书游戏场景开发与设计

小概念

带宽:资源的输入，输出的传输速度
点构成的运用场景:魂系游戏的水晶篝火等
线构成:引导路线，强化体积光影，
面构成→节奏
打光的意义:通过明暗识别物体
打光方法:剪影；体量；结构
打光法:伦勃朗光；显瘦光；显胖光；三点式打光法(主光辅助光边缘光)；明暗对照法(背景衬托)
GI 包含间接，直接光照，当烘焙光照贴图以后动态物体会采用光照探针
光比:亮暗受光比例
中灰:线性空间 18%灰度值
EV:曝光值，测试光比默认 0
IBL(Image Based Lighting):基于图像的光照，光照地图是 360 度全景信息(高光部分计算)
场景生产流程 工业化管线
前期准备:1 🟦项目目标 2 🟦项目元素 3 🟦关卡流程
确立玩法机制与体验目标
确定游戏节奏
确立世界观:地图，时间线，其他
确立落地可行性:项目体量；开发管线；技术验证
关卡设计原则:游戏循环，游戏进程(理性游戏设计)，游戏指引
layout:平面布局图，包括 1 🟦动线设计 2 🟦设计方针 3 🟦分区分类 物件类型 路径类型 区域类型
layout 设计方法:局部到整体；气泡填充法；平面布局图；
↓
路径设计法:欲望路径；关键路径；流转路径
blockout 白盒测试:
体量:物体体积质量传达出的一般感受
比例尺:角色与人物的比例关系 1 🟦玩家操作比例 2 🟦玩家与场景物体比例
如何确立场景物件比例:1 🟦体块法 2 🟦组件法
↓
demo 阶段:
1 🟦确立美术风格
2 🟦资产规范

场景原画→场景资产规划→物件原画→正式生产→灯光布置与烘焙

场景实现常用技术

一 pbr

基于物理的灯光，相机，材质三大模块，涉及完整的资源制作管线
pbr 流程:不包含光影，包含 albedo map 反射率贴图，金属和粗糙法线(金属工作流和高光工作流)
pbr 原理:精确模拟以接近现实世界，动态模拟人眼看见物体的过程(光源照射到材质上→根据材质本身属性进行光反射→相机接收反射光，显示最终结果)
BRDF 核心:微表面理论(粒子碰撞发生散射)和能量守恒(入射光线=反射光线+被吸收光线)
迪士尼原则:艺术导向，不完全物理正确；直观参数，非物理参数；参数精简；参数的合理范围 0-1；参数有意义时超过正常范围；参数组合应该严谨合理
BRDF 公式 $fr=k^{d}\rho/\pi+DFG/cos\theta^{i},cos\theta^{o},^{n}$
输出颜色=漫反射比例 x 表面颜色/π + 镜面反射比例 x 法线分布函数 x 几何函数 x 菲涅尔系数/4
x(视线方向 x 法线)(光线方向 x 法线) x 光源颜色(光线方向 x 法线) DW
↓简化为
输出颜色=漫反射+镜面反射
↓
输出颜色=(直接光+IBL) x 基础颜色贴图 x 金属度贴图值+(直接光+IBL) x 镜面反射颜色 x 金属度贴图值
高光↓
D 项法线分布函数(镜面分布函数):粗糙度相关，微表面与 h 向量方向一致的向量比率
F 项菲涅尔函数:被反射光线与被折射光线的比率
G 项几何函数:凹凸不平微表面之间互相遮挡的比率，粗糙度相关
高光工作流:优点同时包含金属和非金属高光效果并且实现带颜色的高光，缺点贴图无法压缩不易控制，性能开销大
金属贴图不能是灰色的原因:乱用，没有标准，可以用的例子，金属被灰尘覆盖的混合材质→守望先锋
为什么基础贴图不能画光源叠 ao:一个是材质信息，一个是光照信息

二 npr 非真实渲染，风格化渲染(stylistic rendering)

次世代卡通风格
npr 基础实现方案:
1 🟦光照:BlinnPhong 模型，通过 step 或者 smoothstep(抗锯齿只能用它)或者 rampmap 实现 tone-maping 获取清晰高光及阴影边缘；
边缘光实现方法:菲涅尔漏光➕烘焙球谐函数(spherical harmonics)或者深度偏移

对 DFG 进行数值操作
2 🟦描边方案
①法线外扩，渲染背面剔除，断开，储存平滑法线(存于顶点色里或者到屏幕空间)
②菲涅尔描边
③后处理描边边缘算子；深度法线描边；ID 图识边(先绘制彩色贴图再边缘算子)
④本村线 画线位置的 UV 变直，避免斜向采样，缺点是无法绘制文字或者相对复杂图案

三草，树，动画

草🌿:
模型草:优点，不透明，减少半透明计算，缺点对面数要求高，两种类型 摆放模型和几何着色器生成模型
模型草实现过程:摆放模型，dcc 制作，根据模型高度插值混合颜色；几何着色器模型:GPU 计算网格模型，通过遮罩控制草的疏密程度。
面片草:优点，面数少，形状丰富，缺点半透明渲染，重复渲染计算量
面片制作过程:制作交叉网格模型绘制贴图纹理
面片草，面皮草的混合模型通常取决于项目视角，自由度高选择剪裁模式(高配则会开启抗锯齿的后处理计算)，自由度低开启混合模式，依照顶点 ID 排序
模型草面数在面片草十倍以内，模型草性能更优
树🌲:
工具制作
网格模型➕面片
树模型➕houdini 根据顶点生成树叶(风格化植被使用):优点，加入实体模型减少半透明渲染
树叶渲染:需要考虑透光性和反光性，透光性(次表面散射和暗部特殊处理)，反光性(正常 pbr 光照模型) 优先使用剪裁模型和双面渲染
树干渲染:正常 pbr 流程，加入高度图和视差映射或者曲面细分和顶点置换
植被动画:
实现过程
顶点动画 sin abs floor 函数的随 time 变化的运动曲线，配合模型高度作为运动强度遮罩
↓
考虑植被末梢识别，遮罩图或者使用 pivot painter2 将主次枝干的层级信息记录到顶点和 UV 里面
↓
风场
↓
交互，角色世界位置传入植被，计算角色圆形碰撞范围，加入扰动,(缺点植被扰动应该有一个由强到弱的渐变，不因为角色离开改变，且角色兼容有限制) →改善 加入拖尾粒子系统赋予圆形凸起法线贴图，单独渲染到一张贴图里，传入着色器

↓
拖尾效果作为风场数据与风场结合
↓
延伸 植被与技能交互，使用渲染贴图用于渲染阈值并带有法线信息的半透明面片，传入植被材质运动里

四水体

湖泊:
面片，地形绘制遮罩作为水深判断，计算水面整体效果(或者深度图)
河流:
加入 flowmap，顶点色存储
海洋:
网格进行切块处理或者根据相机的可视范围动态创建海洋需要的面片
⚠ 优先生成地形:侵蚀或者绘制曲线图

场景优化原理技术

性能优化:标准，测试，优化
性能优化的核心原则是平衡画面，成本，性能
标准
测试收敛数据:测试性能数据以后对各模块进行详细性能开销拆分，让性能曲线收敛于性能目标下
各项管线优化
性能优化的原理:
CPU
GPU:合理控制渲染层
为什么移动端上剪裁模式性能大于混合模式?
因为剪裁模式会使硬件的 earlyz，binningpass 失效，使得全屏幕像素或者屏幕块失效，让优化操作被废弃
手机端带宽特指内存带宽，更多用到缓存带宽(CPU，GPU 与其缓存之间的数据传输速率)
存储带宽指的是存储设备与 CPUGPU 的速率

优化方案:

1 🟦绘制命令:

①**动态合批 dynamic batching**(将相同材质的多个模型合并，到临时缓冲去，一次 drawcall 绘制）仅 unity 支持，且部分做过优化的手机会增加负担，且经过位置选择的每帧都要计算，要求严格
②静态合批 unity 特有
③GPUInstancing:CPU 只提交一个网格和材质但是多个实例。UE 里是静态，渲染时无法剔除单个个体，移动端无法设置单个实例参数，unity 动态和 UE 静态的区别是 unity 需要耗费和

组装缓冲区性能去换取更少实例渲染。
优点:高效渲染，节省内存，可扩展性，灵活性，兼容性
影响带宽的主要因素
1 🟦贴图纹理(纹理格式建议 ASTC Adaptive Scalable Texture Compression 自适应可扩展纹理)
①纹理格式:ASTC 优点:高质量压缩；可自定义压缩块大小；多种纹理类型；兼容性高；节省资源；减少 CPU 对贴图的整理过程，直接进入内存
②纹理分辨率
③Mipmap Multum in Parvo map 多重细节层次映射

2 🟦纹理过滤方式 filter mode

开启各向异性过滤 Anisotropic 会提升显示效果但是 GPU 带宽会上升
为什么?因为这样会导致采样点增多，缓存丢失概率变高(三线性过滤同样)
建议双线性过滤

3 🟦重复绘制 overdraw

①正常绘制优化核心点在于模型在屏幕里重叠的概率减少
②四边形重复绘制 quad UE 提供了四边形重复绘制可视化，应该避免超过绿色的重复度

4 🟦内存 根据目标机型确立

手机优化方法纹理；动作网格模型；音频；着色器
5 🟦CPU/GPU 计算量优化
例子:GPU 动画系统
①CPU 计算量:谨慎使用后处理，如果发现 Graphics.Blit 消耗大应该降低分辨率或者减少后处理；
大量实例化以后要善用对象池降低压力；
get/find 不要在 update 里使用，暂存；
文本使用二进制，反序列化对象减少文本消耗
②GPU 计算量:指令数，贴图采样数量
常用方法:用 lut 替代复杂函数，如 color grading；
使用静态光照和阴影，烘焙光照和阴影贴图，少实时多光源；
将线性结果挪到顶点着色节点(因为顶点数量低于像素量，所以可以降低计算量)；
lod 技术；
合理使用高消耗函数；
合并贴图数据(如 mix 贴图)；

LOD 技术:可用于不同配置的机型，手机电脑分别匹配不同

1 🟦网格简化

面数为前一级 50%
UE 和 unity 的 lod 切换默认是根据物体在屏幕里占比进行的
UE 包含两种技术:
HLOD Hierarchical Level of Detail 分层细节级别 →可以组织多个静态网格体 actor 成单个代理网格体和远距离材质
Proxy Geometry Toll 代理几何体→可以合并成单个静态网格体单组纹理单组材质，减少三角形面数

2 🟦材质 shader 简化

Mipmap 贴图

3 🟦公告板技术 billboard

基于图像的优化技术，将远距离物体用预渲染图形替代，常用于植被的远景 lod 优化
优点:提升性能；节约内存；成本低，流程简单
缺点:画面质量有限；视角依赖(可通过固定 y 轴改善)；视觉不连续；遮挡和深度排序有误；烘焙阴影下无法判断位置处理阴影

4 🟦Impostor 技术

升级 billboard，进行多角度预渲染，结果存储在序列图，在着色器通过序列图排布公式进行序列图采样
过程:四边形模型拍摄创建 4x4 的倍数的角度相机矩阵，生成一组包含透明通道的图片→组合成符合目标进度的贴图→使用 impostor 适配的材质，lod3 或者 4，手机端建议 4x4 或者 8x8 角度，精度 1024x1024
优点:效果比 billboard 更好；比模型性能高；无视角限制；
缺点:视角突变(跳变)；视角切换不自然；画面质量降低；遮挡和深度问题；没有精确投影(手机)

LOD 的过渡技术

①使用 alphatest 剪裁模式
例如 dither fade
②alphablend 混合模式
③无视
总结 LOD
优点:渲染性能高；相比直接剔除物体画面质量更好
缺点:多个层级网格增加开发成本；视觉突变

物体剔除方法

1 🟦背面剔除

根据法线朝向判断(朝向的方向就是正面)

2 🟦距离剔除

通常和 3 🟦结合

3 🟦视椎体剔除

通常会使用简易包围盒判断来简化计算

4 🟦遮挡剔除

遮挡剔除生成场景数据来确定相机可见内容的过程叫→烘焙
unity:半离线遮挡剔除 UE:全离线遮挡剔除
↓
缺点:无法处理动态模型，增加烘焙时间，增加内存占用

5 🟦earlyz 提前深度测试

可能导致 earlyz 失效的情况:透明物体;深度写入关闭;深度测试关闭;修改了深度值的 shader;无序渲染; 剪裁模式

6 🟦隐藏面消除 Hidden Surface Removal HSR

PowerVR TBDR 下的方案

过程:首先记录在 OnChip Tile Buffer，然后用这个三角形数据计算 tile，然后将深度值比较，看谁更接近视口，最后绘制三角形像素
⚠ 此方法尽量不要使用 AlphaTest

7 🟦低分辨率贴图 Low Resolution Z LRZ

基于 Adreno GPU 的 TBDR
BinningPass 执行过程:运行简化 vertex shader,(包含顶点位置计算）→生成低分辨率深度缓存用于标记哪些需要跳过 renderingpass 计算
binningpass 的两种模式:direct mode(不 binningpass)； binningpass 在包含复杂依赖位置信息的时候会进行两次计算
原理:根据 lrz 的分辨率与屏幕对应像素精度来剔除
优点:与物体渲染顺序无关，速度快于 earlyz
⚠ earlyz 和 lrz 目前不能共存，片元写入深度信息，Vulkan 使用辅助命令缓冲，使用模板测试等需要禁用

场景光照和阴影优化

如何解决交互动态光照方案(光照烘焙和实时阴影)

光照烘焙的类型

1 🟦光照贴图

光照贴图坐标:又称第二套 UV 映射
原理:预计算→生成光照贴图→实时渲染(纹理和光照混合)

2 🟦光照探针

对探针点的间接光照进行预计算和存储
预计算→存储光照信息→实时计算光照信息
优点:实时间接光照，性能高
缺点:生成方式是阵列式，工作量大，精度有限(无法平衡)

3 🟦反射探针

预计算→存储信息→实时计算(手机端直接切换，电脑插值)
缺点:只支持静态反射，带宽占用(信息在立方体贴图里加大读取压力)

实时阴影的算法

1 🟦手绘圆片法

2 🟦平面投射法→改进，集中于一张渲染贴图里在屏幕空间计算，然后读取，然后模糊处理实现软阴影

3 🟦阴影映射 shadow mapping

原理:深度信息存储在贴图，相机渲染场景，计算投影坐标，获取像素距离光源距离，比较像素距离光源和阴影贴图这两个距离
缺点:
①阴影锯齿(解决方法，提高贴图分辨率，锯齿优化算法如 CSMCaded Shadow Mapping 通过相机视角划分多个级联区域生成 lod 阴影贴图)
②阴影泄漏(调整 bias 偏移缓解 几何模型正反面朝向正确，模型封闭，避免太薄物体模型)
③性能开销大
④不支持软阴影

场景模型优化

一级资源优化

大模型拆分 根据空间拆分；根据建筑组件拆分；通用物件拆分(分辨率小的贴图可以合并成 1024)

二级资源优化

第一种方法是组件式优化
第二种方法是常规优化(材质数量过多，一个建筑只使用一个，同风格贴图使用同一张，修改UV）贴图合并工具，制作模型时考虑剪影效果

三级资源优化

GPUinstancing 贴图合并，大量模型合并成新模型
打组
碰撞体使用的注意点
初期就考虑；考虑物体本身特点；保证容错率；限制碰撞体数量；关注碰撞体性能开销；技术美术沟通；特殊使用别的方法
VAT 顶点动画贴图:一种物理模拟技术