帧耗时；看得清与算得省要分开判断。

LOD与UE5的Nanite按屏幕需求控制几何细节和绘制负担。

3.2.2 三维资产管理：几何细节

物体退远时，屏幕上的细节需求随之减少



图：固定视场下按投影尺度切换网格；右侧等大对照便于观察网格结构



3.2 UE场景运行渲染与交互



3.2.2 三维资产管理：纹理坐标与多级纹理

UV把贴图贴准，Mipmap让远处纹理不闪烁

纹理坐标 (UV)：用二维坐标对应表面与贴图，U为横轴、V为纵轴；本例一个纹理单元为0到1，普通UV可超界平铺。

一步算例： 1024×1024 贴图上的 $(u, v) = (0.25, 0.5)$ 的点，对应像素 $(uW, vH) = (256, 512)$ 。

多级纹理 (Mipmap)：纹理预先逐级缩小；1024纹理单次覆盖128屏幕像素时，估计取第 $\log_2(1024/128) = 3$ 级。

纹理过滤：三线性过滤在相邻两级之间插值，各向异性过滤让斜看的地面和跑道保持清晰。

纹理采样要随观察尺度调整，远处与斜视表面才不易闪烁。



3.2.2 三维资产管理：材质输入

材质参数怎样改变表面受光

节点式材质编辑器：把纹理采样值和常数接到基础颜色、金属度、高光度、粗糙度、环境光遮蔽和法线等输入。

反射分布：粗糙度越高，高光越宽、反射越分散；金属度改变反射的组成。

高光度与环境光遮蔽：高光度（Specular）默认0.5，约为4%镜面反射；环境光遮蔽（AO）让缝隙和角落更暗。

法线贴图：改变着色用的局部法线，显出细小凹凸，网格轮廓与碰撞边界不变。

运行时：材质实例可在运行中改参数，虚拟纹理按需加载大贴图。

材质参数改变表面反射，几何与碰撞边界由对应模型决定。

3.2.2 三维资产管理：PBR参数

PBR参数各自改变哪种受光特征

基于物理的渲染（PBR）：用材质参数描述表面受光。

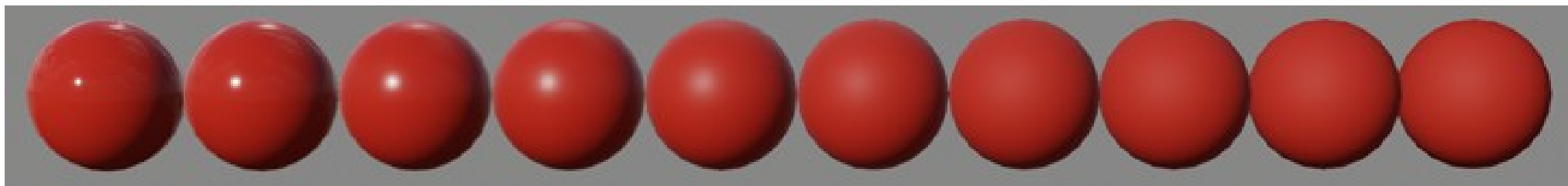
参数	改变什么	观察时关注什么
Base Color	表面的基础颜色	同一光照下的整体明暗与色彩
Metallic	表面是否按金属方式反射	高光和环境反射的特征
Roughness	反射是否集中或分散	镜面高光边缘是否清晰
Normal	不改网格轮廓的局部起伏	细小纹理会怎样改变受光细节

3.2.2 三维资产管理：粗糙度对照

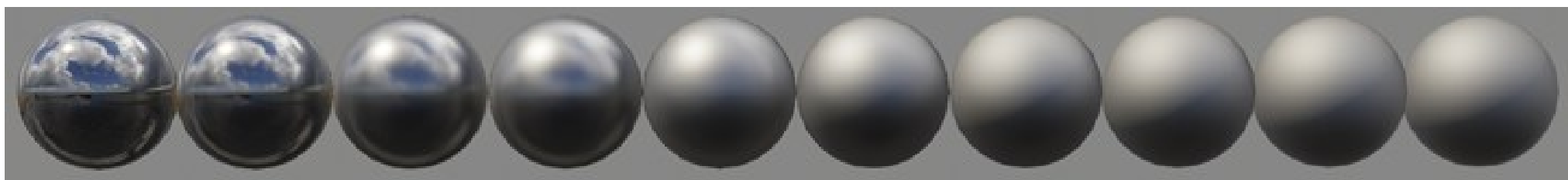
官方对照：粗糙度改变反射的集中程度

每一行从左到右：Roughness由0增加到1

非金属



金属



图：Epic官方材质样图；比较各行内部的高光和环境反射。

几何轮廓保持相同，粗糙度升高使高光与反射更分散。

3.2.2 三维资产管理：地形

Landscape用灰度高程图生成可编辑地形

地形系统 (Landscape)：读入灰度高程图生成三维地形，可叠加材质图层、绘制道路河流，并按视距自动调整细节。

高程图：每个像素是一个16位高度值（0~65535），32768对应零高度；Z缩放为100时高度范围约为-256 m~+256 m。

$$h = \frac{v - 32768}{128} \cdot \frac{s_z}{100} \text{ m}$$

一步算例： $v = 40960, s_z = 100 \Rightarrow h = (40960 - 32768)/128 = 64 \text{ m}$

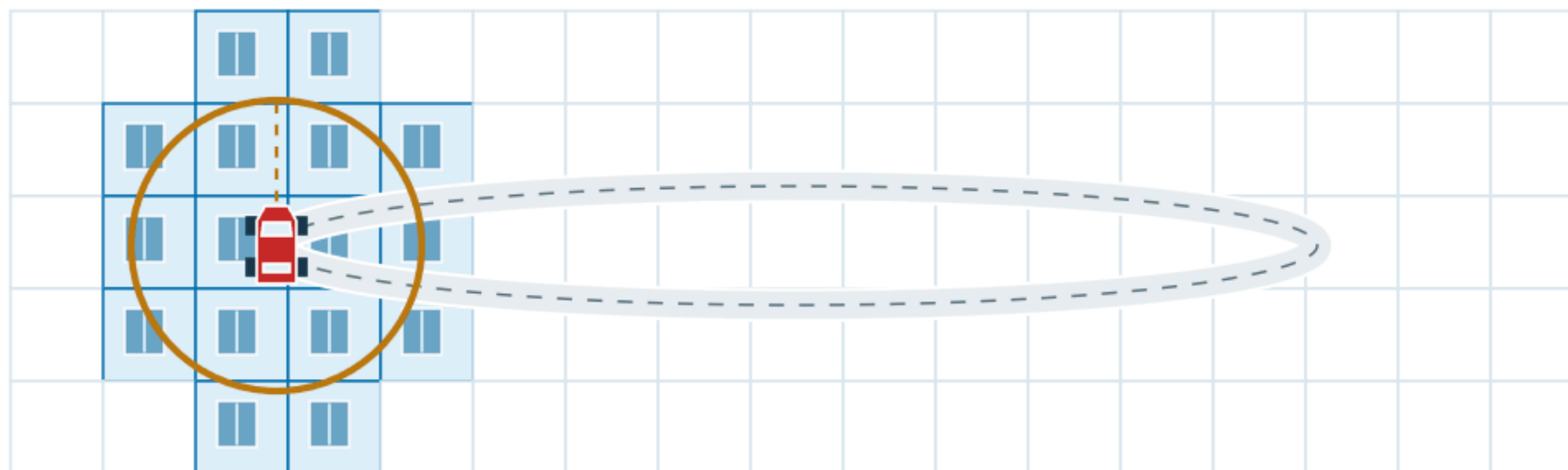
数据来源：数字高程模型（DEM）等地理高程数据可转成高程图导入。

高程零点与Z缩放若不匹配，导入后的地形高差会整体失真。

3.2.2 三维资产管理：大场景流送

世界分区按流送源位置加载周围的网格单元

与圆形范围相交的网格加载



已加载 16 格

- 流送范围
- 已加载
- 未加载
- 🚗 流送源

图：流送源移动时的网格加载与卸载（教学示意）

流送源附近网格加载，远处网格卸载；对象是否常驻还取决于自身设置。



3.2 UE场景运行渲染与交互



本章目录

UE场景：从对象状态到画面与交互

第三章 三维环境建模与仿真

3.1 三维仿真基础与工具选型

3.2 UE场景运行渲染与交互

3.3 多模态传感器数据生成

3.4 分布式仿真架构与数据接口

3.5 场景与载具资源开发

3.6 例程体系与综合实践

3.7 习题案例分析与课程设计

本节 5 个教学单元

3.2.1 框架与渲染流程

教材对应：第3.1.4.1节

3.2.2 三维资产管理

教材对应：第3.1.4.2节

3.2.3 光照渲染系统

教材对应：第3.1.4.3节

3.2.4 物理与动画系统

教材对应：第3.1.4.4节

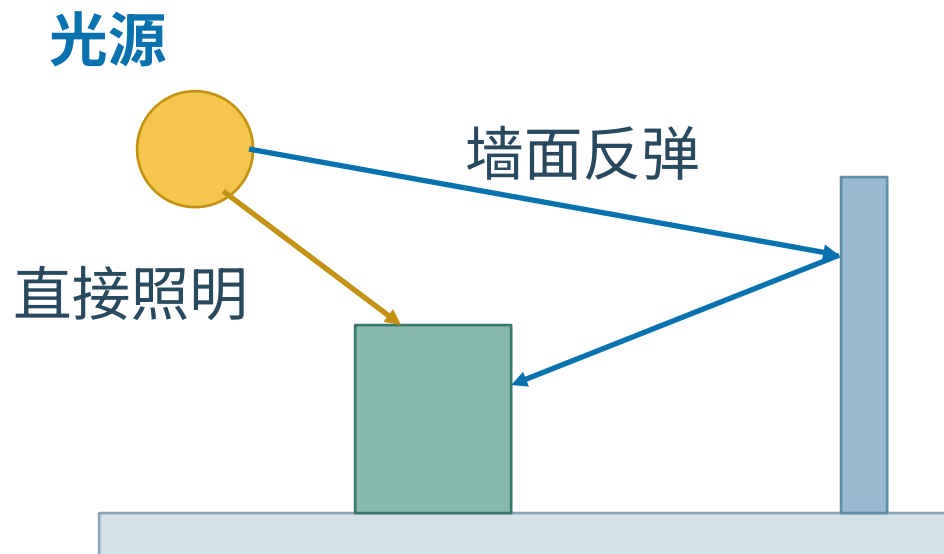
3.2.5 仿真交互框架

教材对应：第3.1.4.5节



3.2.3 光照渲染系统：直接光与间接光

直接光与反弹光走不同路径



图：直接光与反弹光路径（示意）

直接光：从光源到目标表面；中间有遮挡时，目标可能进入阴影。

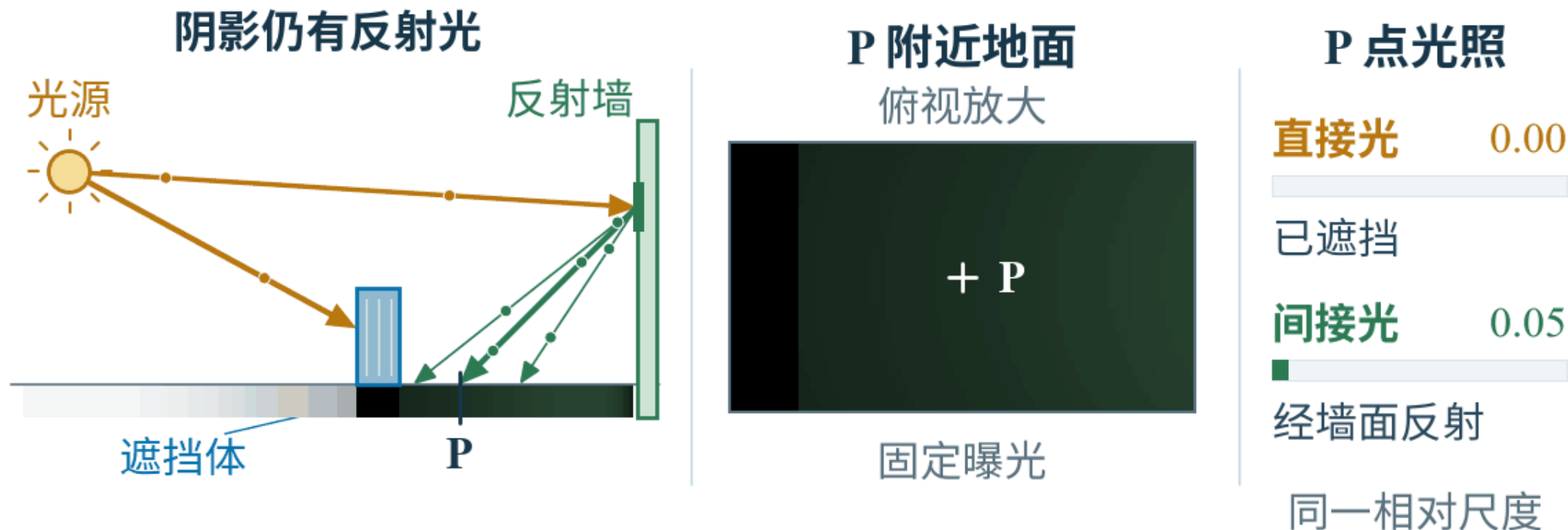
间接光：先照到其他表面，再反弹到目标；墙面颜色会影响反弹光。

间接光作用：照亮阴影区域，增加层次感；计算量大，常用预先烘焙的光照贴图或UE5的Lumen实时计算。

光源、遮挡、材质和曝光共同改变相机的像素值。

3.2.3 光照渲染系统：直接光与间接光

挡住直射光后，墙面反射仍能照亮物体



图：遮挡体运动改变直达光路；墙面一次反射仍可为P点提供照明

3.2.3 光照渲染系统：光源与可移动性

光源按运行时能否变化分为三类

光源类型：定向光模拟太阳，天光模拟天空，另有点光源、聚光灯和矩形灯光。

选择依据：固定场景用静态，只调亮度用静止，太阳要动或车灯随车就用可移动。

静态 (Static)：完全不变；编辑时预计算，最快

静止 (Stationary)：亮度、颜色可调；部分预计算，中等

可移动 (Movable)：位置、方向都可变；全部实时，最贵

光源越能在运行中变化，越依赖实时计算，渲染代价也越高。

3.2.3 光照渲染系统：光照贴图

Lightmap把静态光照预先算好存成纹理

光照贴图 (Lightmap)：把静态光源照到物体上的结果编码成纹理，运行时直接读取，省去实时计算；光源或物体移动后不会自动更新。

Lightmass：UE中负责这一预计算的系统；每个静态网格要有一套不重叠的额外UV存放光照。

一步算例：4 m宽的墙铺满 64×64 光照贴图时，每个纹素（贴图像素）约 $400/64 = 6.25$ cm，比它细的阴影边缘会被抹平。

平台对照：[安装目录]\RflySim3D\RflySim3D\Content中，Changsha.umap旁的Changsha_BuiltData保存该地图的构建数据，主要是光照。

静态光照预先烘焙进纹理，运行最省，但光源和物体一动就不再准确。

3.2.3 光照渲染系统: Lumen

Lumen在UE5中实时计算间接光和反射

Lumen: UE5的实时全局光照系统，自动随场景变化计算光线反弹，无需烘焙光照贴图或手动放置反射球。

追踪路径: Lumen先利用屏幕信息，再按项目配置采用距离场软件光追或硬件光追，估计间接光与反射。

反射: 随场景变化计算表面反射；玻璃反射与折射还取决于材质和渲染设置。

后处理与回退: 可配合景深、色彩校正、辉光等后处理；性能受限时关闭Lumen，改用光照贴图。

场景和光源常变用Lumen，场景固定、追求帧率用光照贴图。

3.2.3 光照渲染系统：阴影

级联阴影贴图让近处阴影清晰、远处节省资源

阴影的作用：没有阴影的物体像漂浮在场景中，阴影交代物体与地面的空间关系。

级联阴影贴图：按离相机的距离把阴影分成几层，近层用高分辨率，远层用低分辨率。

一步算例：同为2048像素的阴影贴图，近层覆盖20 m、每像素约 $20/2048 \text{ m} \approx 0.98 \text{ cm}$ ；远层覆盖200 m，每像素约9.8 cm。

调参入口：定向光的动态阴影距离和级联数量决定各层覆盖范围与精度。

取舍：远处阴影变粗但不易察觉；无人机下视相机近地飞行时，近层清晰度最重要。

近层阴影用高分辨率保细节，远层降低分辨率省资源。

3.2.3 光照渲染系统：反射

反射效果按场景动态程度和平台选择实现方式

反射：光滑表面可映出环境；粗糙水泥墙仍反射光，但通常没有清晰倒影。

Lumen反射：随场景实时计算反射；动态场景，UE5

屏幕空间反射：只用当前画面中已有的像素；动态，画面外物体映不出

反射捕捉器：预先拍下所在位置的环境；场景较静态

光线追踪：沿反射方向追踪虚拟光线路径；质量和开销取决于实现与采样

反射方案要按场景变化、画面外信息与可用算力选择。



3.2 UE场景运行渲染与交互



本章目录

UE场景：从对象状态到画面与交互

第三章 三维环境建模与仿真

3.1 三维仿真基础与工具选型

3.2 UE场景运行渲染与交互

3.3 多模态传感器数据生成

3.4 分布式仿真架构与数据接口

3.5 场景与载具资源开发

3.6 例程体系与综合实践

3.7 习题案例分析与课程设计

本节 5 个教学单元

3.2.1 框架与渲染流程

教材对应：第3.1.4.1节

3.2.2 三维资产管理

教材对应：第3.1.4.2节

3.2.3 光照渲染系统

教材对应：第3.1.4.3节

3.2.4 物理与动画系统

教材对应：第3.1.4.4节

3.2.5 仿真交互框架

教材对应：第3.1.4.5节



3.2.4 物理与动画系统：运动来源

运动求解、碰撞查询与动画分工

物理求解： 启用相应物理行为后，质量、力、约束和碰撞形状参与UE中的运动与接触计算。

碰撞查询： 按形状和通道检查命中或重叠，可以只返回查询结果，不必驱动刚体运动。

部件动画： 更新旋翼或起落架的变换，产生连续可见动作；动作规则来自蓝图或动画系统。

课程平台路线： CopterSim可计算载具运动并向三维端发送状态；旋翼显示转动本身没有计算推力。

动画改变显示部件；动力学由选定的运动模型计算。

3.2.4 物理与动画系统：动画系统

骨骼、动画蓝图与Control Rig逐层驱动动作

骨骼 (Skeleton)：定义角色或机械载具的关节层级和位置，是动画变形的基础。

动画蓝图 (Animation Blueprint)：可视化状态机，按速度、方向、状态等参数混合与切换动画序列。

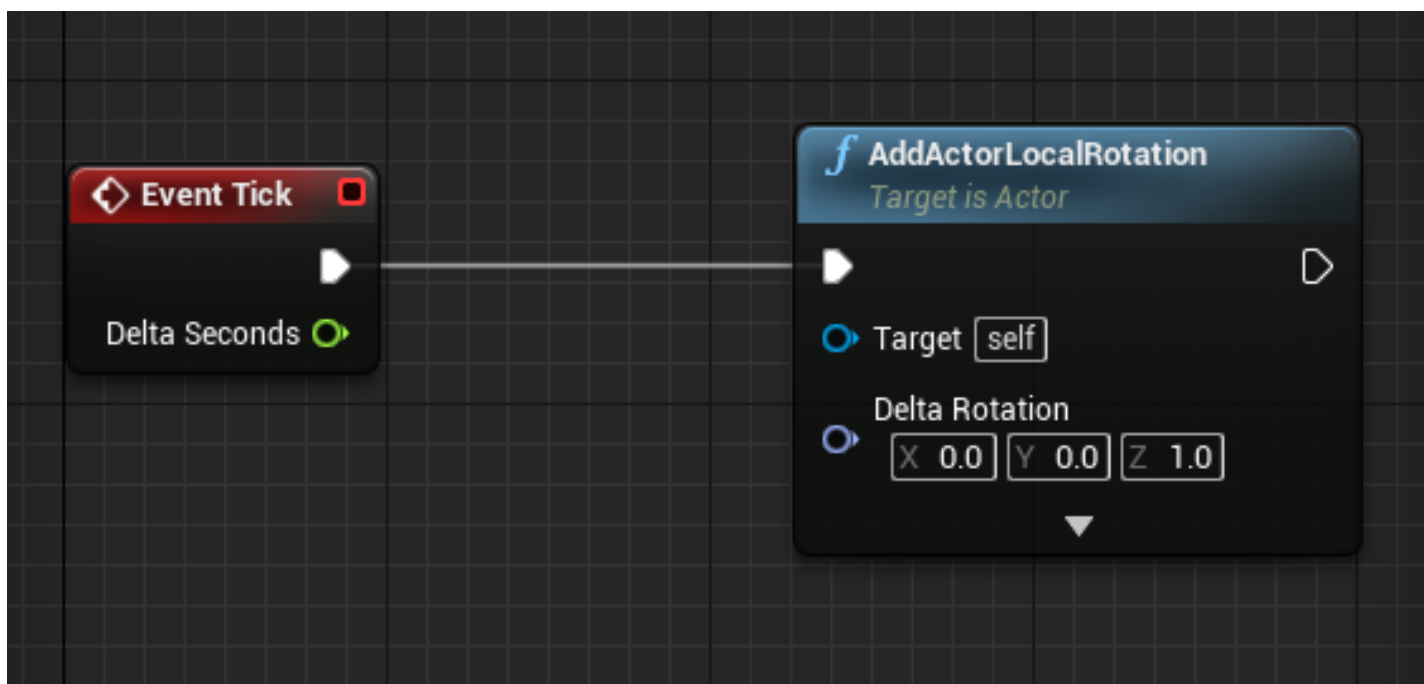
控制绑定 (Control Rig)：在编辑器和运行时生成或修改动作，可加约束与逆运动学（由末端位置反推关节角度）求解。

起落架状态机：离地高于2 m切到“收起”，降到1 m以下切回“放下”；两个阈值不同可防止临界处来回抖动。

关节层级与状态切换分开设计，复杂动作才便于修改。

3.2.4 物理与动画系统：固定逐帧增量

盒体原例：每次Tick增加1度



图：盒体Tick与局部旋转节点

执行线： Event Tick调用AddActorLocalRotation，Target为自身。

角增量：

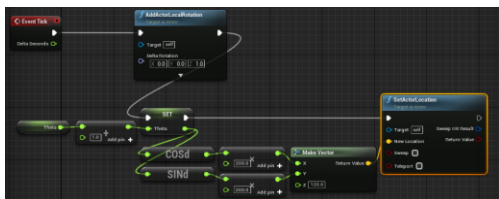
Delta Rotation的Z为1，绕局部Z轴增加1度。

时间输入：

DeltaSeconds未接入；
30FPS约30度/秒；
60FPS约60度/秒。

3.2.4 物理与动画系统：正余弦与位置

盒体原例：角度同时驱动圆周位置

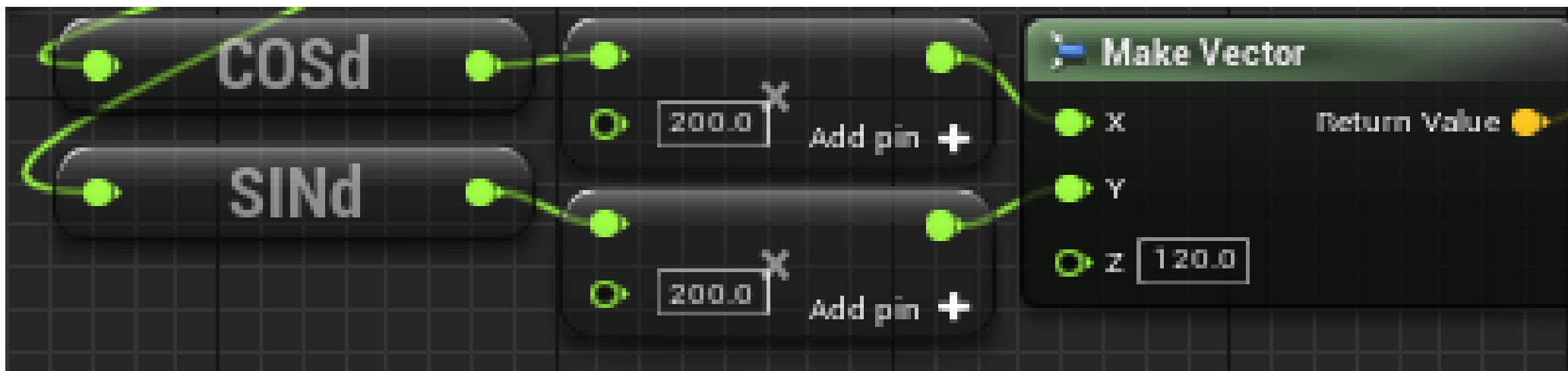


图：总图与位置局部

状态： Theta每次加1；COSd/SINd按角度制。

位置： $X = 200 \cos \theta$, $Y = 200 \sin \theta$, $Z = 120$, θ 为变量Theta。

几何： 单位cm； $\theta = 90^\circ$ 时位置为(0, 200, 120)。



每帧更新Theta，再由正余弦计算世界位置。

3.2 UE场景运行渲染与交互

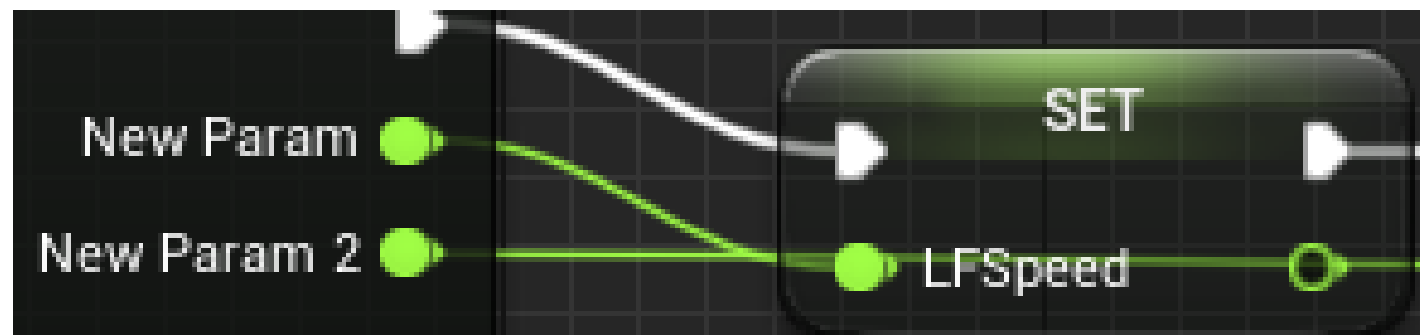
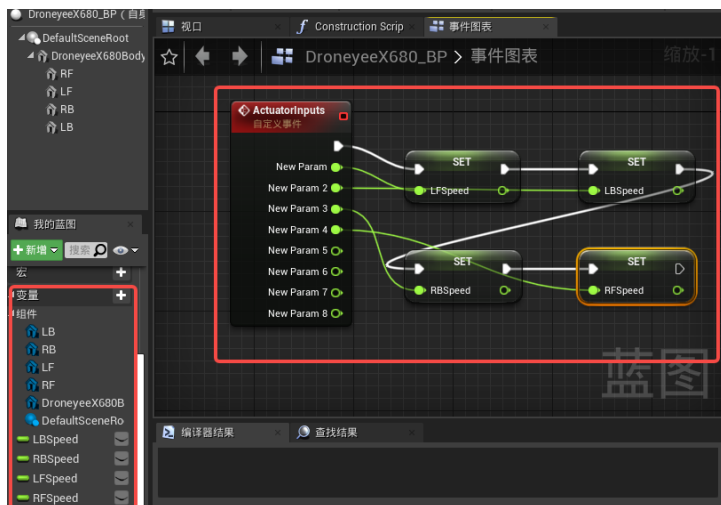
3.2.4 物理与动画系统：事件到变量

X680输入事件先保存四个旋翼速度

事件约定： ActuatorInputs接收8个float参数；本例前4路只保存速度。

目标： 四个旋翼依附机身。

局部： 第1路输入 → LFSpeed



图：组件树与事件总图

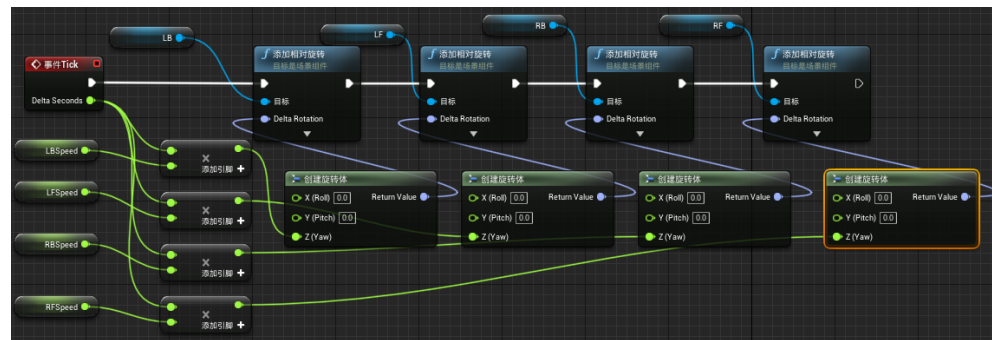
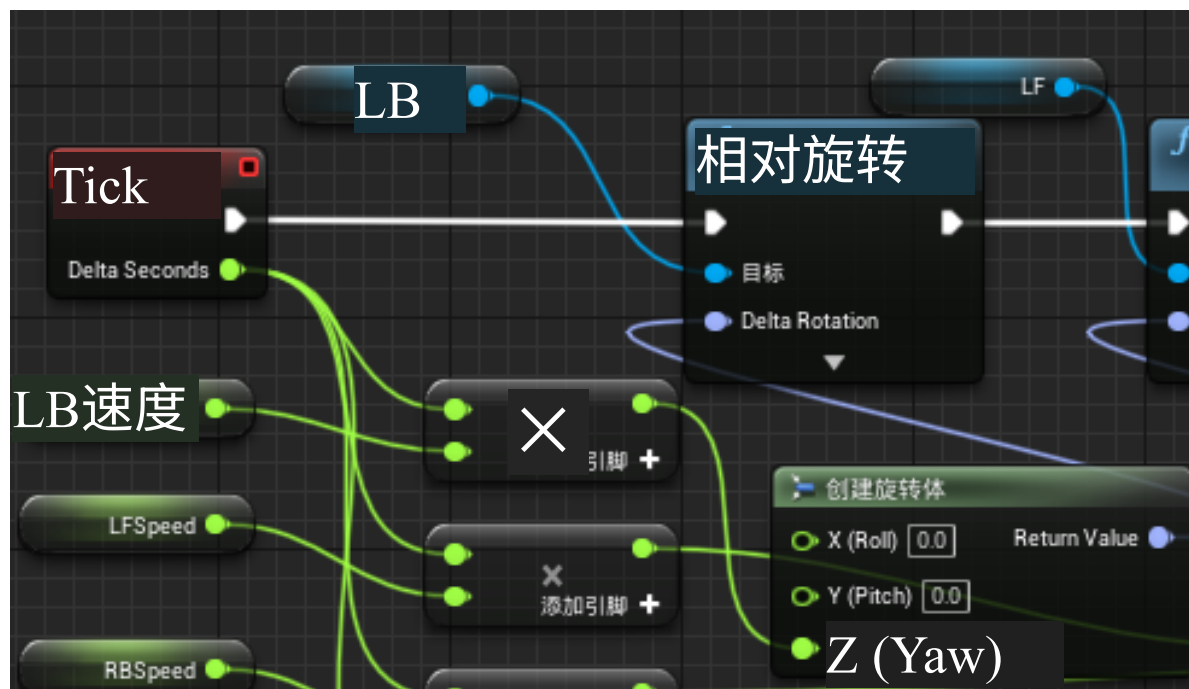
1→LFSpeed / LF左前 2→LBSpeed / LB左后
3→RBSpeed / RB右后 4→RFSpeed / RF右前

ActuatorInputs接收参数，速度变量供后续Tick读取。

3.2.4 物理与动画系统：时间到部件旋转

X680每帧用速度乘时间计算角增量

图：LB单路放大；右为四路总图



$LB\text{Speed} \times \Delta\text{Seconds}$
 $\rightarrow$ Make Rotator的Yaw
 $\rightarrow$ LB: AddRelativeRotation

Yaw角增量： $\Delta\psi = \omega \Delta t$ ， ω 为度/秒， Δt 为秒。

本帧角增量由角速度乘DeltaSeconds得到，再到对应旋翼上。

3.2.4 物理与动画系统：帧率对照算例

同样转一秒，两种更新规则结果不同

教学题设：Tick逐帧执行，分别取30FPS与60FPS；非性能实测。

更新规则	30FPS	60FPS
固定1度/帧	30度/秒	60度/秒
指定60度/秒	每帧增加2度	每帧增加1度
按时间累计一秒	60度	60度

计算： $60^{\circ}/s \times \frac{1}{30} s = 2^{\circ}$ ； $60^{\circ}/s \times \frac{1}{60} s = 1^{\circ}$ 。

固定每帧角度随帧率变速；速度乘时间保持约定角速度。

3.2.4 物理与动画系统：逐帧规则对照

同一秒内，固定增量随帧率变化，按时间累加不变

终点：同一秒后

30 FPS

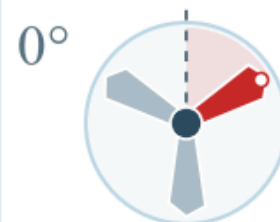
60 FPS

固定每帧
1°/帧



30°

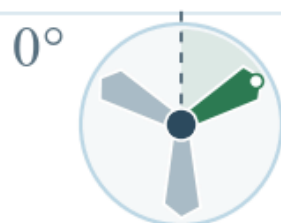
30 帧



60°

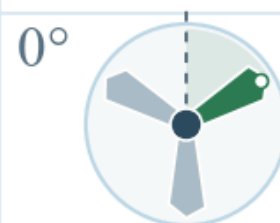
60 帧

按时间累加
 $60^\circ/\text{s} \times \Delta t$



60°

30 帧



60°

60 帧

$t = 1.00 \text{ s}$

0 s

1 s

图：相同模型时间内，两种Tick规则产生的转动差异

固定每帧加1°会随帧率变速；角速度乘帧间隔仍保持每秒60°。

3.2.4 物理与动画系统：蓝图与C++协作

蓝图调用能力由项目实际暴露的接口决定

本例蓝图： ActuatorInputs保存速度，Tick计算角增量，AddRelativeRotation更新指定旋翼组件。

C++扩展： 可实现协议接收、设备访问或可复用计算，再按项目需要暴露为蓝图可调用能力。

接口条件： 空白UE工程不会自动拥有RflySim约定事件；需要匹配的插件、模型和资源注册路线。

资源烘焙（Cook）： 把编辑器资产转换为运行资源；课件3.5再讲按平台UE版本烘焙与注册。

蓝图组织事件与状态，底层接口把具体能力接入节点。

3.2.4 物理与动画系统：碰撞检测

碰撞检测先粗筛候选对，再精确计算接触

宽相检测 (Broad Phase)： 用轴对齐包围盒 (AABB) 或包围球快速排除不可能相碰的物体对。

窄相检测 (Narrow Phase)： 按候选对的碰撞形状精确计算接触，得到接触点、法线方向和穿透深度。

一步算例： AABB相交要求三轴都满足 $a_{\min} \leq b_{\max}$ 且 $b_{\min} \leq a_{\max}$ ；x轴 $[0, 2]$ 与 $[1.5, 3]$ 重叠，y轴 $[0, 2]$ 与 $[2.5, 4]$ 不重叠，直接排除。

分两步的原因： 1000个物体有 $n(n-1)/2 = 499500$ 对，逐对精算太慢；层次包围盒 (BVH) 或八叉树按空间分层，只比较相近的对。

宽相用包围盒快速排除，窄相只对候选对精确求接触。

3.2.4 物理与动画系统：碰撞设置

碰撞设置决定物体参与查询、物理还是两者

碰撞通道： WorldStatic对应静止物体， WorldDynamic对应靠代码或动画移动的物体。

无碰撞： 不参与查询，也不参与物理；纯装饰物、远景

仅查询： 响应射线、扫掠和重叠检测；触发区、测距射线

仅物理： 只参与刚体与约束模拟；只需模拟运动的碎片

查询与物理： 两者都参与；会被撞、也要被检测的物体

只需回答“碰到了吗”时用仅查询，需要被撞动时才启用物理。

3.2.4 物理与动画系统：碰撞响应

两个物体的碰撞响应共同决定阻挡、重叠或忽略

响应：每个物体对每个通道设为阻挡（Block）、重叠（Overlap）或忽略（Ignore）。

A、B分别表示两个物体对对方通道的响应。

A阻挡+B阻挡：互相阻挡，可产生命中事件

A阻挡+B重叠：仅重叠，不阻挡

A重叠+B重叠：重叠；双方需开启重叠事件

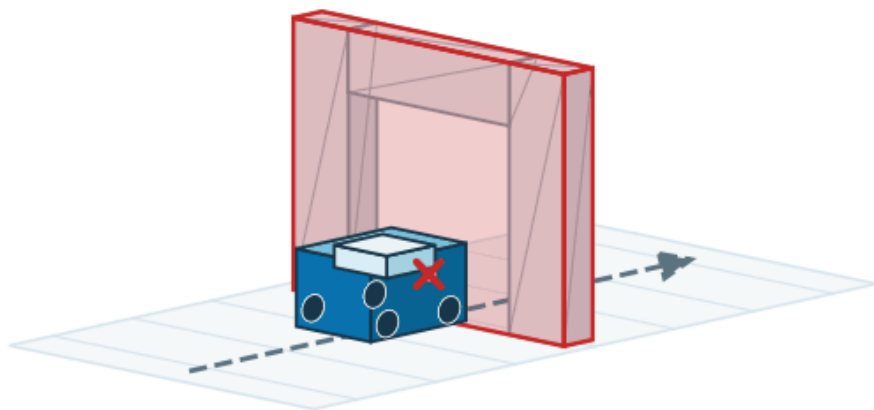
任一方忽略：不阻挡，也不触发重叠

双方都设阻挡才会挡住；任一方忽略，就不发生交互。

3.2.4 物理与动画系统：渲染与碰撞分离

看得见门洞，也可能被碰撞形状挡住

整块碰撞盒：挡住门洞

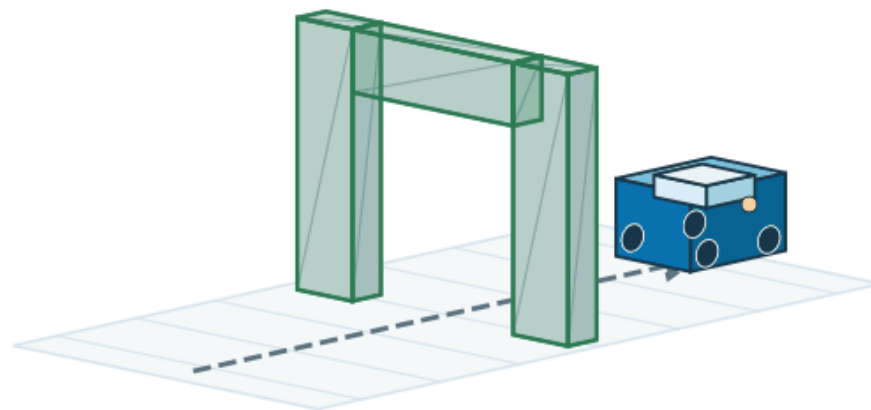


查询命中：停止

渲染网格

碰撞形状

门框碰撞形状：保留门洞



查询无命中：通过

路径查询，不计算碰撞力

图：外观相同，碰撞形状不同，通行查询结果不同

渲染网格、LOD与碰撞形状分别配置，通行结果由碰撞查询决定。



3.2 UE场景运行渲染与交互



本章目录

UE场景：从对象状态到画面与交互

第三章 三维环境建模与仿真

3.1 三维仿真基础与工具选型

3.2 UE场景运行渲染与交互

3.3 多模态传感器数据生成

3.4 分布式仿真架构与数据接口

3.5 场景与载具资源开发

3.6 例程体系与综合实践

3.7 习题案例分析与课程设计

本节 5 个教学单元

3.2.1 框架与渲染流程

教材对应：第3.1.4.1节

3.2.2 三维资产管理

教材对应：第3.1.4.2节

3.2.3 光照渲染系统

教材对应：第3.1.4.3节

3.2.4 物理与动画系统

教材对应：第3.1.4.4节

3.2.5 仿真交互框架

教材对应：第3.1.4.5节



3.2.5 仿真交互框架：输入系统

输入系统把设备信号映射成控制器能处理的动作

输入映射 (Input Mapping)： 把不同设备的按键和摇杆统一映射为游戏内的事件或命令，屏蔽硬件差异。

事件驱动： 按下、松开等变化触发事件；摇杆等连续量按帧读取当前状态。

平台实例： RflySim3D中按F1弹出帮助，按B切换跟随的飞机，按V切换机载相机，双击地面显示击中点信息。

设备差异由输入映射处理，控制逻辑接收统一的动作意图。

3.2.5 仿真交互框架：控制关系

控制器通过占有关系连接受控实体

受控实体 (Pawn)：教材称棋子或角色体，是能被控制器占有的Actor。

角色 (Character)：带常用角色运动能力的Pawn子类，适合行走、跳跃等角色行为。

控制器 (Controller)：玩家控制器处理键盘、手柄等输入；
AI控制器按规则决策。控制器自身不显示网格。

占有关系 (Possess)：建立Controller对Pawn的控制；
Unpossess解除；默认一对一，换对象先解除再占有。
运动仍由实体及组件实现。

控制器处理意图，受控实体及其组件实现运动。

3.2.5 仿真交互框架：规则与全局状态

游戏模式定规则，游戏状态存全局，HUD显示反馈

戏剧模型： Actor像演员， Pawn像被操纵的角色， 游戏模式像统一的剧本。

界面搭建： 虚幻运动图形（UMG）用控件拼出菜单、仪表等交互界面。

游戏模式（Game Mode）： 定义规则与流程；起飞条件与任务结束

游戏状态（Game State）： 记录全局状态；任务进度与得分

平视显示器（HUD）： 显示关键实时信息；高度、速度与告警

游戏模式定规则，游戏状态记进度，HUD和UMG把信息显示给人。

3.2.5 仿真交互框架：输入与反馈

从控制意图追踪到场景状态



正向：控制意图交给Pawn与运动组件，计算位移并更新场景状态。

反馈：相机或状态接口读回结果，控制逻辑据此更新下一次输入。

控制意图改变状态，相机或状态观测再返回控制逻辑。

3.2.5 仿真交互框架：AI行为逻辑

行为树、黑板与感知组成智能体的决策回路

行为树：可视化决策流程图，由选择、序列和任务节点组成，按条件依次执行移动、搜索等动作。

黑板：行为树的数据中心，保存目标位置、目标是否可见等信息，供条件判断读取。

感知系统：提供视觉、听觉等感知，把发现的目标写入黑板，驱动行为切换。

导航网格（NavMesh）：根据场景自动生成可通行区域，用于路径规划和避障，可动态更新。

大规模实体（Mass Entity）：UE5的轻量架构，可同时控制上千个智能体，用于人群、交通和编队仿真。

感知写黑板，行为树读黑板做决策，导航网格负责走到目标。

3.2.5 仿真交互框架：网络与多智能体

服务器权威、属性复制和RPC保持多机状态一致

网络驱动 (NetDriver)： 管理客户端与服务器的连接和数据传输，支持各参与方的状态同步。

属性复制 (Replicated)： Actor已开启复制且满足相关性与复制条件时，服务器按调度向相关客户端同步最新状态。

远程过程调用 (RPC)： 本地调用、远端执行的函数，如客户端请求开始任务。

一步推演： 服务器把2号无人机的位置改为(100, 0, 50) m，满足上述条件的客户端收到更新后显示该位置，并非同时到达或逐次变化都送达。

Iris： UE5新增的网络复制系统，面向上百客户端的大规模多智能体仿真。

服务器保存权威状态，复制把变化推给客户端，RPC传递一次性请求。



3.2 UE场景运行渲染与交互



学习资源

对象学习：实体、组件与控制关系

Actor：根组件、变换与实例生命周期

<https://dev.epicgames.com/documentation/en-us/unreal-engine/actors-in-unreal-engine>

Component：逻辑组件与空间依附关系

<https://dev.epicgames.com/documentation/en-us/unreal-engine/components-in-unreal-engine>

Controller：Possess与Pawn控制关系

<https://dev.epicgames.com/documentation/en-us/unreal-engine/controllers-in-unreal-engine>





3.2 UE场景运行渲染与交互



学习资源

节点实践：盒体到X680的时间更新

Tick：时间间隔、更新分组与先后依赖

<https://dev.epicgames.com/documentation/en-us/unreal-engine/actor-ticking-in-unreal-engine>

盒体Readme：组件、每帧1度与圆周位置

https://rflysim.com/doc/zh/px4psp/RflySimAPIs/3.RflySim3DUE/0.ApiExps/e0_DevToolsUsage/1.UEBlueprintUsage/Readme.pdf

X680 Readme：ActuatorInputs与速度乘时间

https://rflysim.com/doc/zh/px4psp/RflySimAPIs/3.RflySim3DUE/2.AdvExps/e2_BlueprintModel/1.BlueprintBuild/Readme.pdf





3.2 UE场景运行渲染与交互



学习资源

成像学习：材质、光照与渲染路径

PBR：颜色、金属度与粗糙度对照图

<https://dev.epicgames.com/documentation/en-us/unreal-engine/physically-based-materials-in-unreal-engine>

前向渲染：与延迟路径的能力取舍

<https://dev.epicgames.com/documentation/en-us/unreal-engine/forward-shading-renderer-in-unreal-engine>

Lumen：UE5动态间接光与反射的适用条件

<https://dev.epicgames.com/documentation/en-us/unreal-engine/lumen-global-illumination-and-reflections-in-unreal-engine>





3.2 UE场景运行渲染与交互



学习资源

性能与交互：读指标并解释碰撞

Stat命令：Frame、Game、Draw与GPU指标

<https://dev.epicgames.com/documentation/en-us/unreal-engine/stat-commands-in-unreal-engine>

碰撞复杂度：简单形状与三角网格的使用条件

<https://dev.epicgames.com/documentation/en-us/unreal-engine/simple-versus-complex-collision-in-unreal-engine>

控制台变量：帧率上限与强制LOD设置

<https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-console-variables-reference>



课堂练习

用本节方法分析一架无人机的仿真画面与交互

帧耗时: Game 12 ms、Draw 8 ms、GPU 21 ms时整帧约多少？先优化哪一段？

旋翼转角: X680的RF旋翼命令值 $120^\circ/\text{s}$ ，帧率从60降到40FPS时每帧转多少？一秒共转多少？

碰撞设置: 门洞渲染网格有开口，碰撞却是整块盒子，无人机穿门时会怎样？应改哪一项设置？

光照选择: 模拟上午到傍晚的巡检，太阳应设哪种可移动性？代价是什么？

地形换算: 高程图像素值30720、Z缩放100时，该点高度是多少米？

先写出依据的规则或公式，再代入数值，最后说明结果含义。

参考要点

练习参考要点：先写规则，再代入数值

帧耗时：三段流水并行，整帧接近最慢的GPU，约21 ms、约48FPS；应先减少GPU工作。

旋翼转角：每帧 $120/40 = 3^\circ$ （60FPS时为 2° ），一秒仍为 120° ，按时间累加与帧率无关。

碰撞设置：查询命中整块碰撞盒，无人机被挡住；应把碰撞改为贴合门框的形状或使用复杂碰撞。

光照选择：太阳用可移动光源，阴影随时间变化；代价是实时计算、帧耗时增加。

地形换算： $(30720 - 32768)/128 = -16$ ，高度为-16 m，即低于零高度16米。

流水看最慢段，转角看角速度，碰撞看形状，光照看可移动性。

本节小结

从对象状态到画面与交互的五条主线

框架与渲染：交互框架每帧更新状态，CPU剔除并提交，GPU完成顶点变换、光栅化、延迟光照和后处理。

资产与光照：资产被组件引用；LOD、Mipmap与世界分区控制细节和加载；光照在预计算与实时之间取舍。

物理与交互：动画只改显示，碰撞由查询与双方响应决定；输入经映射交给控制器，规则与全局状态由专门对象管理。

易错点：帧耗时不能相加；每帧固定增量会随帧率变速；看得见的开口不等于碰撞能通过；属性复制只从服务器发往客户端。

每帧先定状态再成像，读懂耗时、光照与碰撞才能解释仿真结果。