

张文杰 Unity 客户端面试手册

上海高级客户端与主程方向 2026 年 10 月 28 日

这份手册把你的实际项目拆成可练习的面试回答，并针对目前筛选的 8 个岗位安排复习重点。主线是：用 P7 证明战斗系统设计能力，用 P15 证明性能分析与验证能力，用共用框架、SDK 和构建工具证明完整交付能力。

先练这三个回答

第一，90 秒讲清自己负责过什么；第二，画出 P7 从技能命令到伤害和清理的链路；第三，解释 P15 的 3.659 ms 到 1.911 ms 究竟测了什么。基础题用于支撑这三段经历，遇到追问时能继续往下讲。

复习内容	页码	练习成果
真实经历与岗位重点	2—4	分清主攻岗位和能力缺口
开场与项目深挖	5—15	每个案例能讲 90 秒与 5 分钟
基础八股与常见追问	16—24	先回答原理 再连接项目
现场编码与参考实现	25—28	先说明边界 再写代码和测试
模拟面试与谈薪	29—30	练完整一轮并复盘
七天训练与核对清单	31	补齐证据和最薄弱的题
岗位及技术参考	32—34	查看原文与版本

只有两小时怎么用

先读第 5、6、9、13、16、17、18、30 页；最后闭卷讲一次 P7，手写第 25 页的计数题。把说不出的细节记下来，回查实现，不临时编造数字。

怎样使用示范答案

“经历回答”按简历和个人网站整理，仍请用你熟悉的表达复述。“设计题”是供面试讨论的方案，不能说成已上线经历。题目均为按岗位要求设计的模拟题，不代表公司的真实题库。

真实经历与回答边界

你的优势是独立承担复杂模块并贯穿交付过程。把每项贡献讲清楚，比堆技术名词更有说服力。

可以作为经历讲	应保留的边界
银梭云造 6 个 Unity 客户端项目 4 个已上线 P7 和 P15 与发行合作	不要说 6 个全部上线 不要补写流水 用户量或团队人数
P7 战斗核心独立设计并编写	P7 的 ORCA 使用现有库 不要把基础算法说成独立原创
P15 属性热点基准下降约 47.8%	是指定路径耗时变化 不是手机 FPS 或全战斗提升
MVCS 与 IoC 框架 SDK 工具链 服务端和运营后台维护	框架整合了现有技术及开源库 跨端维护不等于大型后端架构经验
P9 P10 P15 大量业务使用 AI 辅助	准确解释需求拆解 设计 审查和验证 不要说所有代码都手写
Lua 了解语法	没有正式项目使用经历 UGUI 深度 SLG 规模经验也需单独核实

经历时间线

简历列出：大连新视域 2020.12—2021.11；上海字逗 2022.03—2023.05；网易 2023.07—2024.05；银梭云造 2024.07—至今。按这些时间讲累计经验，避免直接说“连续六年”。本科为大连东软信息学院数字媒体技术。

面试前必须核对

BOSS 在线经历与 PDF 的第一段起止时间存在差异，是否包含实习需要你确认。当前公司仍在职，在线结束日期也需统一。核对劳动合同或离职证明后再改，不猜日期。

数据不记得时怎么答

“我能说明当时定位和改动的方法，但设备型号或精确数字我需要回看记录。现在可以先讲测量口径和验证步骤。”比临时给一个漂亮数字可靠。

依据 个人简历及作品集 S01

主攻岗位的准备重点

工资是职位展示的税前区间。额外薪数、固定薪资、试用期及团队要求尚未向招聘者核实。

波克城市 客户端开发工程师

25—50K · 15 薪。优先讲跨项目模块、复杂系统、资源生命周期与发布。JD 接受 XLua、ILRuntime、HybridCLR 等方案中的相关经验，你可以围绕真实的 HybridCLR 工作展开。

模拟追问：棋牌或休闲系统中，活动页、背包和红点怎样解耦？切渠道需要修改哪些层？准备第 13—15、19、24 页。

Garena 客户端程序系统

25—50K · 15 薪。主打系统架构、旧模块优化、性能和开发文档。讲一个模块如何从界面请求进入命令、服务和数据，再通知界面；说明版本演进与回归。

模拟追问：如何重构一个仍在迭代的旧系统？怎样避免一次重构影响全部项目？准备第 9、13、18、21 页。

鹰角 高级客户端开发 系统向

25—35K · 16 薪，明日方舟方向。局外系统、UI、协议、工具链与你相关。FairyGUI 是你现有证据，UGUI 的 Canvas、Layout、事件与复杂列表是需要补强的部分。

模拟追问：多个入口同时打开同一界面怎么办？千条物品列表如何复用？刘海屏、多语言与弹窗跳转怎样验收？准备第 13、19、26 页。

淘米 高级 Unity 客户端 回合制战斗

25—35K · 14 薪。P7 技能、Buff、状态与回合收尾很有价值，但岗位明确看重完整回合制模块以及弱网、重连。先核实对“回合制”的具体要求。

模拟追问：服务端结果怎样驱动动作队列？表现中断与断线如何恢复？P7 有回合流程，不自动等同于网络回合制项目。准备第 6—10、23 页。

讲述顺序

系统岗：框架 → UI 资源 → 业务交付 → 性能。战斗岗：P7 → P15 → 数学与 ECS → 网络设计。不要给每家公司背完全一样的自我介绍。

岗位原文 S02—S05

莉莉丝与备选岗位的准备重点

莉莉丝已看到合适的 Unity 岗位，但这些具体职位的描述没有明确写小游戏团队。面试时先确认产品和团队。

莉莉丝 高级 U3D 客户端开发

20—40K · 16 薪。优先准备玩法模块、移动端性能、CI 工具和独立解决问题的案例。用 P7 加工具链作为开场，再讲 P2、P3 的完整玩法与渠道落地。

向招聘者确认：负责新玩法还是成熟系统？项目技术栈和目标平台是什么？小游戏经历在该团队怎样使用？

莉莉丝 Unity 客户端 欧美 SLG

20—40K · 16 薪。SDK、业务、发布与你直接相关；3D 数学、渲染、Timeline、Animator 是补强重点。P10 群体移动可作为相关案例，不能代替大型 SLG 上线经历。

模拟追问：大量单位的更新和渲染怎样分级？跨平台 SDK 回调怎样进入业务？准备第 11、12、15、22 页。

莉莉丝 客户端储备主程

40—65K · 16 薪，作为冲刺岗位。准备模块边界、技术选型、代码审查、性能预算、迭代和交付。你可以说技术负责与跨端协调；人员管理范围要按真实情况回答。

模拟追问：接手一个性能差且仍在上线的项目，前三周怎么安排？如何决定继续 ECS 还是保留 GameObject？准备第 18、21、29 页。

游之趣 Unity 游戏开发 甲方自研

26—35K · 13 薪。框架、UI、资源、工具方向匹配，但 JD 有熟悉 Lua 的要求。坦诚说明目前只了解语法，询问是否允许以 C# 与 HybridCLR 经验进入岗位。

如果核心业务要求立即维护 Lua 热更，不把“可以学习”包装成现成生产经验。可以补一个练习项目，但要明确它是练习。

暂不作为主攻

Playable 广告岗位要求的 Unity/Cocos、单文件 HTML、小包体等，与普通小游戏研发有区别。要求两年 Lua 或精通 Lua 的岗位，目前不应靠改简历硬贴。

岗位原文 S06—S09

自我介绍与项目选择

把自己的定位说成能交付系统的 Unity 客户端开发者。跨端能力是加分项，核心技术深度仍放在客户端。

90 秒通用开场 经历回答

“我是张文杰，主要做 Unity 和 C# 客户端开发，目前在上海担任客户端开发主程。在现在的公司，我负责过 6 个项目的完整客户端研发，其中 4 个已上线，P7 和 P15 有发行合作。我的工作主要有两块：一块是战斗和玩法，例如独立设计并编写 P7 的 ECS 战斗核心，覆盖技能、Buff、词条、碰撞和伤害；另一块是工程基础，包括跨项目框架、界面和资源生命周期、多平台 SDK、热更新与自动构建。后续也接手了部分服务端和运营后台维护。最近比较有代表性的工作是 P15 属性热点优化，我做了固定负载基准来验证。希望下一份工作继续承担复杂客户端模块，把系统质量和交付做得更扎实。”

系统岗再补一句

“如果方便，我想先用共用框架里的异步开窗和资源释放讲一个具体问题，再展开局外系统和工具链。”这能把讨论引到你把握的范围。

战斗岗再补一句

“我可以从一次施法如何推进，讲到 Buff 贡献撤销、命中间隔以及回合完成条件；然后再讲后续项目怎么处理性能与时序。”

五分钟项目讲述顺序

前 30 秒讲项目玩法与责任；接着 60 秒画模块和数据流；再用 90 秒讲一个难点；用 60 秒说明方案取舍和验证；最后 60 秒讲限制与后续改进。每次只展开一个主问题，让面试官决定追问方向。

被问最难的问题

优先选你能复原过程的事实：P15 属性访问与副作用、异步开窗关闭竞态、P10 群体拥堵和透明遮挡。先说明问题如何被确认，再解释为什么改这一层。不要凭空补一个“凌晨上线事故”。

被问为什么做很多事

“客户端是我的主线。接服务端、SDK 和工具，是为了把接口、版本和交付闭环起来。我希望在新岗位继续承担技术责任，同时让职责和优先级更清晰。”

P7 技能系统怎样讲深

经历回答 命令配置描述动作 执行组件保存施法进度

90 秒回答

“P7 一次技能会包含索敌、前摇、发射、位移和效果。我把它拆成共享配置、施法者自己的执行状态，以及按类型分派的命令处理。配置保存动作、延迟、持续时间和参数；执行状态保存命令游标、时间原点和是否已经执行。更新时到达计划时间才派发动作，后续只判断完成，避免每帧重复生成子弹。技能结束和子弹销毁有不同生命周期，表现也不负责决定伤害。”

为什么执行游标不能放在配置里

多个角色会使用同一技能配置。配置属于静态定义，进度属于某一次施法。把游标放进共享数据会让角色互相覆盖状态；需要施法者或施法实例独立保存。

继续追问：同一角色可否同时施放两个技能？先明确当前玩法限制；若支持并行，按施法实例管理游标和取消范围。

怎样保证发射只发生一次

到达命令时刻后派发一次并设置执行标记；等待期间检查完成条件；推进下一条时重置标记。帧率变化导致跨过多个时点时，要明确是否补执行以及单帧上限。最后一项是扩展设计，需对照实际代码。

找不到目标怎么办

P7 的 FindTarget 在命令执行时同步搜索。没找到时按 Duration 处理等待或完成；等待超时不代表每帧重新搜索。普通索敌系统与本次施法目标也要隔离，避免正在施法时被覆盖。

如何实现打断

先区分技能流程、已创建子弹、持续 Buff、动画和音效。流程终止时清游标与施法状态；其他对象依据规则独立继续或撤销。面试设计中可加入施法实例标识追踪所有者，不能说 P7 已有未核实机制。

配置化是不是就不用写代码

新组合可复用已有命令；新行为仍要新增处理器、参数校验、完成条件和清理逻辑。参数位置和单位模糊会把错误推到运行时，编辑器校验与命名参数应一起考虑。

白板练习

画出 配置 → 执行状态 → 命令派发 → 子弹或效果 → 数值结算 → 表现。逐一标出谁创建、谁销毁、能否跨回合存在。

作品集 S01 的 P7 技能专题

P7 Buff 与词条的追问

经历回答 先定义合并 触发和撤销语义 再选择数据结构

Buff 到期为什么不能恢复旧属性

假设基础攻击 100，A 加固定 10，B 加 30%，结果是 140。A 到期只移除自己的固定贡献，结果应为 130。如果恢复 A 生效前的 100，会抹掉 B。P7 将普通属性拆为基础、固定和百分比部分，并记录已施加贡献。

多个施加者的同名 Buff 如何处理

P7 当前查找已有 Buff 的键是目标与 BuffId，不含 Caster，因此不能说每个施加者天然独立。若玩法要求独立存在，可以扩展实例身份或合并键；还要同时更新刷新、驱散、来源死亡与上限规则。

叠层和周期触发是一回事吗

不是同一个状态。实例层数用于叠加规则；周期触发还需要计时与已应用贡献。不可叠层可刷新时间，达到层数上限仍可能刷新，但不能重复增加已经达到上限的属性。

所有效果都能减回去吗

普通加成可以按贡献撤销，控制、覆盖、护盾消耗和永久效果必须各自定义。P7 有 RestoreOnExpire 分支。攻击间隔还采用与攻击力不同的计算方式，百分比增速应缩短间隔，不能套同一公式。

词条与 Buff 的边界是什么

Buff 侧重实例生命周期和持续效果；词条更接近角色附带的规则：匹配事件、判断条件、派发效果，也可能修改技能参数。P7 按效果类型选择策略，避免每个词条 ID 都堆一个分支。

冷却从什么时候开始计算

必须说清尝试触发还是成功生效。P7 当前最小间隔时间戳在条件评估前更新，条件失败也可能占用间隔。若产品希望成功后才冷却，应调整时机并测试概率、目标失效和多事件同帧等情形。

必须能手算的例子

基础 100，加固定 10，再加 30%；撤销固定后为 130，再撤销百分比后为 100。随后把基础值改为 120，验证仍是按当前基础与各项贡献重算，不恢复历史快照。

作品集 S01 的 P7 Buff 与词条专题

P7 空间查询 碰撞与结束流程

经历回答 候选检索 几何相交和玩法命中是不同阶段

空间网格怎样降低索敌成本

用 $\text{floor}(\text{position} / \text{cellSize})$ 将对象映射到格子；查询搜索范围覆盖的格子，去重后检查阵营、存活、距离及策略。它减少远处无关对象的检查，但需要维护索引，密集格子和超大范围仍会退化。

不要回答“哈希表所以 $O(1)$ ”。复杂度应包含访问格数、候选数与精确过滤。

为什么负坐标不能直接强转整数

整数截断朝零，例如 -0.2 会变 0；网格需要向下取整得到 -1。对象占多格时还要去重，边界与尺寸约定必须统一。

圆和旋转矩形如何判定相交

把圆心转到矩形局部坐标，逐轴限制到矩形半宽和半高内，得到最近点，再比较距离平方与半径平方。两个矩形可用分离轴检测；找到一条不重叠投影即可排除。

持续伤害为什么不能只用 HasHit

同一子弹或区域可能影响多个目标。P7 按目标记录 LastHitTime，再根据间隔判断是否允许再次生效。几何碰撞发生后仍需要阵营、穿透次数和重复命中规则，不能检测到相交就直接扣血。

高速子弹为什么还会穿过去

离散检测只检查采样时刻，可能一帧越过目标。可讨论扫掠检测、子步进或从前位置到新位置的形状扫描，取决于速度和成本。P7 的现有实现不能直接宣称具备通用连续碰撞保证。

敌人清光后为什么不立刻结算

还可能有未生成波次和待生成请求。P7 正常清怪路径检查波次完成、已生成数量、存活数量和待生成请求，再提交一次结果；掉落、伤害收尾、清理与界面切换继续按阶段推进。

测试时最容易漏

大单位跨格、负坐标、圆角擦边、目标销毁、子弹穿透多人、同帧补生成、场上为零但仍有待生成请求。失败、复活和特殊关卡需要独立验证，不能套正常清怪条件。

作品集 S01 的 P7 索敌 碰撞和关卡专题

P15 性能案例的完整回答

经历回答 用测量口径支撑优化结论

90 秒回答

“P15 的属性施加和撤销很频繁。原路径为更新少数字段反复取回和写回整个 StatsComponent。我集中属性到字段的映射，借助公开 Chunk 访问接口定位字段，在一次同步操作内完成读写，并保留生命、移速、控制状态等副作用。直接访问前完成相关依赖，不跨结构变化保存地址。在同机固定负载中，1024 个实体各做一组 Add/Remove，中位耗时从 3.659 ms 降到 1.911 ms，下降约 47.8%。这是这条路径的历史基准，不是整场战斗或手机帧率。”

基准条件	记录
运行环境	Windows Unity Editor batchmode 独立 ECS World
硬件和版本	i5 12600KF Unity 2022.3.14f1 Entities 1.0.16
测量方法	同机同入口 Burst 开启且预热 安全检查开启
负载和统计	1024 实体 每实体一组 Add/Remove 7 轮 × 8 次 取中位数 不含准备时间

为什么不用关安全检查证明更快

比较必须保持相同条件。安全检查可能有成本，但关闭后得到的结果不能与开启版本直接混用；先比较同条件改动，再独立测发布环境。

性能更好但结果不一样怎么办

优化首先保持语义：不同属性类型、正负贡献、叠层、撤销、最大生命变化和控制状态都要做等价检查。不要只比较最终攻击力而漏掉联动副作用。

这能提高多少手机帧率

“目前这组数据不能推导手机 FPS。我会在目标设备、固定场景中测主线程、Job、渲染、内存和帧时间分布，再看该热点占比与同步等待是否变化。”

继续改进怎么答

优先回到测量：如果属性已不是主瓶颈，就检查剩余的重复扫描、同步点或表现提交。固定种子回归能减少输入差异，但不自动证明跨平台确定性。

作品集 S01 的 P15 属性专题 ECS 原理 S11—S13

P15 时序 棋盘与断点恢复

经历回答 顺序与生命周期需要成为可检查的规则

为什么先快照所有请求队列

效果执行过程中可能增删组件，导致原 Buffer 引用失效。P15 先复制并清空全部待处理队列，再执行效果。缓冲区复用降低临时数据成本，同时要定义执行期间新产生请求何时处理。

所有伤害都在同一帧结算吗

不能这样说。P15 按系统依赖运行，Buff 请求转交专用系统；技能伤害请求在当前约定下由下一帧伤害系统消费。讲清请求产生、消费和表现时点，比笼统称“实时结算”准确。

阵旗经过同一格如何防止重复出兵

FlagPass 携带时间、稳定顺序和旋转周期。解析器排序后按旗、格子、周期去重，累计鼓舞，达到阈值产生结果，再通过战斗适配出兵。动画只展示结果，不各自重复修改核心状态。

一次经过跨过多个鼓舞阈值怎么办

处理获得的总量和阈值，计算完整触发次数并保留余量，不能只用一个 if 发一次。还需要明确最大次数、资源上限和额外奖励规则；第 25 页提供独立练习实现。

存档是不是恢复到战斗同一帧

现有主线断点恢复到该回合的准备阶段，不是任意帧精确续战。若要帧级恢复，还需覆盖随机状态、命令游标、子弹、计时、未处理队列和实体身份，并验证加载后结果。

怎样证明阶段切换正确

按 Preparation、Combat、RewardDraft、Won、Failed 画状态图，列合法操作、前置条件和失败原因。测试连点开始、重复结算、奖励返回、失败重试，以及跨会话残留事件。

AI 参与的真实表述

“P15 的业务、战斗和性能改动中有 AI 辅助。我负责需求拆解、方案约束、集成、审查和验证。这里我可以具体解释请求时序、指针生命周期和基准条件。对于代码细节，我以实际实现和测试结果为准。”

作品集 S01 的 P15 技能请求 棋盘与阶段专题

P10 流场与 ORCA 的分工

经历回答 先选择方向 再选择局部可行速度

90 秒回答

“P10 的问题是后排单位一直挤向队友形成的人墙。仅做局部避障并不能规划从人群外绕行。我们把单位密度投到网格并平滑，结合基础通行代价与硬障碍，从目标格传播积分代价，再生成方向场。场按阵营和体型分组，四张场轮换更新。方向场给期望速度，ORCA 根据邻居与边界约束求局部速度，最后还有受阻与残余重叠处理。”

为什么不每个单位都跑 A 星

共享目标或区域的许多单位可复用方向场；独立路径会重复搜索。代价是重建延迟、分组近似和个体目标表达能力不足。对于少量单位、强个体目标或复杂动态障碍，独立寻路仍可能更合适。

流场怎样从目标产生方向

先确定可通行格与代价，再由目标传播累计代价，沿邻接中代价下降方向生成流向。P10 使用多源松弛传播；网站交互演示简化为单目标 Dijkstra，二者不能当作完全同一实现。

ORCA 为什么不等于全局寻路

它根据相对位置和速度排除可能碰撞的速度区域，通过半平面约束选择接近期望速度的结果。它不负责整张地图路线，在完全包围或窄口下可能几乎无法移动。

怎样避免更新顺序影响结果

使用相同一份速度快照与冻结意图，先求解后统一提交。再将受阻统计和持续一段时间的脱困方向独立处理，避免每帧左右反复切换。

你能保证严格不碰撞吗

不能。P10 的低速脱困允许受控的临时约束违反，并且有残余重叠分离。算法的理论条件不能直接覆盖这些工程兜底。P7 则是接入 Nebukam 库，应分别说明。

面试官要看什么

能区分目标选择、全局方向、局部速度和最终位移；能解释邻居数、时间视野、刷新频率和障碍处理的取舍；能说明失败场景。没有实测数据就不报单位数上限。

作品集 S01 的 P10 流场和 ORCA 专题 原理 S22

P10 序列帧渲染与图形基础

经历回答 合批必须服从正确遮挡

90 秒回答

“P10 用共享网格和材质承载多份序列帧实例，实例参数保存位姿、UV、纹理层和颜色。收集显示项后先遵守绘制顺序，再在同资源和类别间尝试合批。达到实例上限、穿插 HUD 或投影范围重叠时拆批，最后调用 `RenderMeshInstanced`。优化目标是减少提交工作，同时保留透明单位、武器和 HUD 的正确遮挡。”

为什么同材质也不一定合成一个批次

透明对象需要正确的混合顺序。实例内部排序不能直接当作每个透明精灵的可靠前后顺序；投影重叠和穿插对象可能要求拆批。图像正确性要先通过，再比较提交开销。

Draw Call 少了为什么仍然卡

瓶颈可能在填充率、透明 Overdraw、纹理带宽、复杂片元或 CPU 模拟。先用 Frame Debugger 看提交，再结合 GPU 与 CPU 时间分析。减少提交次数不意味着 GPU 总工作必然下降。

GPU Instancing 与 SRP Batch 有何区别

Instancing 让兼容的相同网格和材质用多份实例数据绘制；SRP Batch 主要降低兼容 Shader 的 CPU 状态设置开销，并不简单等于把所有物体合成一次绘制。具体兼容、优先级和提交路径要看渲染管线与版本。

点积和叉积怎么在游戏中用

归一化方向的点积可判断夹角和朝向；点积与阈值比较可判视野范围。叉积给垂直方向和旋转侧向，二维可看其有符号标量。求角前注意零向量，避免无意义归一化。

世界与局部坐标有什么区别

局部坐标相对于父变换；世界坐标通过层级变换得到。位置变换包含平移，方向变换不应当作位置处理；缩放尤其是非均匀缩放会影响法线与距离判断。

证据边界

该案例没有固定倍数的目标设备收益记录。可以画实例参数与拆批过程，不能补写“从某帧率提升到某帧率”。面试展示使用可公开内容，勿直接带公司私有资产或源码。

作品集 S01 的 P10 渲染专题 官方说明 S21

共用框架与异步界面生命周期

经历回答 依赖归属和资源归属必须清楚

90 秒回答

“框架通过根上下文装配公共服务，模块在自己的上下文组织业务。View 显示、Mediator 接交互、Command 承接动作、Service 提供能力，数据或事件再驱动界面。资源方面用 Scope 记录资产、实例、场景和子作用域；FairyGUI 面板区分准备、显示、隐藏、缓存和销毁。这样关闭界面时，既能终止后续流程，也能找到需要释放的资源。”

为什么不用全局管理器互相调用

全局入口最初方便，后期容易形成隐式依赖与难以清理的状态。上下文把依赖和生命周期显式组织起来，但 IoC 本身也有装配与调试成本，简单局部逻辑不必过度抽象。

关闭时加载还没结束怎么办

把关闭意图传播给业务流程，在依赖准备及创建内容前检查。任务完成后若该次打开已经失效，应释放它持有的资源，不让旧结果重新显示。取消等待不等于底层操作或已加载资源自动消失。

关掉之后立即重新打开会怎样

需要区分不同打开请求的身份；旧结果不能覆盖新请求。资源管理层可以合并共享加载，但每个使用者仍要有明确的引用或租约。第 26 页展示一种练习方案，不等同于当前框架源码。

隐藏和销毁有什么不同

隐藏仍可能保留内容、事件订阅或资源；缓存减少重复加载但保留内存；销毁要解除订阅和释放自己持有的资源。缓存有数量、容量或低内存回收策略才不会无限增长。

Addressables 的 Release 为什么不立刻降内存

引用计数减少只是生命周期的一步。资产仍可能被共享 Bundle 或其他引用保留；系统内存指标也不必即时回落。要结合资源依赖和内存快照检查，不能每次关窗就强制 UnloadUnusedAssets。

现场画图

画出一个模块 Scope 拥有面板，面板持有资源租约，多个面板共享底层包。标出关闭、失败、重开和模块退出四条路径，逐条检查谁释放且只释放一次。

作品集 S01 的框架专题 Addressables S14—S15

HybridCLR 与构建发布

经历回答加设计补强 重点是完整产物与版本兼容关系

经历回答

“我维护 BuildTool，将 FairyGUI 发布、Luban 导表、Addressables 和 HybridCLR 构建接到 Editor 工具，并用 TeamCity 做自动构建。包体、资源和热更代码分开处理，区分平台及 A/B 版本。以前也接触过 Jenkins。重点是能追溯构建来源和适用范围，避免不同包体拿到错误产物。”

HybridCLR 大致解决什么问题

在 IL2CPP 的 AOT 基础上提供解释执行热更新代码等能力。它不是让所有 C# 都变成 JIT；原生代码、引擎和平台 SDK 等边界仍受到包体约束。面试按实际使用版本说明，不把最新文档功能视作项目现状。

AOT 泛型为什么可能缺实现

热更代码可能使用 AOT 构建时未生成的泛型实例。泛型共享和补充元数据是相关机制。补充元数据需要匹配对应包体的 AOT 程序集，并在相关使用前加载；代码裁剪导致缺失的类型或方法还需单独处理。

只传一个新 DLL 到 CDN 就够了吗

不够。需要校验代码、资源目录、配置、AOT 元数据和基础包版本的兼容关系。原生插件变更可能必须发新包。校验完整性不等于验证真实性，签名与可信发布渠道解决的是另一类问题。

发布半途失败怎么办 设计题

建议为每次发布生成不可变产物目录和清单，先上传全部文件并验证，再切换版本入口；客户端先校验下载完整性再激活。保留上一稳定版本，兼容性判断与回滚策略纳入启动流程。以上是发布设计，不声称现网已完整实现。

CI 应该在哪些点失败

缺配置、错误渠道、资源引用缺失、脚本编译失败、产物不全、版本不兼容、冒烟启动失败等都应阻止发布。并发构建应隔离输出路径与版本状态，避免 A/B 和多平台相互覆盖。

补准备一个真实例子

从你的记录选一次构建或热更问题：哪一步失败、怎么看日志、如何复现、改了哪个校验。没有确认前，不编造构建时长或事故规模。

作品集 S01 的构建专题 官方说明 S16—S17

小游戏 SDK 与跨端维护

经历回答 客户端交付能力可以用 P2 P3 和支付链路证明

小游戏项目最适合怎么讲

P2 可以讲翻格探索、路径判断、楼层与自动战斗，连到成长、任务和引导；P3 可以讲养成收集与钓鱼、2048 等小游戏之间的状态隔离和资源进出。重点是玩法闭环、边界清理与多平台交付，不只是功能列表。

平台差异怎样隔离

业务经过 PlatformAPI，由各 Provider 负责初始化、回调、登录、广告、支付与分享。统一入口不代表能力完全一致；不可用能力、取消、失败和超时必须有可区分结果。

重复广告或支付回调会不会发两次奖励

客户端可做重复点击控制和回调关联，但最终权益发放不能只依赖本地布尔值。P7 有服务端验单、补单和权益同步经历。面试要具体讲你负责的接口与流程；事务设计可按下段方案讨论。

怎么设计服务端幂等 设计题

用可信订单或业务操作 ID 建唯一约束，在事务内检查状态、更新订单并写权益或发放记录。重复请求返回已有结果。第三方调用无法直接被本地事务包住时，用可靠状态、重试和对账收敛，不能保证网络只传一次。

接手服务端和后台做过什么

按实际范围讲 P9、P10、P15 的成长、商店、活动、广告奖励、配置同步、部署和问题排查。公开资料涉及 Go 服务端与 MySQL，以及 Vue 3、TypeScript、Node/Koa 后台链路；面试前确认各技术属于哪个项目和哪个层，避免混成同一服务。

小游戏恢复前台要注意什么

这是补强设计：回到前台重新核对登录、网络、广告与支付未完成操作；使用业务请求身份避免过期回调写入新会话；计时要区分真实时间与暂停时间。不能声称已经处理过所有渠道的每一种异常。

面试展示顺序

先跑一个 P2 或 P3 的完整循环，再展示切场景、退出和重新进入，最后讲平台适配与构建。被问具体首包、加载时长、帧率时，以实际包和记录为准。

作品集 S01 的项目及平台专题

基础八股 CSharp 类型与集合

回答顺序 一句结论 一个例子 一个限制条件

值类型与引用类型的区别

值类型变量包含其值，赋值通常复制值；引用类型变量保存对象引用，赋值复制引用。struct 内含引用字段时只复制那个引用，并不会深拷贝对象。存放在栈还是堆取决于使用位置，不能说 struct 永远在栈上。

装箱与拆箱是什么

把值类型转成 object 或其实现的接口可发生装箱，产生包含值副本的对象；拆箱取出正确的值类型。例如装箱的 int 不能直接拆成 long，应先拆为 int 再做数值转换。泛型可避免许多不必要装箱，但不保证所有路径零分配。

class struct readonly 怎样选

小而稳定的值语义数据适合 struct；需要共享身份或多态对象模型时选 class。大 struct 的重复复制可能更贵。readonly 有助于表达不可变意图，但引用字段指向的对象不因此自动不可变。

List 与 Dictionary 的常见复杂度

List 尾部添加摊还 $O(1)$ ，扩容时 $O(n)$ ，中间插删通常 $O(n)$ 。Dictionary 常规查找平均 $O(1)$ ，碰撞、扩容和哈希实现影响实际表现。键的 Equals 与 GetHashCode 应一致，作为键期间不应修改影响相等性的字段。

委托 事件与闭包有什么坑

委托可引用目标对象；事件限制外部直接触发与赋值。长期发布者持有订阅者会延长其生命周期，所以解绑归属于窗口或模块退出。闭包捕获变量及对象可能产生分配或保留引用；不能每次用新 lambda 取消旧订阅。

foreach 与 LINQ 一定有 GC 吗

不能一概而论。具体集合、枚举器类型、接口转换、闭包和终端操作决定分配。List 的直接枚举与转成接口后的枚举可能不同。热路径以 Profiler 和实际编译结果验证，避免仅凭语法下结论。

接口 抽象类和组合怎么取舍

接口表达能力契约，抽象类可共享状态与实现；组合用独立对象提供行为，降低继承链牵连。PlatformAPI 的多渠道适配适合讲接口边界，Buff 的效果组合适合讲组合而非大量子类。

参考 S18 S20 练习答案为通用原理

基础八股 CSharp 异步与内存语义

先把线程 调度和生命周期分开 再谈具体语法

async 和 await 会创建线程吗

不会自动创建线程。async 方法通常先同步执行，遇到尚未完成的 await 才把后续逻辑登记为继续执行部分。Task 表达异步完成，不代表线程。CPU 密集任务是否放线程池，要由任务性质和平台约束决定。

协程与异步 Task 有什么区别

Unity 协程按引擎调度在主线程分段执行，本身不是后台线程；Task 可表达 I/O 或异步组合，继续执行位置受上下文和 await 对象影响。两者都需要处理取消、对象销毁和异常，不能在后台直接随意调用 Unity API。

为什么少用 async void 与 Result

除事件处理等必要入口外，优先返回 Task，方便等待和观察异常。Result、Wait 会阻塞调用线程，在依赖该线程继续执行的场景可能死锁。ConfigureAwait(false) 不是切回 Unity 主线程的方法。

CancellationToken 会自动停止工作吗

取消是协作机制。被调用方必须检查并响应，底层 I/O 或资源加载不一定立即停止。已获得资源仍需释放；过期结果仍需隔离。关闭窗口只取消等待，可能留下加载完成后的无主资源。

IDisposable 与 GC 有什么区别

GC 处理不可达托管对象，Dispose 约定确定性释放资源或撤销注册。using 在退出作用域时调用 Dispose，不能代替 Unity 原生对象的正确销毁 API，也不应依赖终结器在主线程清理引擎对象。

lock 能解决所有并发问题吗

lock 对同一锁保护的临界区提供互斥。多个锁顺序不一致仍可能死锁，锁内不应做不可控长操作。线程安全容器只保证其规定的操作，多个操作组成的业务流程仍可能竞态。

异常处理怎样避免吞错

只能在补救、转换语义或记录上下文的边界捕获；finally 用于清理；重新抛出保留原异常链。失败要让上层可见，重试需分类并有上限。支付超时不等于失败到账，重试还需幂等。

参考 S18—S20

基础八股 Unity 生命周期与性能

围绕输入 更新 物理 渲染和测量回答

Awake OnEnable Start 怎样理解

Awake 用于对象初始化，OnEnable 会在启用时再次触发，Start 在首次更新前执行一次。不同对象之间不依赖未明确定义的顺序，运行时创建和非激活对象还会影响时机。依赖装配应明确，不靠某个脚本碰巧先跑。

Update FixedUpdate LateUpdate 区别

Update 按渲染帧更新，FixedUpdate 按固定步长推进物理相关逻辑，某渲染帧可零次或多次；LateUpdate 常用于跟随等后处理。输入采样、物理施力与画面插值要按系统职责安排。

deltaTime 可以解决所有帧率差异吗

它可帮助速度按时间积分，但不能自动解决离散检测漏碰撞、非确定浮点运算或不正确的事件补偿。需要固定步长、子步进还是扫掠检测，由模拟和成本要求决定。

对象池需要重置哪些状态

计时器、位置、目标、Buff、事件订阅、协程、异步请求、碰撞和表现。复用对象时引入 generation 或会话身份，可阻止上一次生命周期的回调写入新用途。池也应设容量与回收策略。

掉帧先优化哪里

先在目标设备复现，区分 CPU 主线程、工作线程、GPU 或同步等待；看 Timeline、热点采样及 GC，再形成最小可比较场景。只看到某个函数耗时高，不代表它是根因，可能是在等待其他工作。

为什么编辑器数据不能当发布结果

Editor 本身有额外逻辑，脚本后端、平台、分辨率、构建配置和诊断开销可能不同。Deep Profile 会改变开销，用于定位后还要回到正常条件验证；优化前后输入和采样方式保持一致。

怎样报告性能结果

报告设备、版本、场景、负载、预热、样本量、统计口径，以及正确性是否一致。用毫秒与帧时间分布描述，说明平均数和峰值；没有目标机实测就不换算设备帧率。连接你的 P15 案例回答。

参考 S23—S25

基础八股 GC 资源和 UI

避免把托管堆 引擎原生资源和显存混成一个指标

Unity 的 GC 是分代的吗

按你项目的 Unity 2022.3 文档，采用 Boehm GC，属于非分代体系。增量 GC 将部分工作分摊到多帧，目的是降低长暂停，不代表没有分配和收集成本。不要直接照搬桌面 .NET 的 Gen0 Gen1 Gen2 回答。

GC Alloc 为零就没有内存问题吗

不是。可能仍有存活对象持续增加、Native 容器未释放、Texture 或 RenderTexture 占用、Bundle 依赖和显存增长。比较同场景进出前后的快照与引用链，检查预期应该释放的所有者。

Addressables 加载与实例化怎样配对

明确一次成功获取对应谁的释放责任。LoadAssetAsync 后自己 Instantiate 与 Addressables.InstantiateAsync 的所有权不同，后者通常配对 ReleaseInstance。缓存、失败句柄和共享引用也要按 API 约定释放。

UGUI 重建与合批怎样理解

布局和图形数据变化会使相关 UI 变脏并在更新流程中处理；大量 Layout 嵌套、频繁改动或大 Canvas 都可能增加开销。可按更新频率隔离动态区域，但增加 Canvas 也可能影响批次，需实测。

一万条列表怎么做

只创建视口覆盖及少量缓冲项，滚动时复用并重新绑定。固定高度可按偏移算索引；变高列表需要高度缓存和索引映射。异步图片回包要核对当前条目身份，避免复用后串图。

UI 事件与适配怎么答

解释 EventSystem、输入模块与 Raycaster 的职责；关闭不需要的射线目标，检查遮挡与点击穿透。适配考虑锚点、布局、CanvasScaler、Safe Area、多语言长度和字体缺字，最后用具体设备和语言验证。

FairyGUI 经历如何连接 UGUI

“我的主要实战是 FairyGUI，资源所有权、异步竞态、列表复用和生命周期问题有共性；UGUI 的 Canvas 与布局重建机制我会单独说明。”不要把一个框架的实现细节硬套给另一个。

参考 S14—S15 S24 S26

基础八股 ECS Job 与 Burst

记住 数据布局 调度和机器码优化分别解决不同问题

Entity Component System 分别是什么

Entity 是实体身份；Component 保存数据；System 按匹配数据执行逻辑。相同组件类型组合形成 Archetype，实体数据在 Chunk 内按组件类型组织，有利于批量访问。收益取决于数据布局与实际访问模式。

哪些操作属于结构变化

创建销毁实体、增删组件、改变共享组件值等可能改变存储布局。结构变化可能造成同步和搬移，使已有直接引用失效。普通组件字段赋值与改变组件集合不是一回事。

ECB 是让结构变化并行执行吗

ECB 允许在 Job 中记录命令，稍后统一回放；并不意味着实际所有结构变化都同时在多个线程完成。选择合适回放时点，建立生产者依赖，并考虑命令排序和目标失效。

Enableable Component 有什么作用

可切换匹配状态而不必增删组件和改变 Archetype，适合高频开关状态。它仍有查询和调度方面的考虑，不应把所有布尔字段都改成 Enableable。与真实增删组件的语义差异要说清。

Job 为什么仍然需要依赖管理

同时写同一数据或读写冲突会有竞态。声明读写关系、串联依赖，在主线程访问前完成相关任务。过早 Complete 会损失并行度；忽略依赖则会牺牲正确性。Native 容器的分配器和释放时机也要匹配。

Burst 等于多线程吗

不是。Burst 对支持的 C# 子集生成优化机器码，Job 调度决定如何并行。Burst 热路径通常避开托管对象，使用支持的数据类型；具体受包版本限制。不要写一个特性就宣称所有循环自动向量化或并行。

怎样回答 ECS 的缺点

结构变化、碎片化 Archetype、随机访问、跨 GameObject 表现桥接、开发与调试成本都可能抵消收益。你可用 P15 讲减少组件复制，也要说明指针不能跨结构变化保留；低频 UI 没必要为形式统一迁入 ECS。

参考 S11—S13

基础八股 架构 设计模式与工程质量

模式服务于依赖和变化 不需要把所有名词都装进项目

MVC MVVM MVCS 有什么区别

MVC 划分数据、显示和控制；MVVM 通过 ViewModel 暴露显示状态和交互，常配合绑定；MVCS 常强调 Command、Service 等协作角色，具体框架定义不同。按你的真实 View、Mediator、Command、Service 职责解释。

依赖注入与服务定位器区别

注入让依赖通过构造或装配显式进入；服务定位器由对象运行时主动获取依赖，使用方便但容易隐藏依赖。IoC 不等于到处从全局容器取对象，模块生命周期和释放规则仍要设计。

策略 命令 状态机分别适合什么

策略选择一组可替换算法，例如索敌；命令封装动作与参数，例如技能步骤；状态机约束当前阶段允许的行为，例如准备、战斗和选奖。共同点是隔离变化，但不能用模式名替代数据流解释。

怎样避免事件总线变成黑盒

明确事件是事实通知还是请求命令，写清生产者和消费者；限定作用域、身份和顺序，订阅生命周期有归属。关键流程需要日志关联或显式调用链，不能靠一串隐式事件完成全部业务。

怎样重构仍在运营的模块

先找公开边界和回归场景，在现有行为旁建立可验证的新实现；按小范围逐步替换，记录兼容与回滚点。先改最影响维护或性能的部分，避免一次连同协议、资源与框架全改。

主程怎样做代码审查

先看需求和接口语义，再看生命周期、异常路径、并发时序、分配与性能、可测试性和发布兼容。优先指出会影响正确性和交付的问题，对风格用自动规则统一，避免靠个人喜好阻塞。

你怎么用 AI 保证质量

“我用 AI 辅助分析和实现，但先定义输入输出、边界和验收，再审查改动并运行验证。”对于真实 P15 可具体讲依赖同步与属性副作用；没有记录的成功率、提效倍数不报数字。遇到陌生实现先查文档和做小实验。

基础八股 图形学与动画

系统岗位也可能追问 重点掌握可解释的基础和工程用途

渲染流水线怎样简述

从顶点输入和变换开始，经过图元装配、裁剪与光栅化，产生片元并进行着色，再与深度、模板和混合等测试及处理结合生成输出。具体优化可能改变执行细节，别把它背成所有硬件固定不变的顺序。

透明为什么经常不写深度

普通半透明混合通常依赖后向前顺序；如果不恰当地写深度，后面的透明层可能被提前挡掉。Alpha Test 和 Alpha Blend 不是同一效果；透明排序、深度写入和材质队列要一起设计。

四元数比欧拉角适合在哪里

四元数便于组合和插值旋转，避免使用欧拉角表示时的万向节锁问题，但乘法不交换，旋转顺序仍重要。插值结果和归一化、最短路径有关；不要直接逐分量 Lerp 后当作已处理所有情况。

点 法线与方向怎样变换

点包含平移；方向不包含平移。非均匀缩放下，法线需要与切平面保持垂直，通常使用逆转置相关变换；直接用位置矩阵乘法线可能错误。

Animator 与 Timeline 怎么分工

Animator 通常表达角色动画状态与转移，Timeline 适合按时间组织演出片段。游戏规则应与纯表现分层；表现播放完毕事件不能未经约束就成为权威伤害或奖励依据。以你实际扩展 Timeline 的范围回答。

Root Motion 与代码移动怎么选

Root Motion 从动画提取位移，适合强调动作匹配；代码控制便于规则和网络约束。混合使用时要避免双重位移，明确位置所有者、碰撞修正和被打断时的处理。

纹理和网格内存怎么估算

先看尺寸、格式、Mip、平台压缩、是否可读及拷贝，再看引用与实际驻留。纹理文件大小不等于运行时内存；网格还包括顶点属性和索引。排查应看目标平台导入设置与快照。

参考 S21 S27—S28

基础八股 网络与服务器设计

以下是知识与设计练习 不能说成已做过大型联网战斗

TCP 与 UDP 的区别

TCP 提供可靠、有序的字节流，不保留应用消息边界，因此需要长度或分隔等应用层协议。UDP 以数据报为单位，不替应用保证可靠和顺序；业务可按需要设计序列号、重传或丢弃过期数据。不能简单说 TCP 慢 UDP 快。

为什么会出现粘包和半包

应用一次写入不等于对端一次读取。对 TCP 应维护接收缓冲，按长度解析完整消息，限制最大包长并保留未完整部分；不要假设一次 Read 就是一条协议。

HTTP WebSocket 怎么选

HTTP 适合请求响应与标准基础设施；WebSocket 在握手后提供持续双向消息通道。选型看交互频率、服务器架构、平台和运维。采用 WebSocket 不代表业务自动可靠、幂等或可恢复。

超时后直接重试可以吗

超时说明结果未知，服务端可能已处理成功。写操作需要业务幂等标识、状态查询和明确重试策略。建议退避与抖动，限制次数；登录过期、参数错误和暂时网络失败应分类处理。

回合制弱网重连怎么设计

服务端确认动作是否合法，并产生带回合与序列的结果。客户端按顺序播放，去重旧结果。重连时先拿一致性快照与最后已确认序列，再补后续结果；动画播放进度与规则状态分离，可跳过或加速旧表现。

帧同步与状态同步怎么比较

帧同步主要传播输入并依赖各端一致模拟；状态同步主要传播权威状态，可结合预测和校正。确定性、带宽、延迟、作弊风险、回滚成本和平台差异决定选择。固定步长和相同随机种子只是必要因素的一部分。

如何设计一次领取奖励

请求带业务 ID；服务端校验领取资格，唯一记录与权益更新在一致性边界内完成；重复请求返回已有结果。客户端显示“处理中”不等于已到账。跨系统发放可用可靠事件、重试与对账，不声称网络恰好一次。

网络基础参考 S29—S30

基础八股 算法 数据结构与复杂度

先明确输入规模与正确性 再选结构 不必把所有问题变成复杂算法

问题	常用结构或算法	要说明的边界
最近目标 Top K	线性扫描 或大小 K 的堆	单次最近 $O(n)$ 频繁查询先空间粗筛
定时 Buff 到期	最小堆 或时间轮	大量同刻到期 取消 版本失效
资源缓存淘汰	字典加双向链表 LRU	容量 在用资源不能直接释放
无权最短路径	BFS	队列 去重 不可达
非负权最短路	Dijkstra	过期堆项 负权不适用
目标导向搜索	A 星	启发式与图搜索实现决定最优性
配置依赖顺序	拓扑排序	检测环 不依赖哈希遍历顺序

二分查找最常错什么

区间定义不一致、重复值、空输入和边界更新。先说明要任意命中还是第一个大于等于目标。使用半开区间 $[left, right)$ ，条件成立缩右，否则 $left = mid + 1$ ，最后返回插入位置。

什么时候排序比哈希更合适

需要稳定有序遍历、范围查询或批量处理时，排序后的结构可能更易维护。哈希快查并不提供顺序和范围能力，还要考虑内存与键一致性；选择由查询模式决定。

递归有什么问题

深度受调用栈约束，循环引用会无限递归。树深可能很大时用显式栈，图遍历还需访问状态。业务配置的依赖图要区分“当前路径中”与“已经处理完”，才能准确发现环。

复杂度怎么说得像做过项目

空间网格说 $O(\text{访问格数} + \text{候选数})$ ，而非所有查询 $O(1)$ ；LRU 说字典平均 $O(1)$ 加链表 $O(1)$ ，并交代容量和在用资源；优先队列说插入、弹出 $O(\log n)$ ，保留失效记录需清理。

现场习惯

写之前先举一个普通例子和一个极端例子；编码过程中说清不变量；最后测试空输入、边界、重复、溢出和失效对象。不会最优解时，先给正确基线再讨论优化。

编码练习 计数与属性贡献

题目一 一次获得大量鼓舞时 返回触发次数并保留余量

约定：amount 非负，threshold 正数，原 remainder 位于 $[0, \text{threshold})$ 。使用 long，溢出时显式报错，不悄悄绕回。

```
public static long Consume(ref long remainder,
    long amount, long threshold)
{
    if (threshold <= 0 || amount < 0 ||
        remainder < 0 || remainder >= threshold)
        throw new ArgumentOutOfRangeException();
    long total = checked(remainder + amount);
    long count = total / threshold;
    remainder = total % threshold;
    return count;
}
```

口头解释与测试

时间和额外空间 $O(1)$ 。余量 9、增加 25、阈值 10，结果 3 次且余 4；增加 0 不触发；恰好一个阈值触发一次。还要测试非法阈值、负数和 long 溢出。该题只算次数，发奖事务另行设计。

题目二 按来源管理属性贡献

约定只讨论普通属性： $\text{value} = \text{base} \times (1 + \text{百分比总和}) + \text{固定总和}$ 。同一 source 再次写入替换该来源贡献。移除 A 不影响 B；这不是所有 Buff 类型的通用实现。

```
// modifiers: Dictionary<int, Modifier>
public double Evaluate(double baseValue)
{
    double flat = 0, percent = 0;
    foreach (var m in modifiers.Values)
    {
        flat += m.Flat;
        percent += m.Percent;
    }
    return baseValue * (1 + percent) + flat;
}
```

继续追问

重算 $O(n)$ ，更新字典平均 $O(1)$ 。要 $O(1)$ 求值可维护聚合量，但替换需先减旧贡献，且留意浮点误差与回归。生产 ECS 需换成适用的存储方式，不把托管 Dictionary 放进 Burst Job。

验证案例

基础 100，A 为固定 10，B 为 30%，应为 140；删 A 后 130；删 B 后 100。同源替换不累加，基础值改变后按当前基础重算。

编码练习 异步打开的过期结果

设计练习 租约负责释放 每次打开有独立版本

假设所有调用和继续执行都在同一主线程。AcquireAsync 返回一个租约，租约拥有资源；每次取得的租约都必须 Dispose。关闭只使旧请求失效，不能假定底层加载被取消。

```
int version;
bool disposed;
Lease current;

async Task<bool> OpenAsync(string key)
{
    if (disposed) throw new ObjectDisposedException();
    int mine = ++version;
    ReleaseCurrent();
    Lease candidate = null;
    try
    {
        candidate = await AcquireAsync(key);
        if (disposed || mine != version) return false;
        current = candidate; // transfer ownership
        candidate = null;
        return true;
    }
    finally { candidate?.Dispose(); }
}

void Close()
{
    ++version;
    ReleaseCurrent();
}
```

为什么这个检查有用

A 请求较慢，关闭后又打开 B；B 先完成可以成为 current，A 之后完成发现版本过期，只释放自己取得的租约，不影响 B。异常向调用者传播；关闭当前内容由 ReleaseCurrent 完成。

必须问清的工程条件

资源合并加载由底层租约计数管理；取消后仍完成的请求必须有清理路径；销毁状态应禁止再次打开。多线程使用需要同步或主线程派发，这份版本号检查本身不保证线程安全。实际 UI 展示失败也要回滚其新租约。

最小测试

A 后完成 B 先完成；关闭后旧结果返回；打开失败；Dispose 后打开；同 key 两个租约；重复 Close。核对每个已取得租约恰好释放一次。

本页伪代码说明生命周期 完整可编译的泛型练习见配套源码

编码练习 空间网格与最近目标

题目 从邻近格中选择半径内最近的敌方存活目标

练习实现只把点目标登记到一个格。圆形或大型单位的碰撞需要扩大登记范围或查询范围，不能直接把点索敌实现当作通用形状碰撞。

```
int Cell(double x, double size)
    => (int)Math.Floor(x / size);

// Query(minCell ... maxCell) returns cell candidates.
Target best = null;
double bestD2 = radius * radius;
foreach (var t in candidates)
{
    if (!t.Alive || t.Team == myTeam) continue;
    double dx = t.X - x, dz = t.Z - z;
    double d2 = dx * dx + dz * dz;
    if (d2 > bestD2) continue;
    if (best == null || d2 < bestD2 ||
        (d2 == bestD2 && t.Id < best.Id))
    {
        best = t;
        bestD2 = d2;
    }
}
return best;
```

算法步骤

用位置减半径与加半径确定查询格的最小和最大坐标，遍历这些格取候选；过滤阵营与生命，再比较距离平方。距离相同时按 ID 决定，避免靠字典枚举顺序随机选中。

复杂度

建立点索引 $O(n)$ ，查询 $O(\text{访问格数} + \text{候选数})$ ，额外存储 $O(n)$ 。局部密度高或范围覆盖全图时仍可能接近扫描全部单位。计算平方前需要限制输入范围，极端坐标和 NaN 不能不处理。

测试列表

负坐标 -0.1 的格子应为 -1；半径边界应按约定包含；相同位置但同阵营不能选；死亡对象不能选；等距离按小 ID；空索引返回 null；跨格候选仍可选中。

优化追问

空间数据何时更新，查询是否读取快照？移动和死亡导致索引变旧怎样处理？多格登记如何去重？Job 内读写如何同步？生产实现还要处理 Entity 版本，不能只看实体索引。

编码练习 事件订阅与 LRU

题目 把调用期间修改订阅和缓存淘汰的语义说清楚

事件题的明确约定

单线程；发布时快照本轮订阅者；发布过程中新增或移除订阅，下次发布才体现。重复订阅算独立订阅，通过返回的独立 ID 取消。每个回调异常被收集，其他回调继续执行。

```
public List<Exception> Publish(T value)
{
    var snapshot = handlers.ToArray();
    var errors = new List<Exception>();
    foreach (var item in snapshot)
    {
        try { item.Value(value); }
        catch (Exception ex) { errors.Add(ex); }
    }
    return errors;
}
```

事件题追问

本轮已取消的监听仍会调用，是本题约定；若希望立即失效，要另做状态检查。嵌套 Publish 会再次创建快照，可能无限递归；可改成排队按批次执行。快照和错误列表会分配，不能宣称零 GC，热路径需评估复用。

LRU 的解题骨架

字典保存 key 到双向链表节点，表头最近使用，表尾最久未用。Get 命中就移到表头；Put 已有键时替换并移到表头；新键插头，超容量同时删除表尾和字典项。约定容量必须大于 0。

```
Get(key):
    missing -> return false
    unlink node; move to front; return value
Put(key, value):
    existing -> replace; move to front
    otherwise -> add node and dictionary entry
    over capacity -> remove tail from both structures
```

复杂度与测试

Get 和 Put 平均 $O(1)$ ，空间 $O(\text{capacity})$ 。容量 2：放 A、B，访问 A，再放 C，应淘汰 B。更新已有 A 不增加数量；容量 1、缺失键和重复更新都要测。

连接资源管理

LRU 只决定淘汰顺序。真实资源仍在使用时不能立即卸载；缓存所有权、业务租约和底层引用计数需分开。配套源码提供事件、LRU 及前四题完整实现和自动验证。

两轮模拟面试与自评

题目按 JD 与项目设计 练习时先闭卷作答 再核对

阶段	系统方向 60 分钟	战斗方向 60 分钟
0—5 分钟	90 秒介绍 再选框架案例	90 秒介绍 再选 P7 案例
5—20 分钟	异步开窗 资源归属 模块依赖	命令推进 Buff 合并 贡献撤销
20—35 分钟	GC UI 构建与热更兼容	空间查询 ECS 依赖与伤害时序
35—50 分钟	写事件或异步租约题	写鼓舞计数或最近目标题
50—55 分钟	旧系统重构与回归	弱网回合制扩展设计
55—60 分钟	反问职责与质量要求	反问战斗形态与权威边界

系统方向必追问

关闭后立刻重开，旧回包为什么不会显示？缓存命中为何仍要有使用者身份？一次热更能否被旧包使用？你的答案应明确生命周期、兼容边界与失败路径，而不只说“加判断”。

战斗方向必追问

两个人给同目标加同名 Buff，P7 如何合并？子弹未销毁时技能能否结束？技能伤害为何下一帧消费？分别用实际规则回答，再谈你会怎样改，避免把理想方案说成现状。

储备主程加试

接手一个正在上线且掉帧的项目，三周如何安排？参考：先建立版本、设备和问题基线，定位影响最大风险；第二阶段做可回滚的小范围修复并比较；第三阶段固化回归与发布检查。人员和时间不足时明确取舍，不承诺全面重写。

每题按四项各打 0 到 2 分

结论是否准确；能否给自己的具体例子；是否说明边界或代价；能否给验证方法。总分 0—8 分。低于 6 分的题回查原理，讲不清项目事实的题回查实现。这里是练习评分，不是录用预测。

面试后十分钟复盘

记录原问题、你的原回答、追问卡点和下一步补证据。不要只记“算法不会”。具体到“同名 Buff 合并键答错”或“没说明资源句柄所有权”，下一轮才有可检验的改进。

HR 沟通 薪资与反问

把职责 成果与期望讲清楚 不需要对招聘者披露私人债务

为什么想换工作

“我在当前公司两年多，职责从 Unity 客户端扩展到框架、服务端联调、运营后台、SDK 和 CI/CD。我希望下一阶段继续深耕客户端复杂系统，在团队协作、职责边界和回报上与承担的工作更匹配。”避免只说公司事情杂，也不编造内部矛盾。

期望薪资怎么表达

“我会结合负责范围看整体方案，目前希望固定税前月薪在 30—35K 左右，额外薪数与绩效另行确认。如果岗位需要更完整的主程职责，可以进一步讨论。”这是谈薪示例，最终口径由你确定；不把岗位上限当作已拿到报价。

如何保证月度现金流目标

你关注稳定月度到手，谈 offer 时核对固定月薪、试用期比例、社保与公积金基数、奖金条件和实际发薪日。让 HR 按你的扣缴情况试算，不用“全年包除以十二”替代每月到账，也不把某个税前数保证成固定到手数。

现在工资和其他 offer 怎么说

当前工资按真实情况说明，接着把讨论转向目标职责和能力。不虚构已有 offer、竞价或管理人数。在职状态和到岗时间按合同、交接与沟通结果确认，不先许诺立即入职。

Lua 或某项能力不满足怎么办

“Lua 我了解语法，没有正式项目使用经验。我的生产经验主要在 C#、HybridCLR 和客户端框架。想了解团队是否需要入职就接 Lua 核心模块；如果有学习窗口，我可以用小模块验证学习进度。”

你应当问面试官的事

该职位接下来三个月交付什么；新功能与维护各占多少；技术方案由谁决定；当前最难的问题是什么；性能与发布如何验收；岗位是新增还是补位；跨端和非研发工作是否长期计入职责。

拿到 offer 后核对

书面主体、职位、地点、固定薪酬、额外薪数条件、试用期、社保公积金、竞业约定和入职日期。比较具体条件，等录用条件与入职安排确认后再处理离职，不用口头意向替代正式 offer。

七天训练与面试前核对

每天按约两小时安排 可以随面试时间压缩 重点是闭卷讲述与手写

天	练习	完成标准
第 1 天	核对经历 录制介绍 讲 P7 技能	90 秒介绍 不把配置与状态混淆
第 2 天	Buff 词条 碰撞 手写计数和属性	手算贡献撤销 能解释边界
第 3 天	P15 基准 ECS Job Burst	准确报口径 解释结构变化
第 4 天	框架 UI 资源 异步编码	能画所有权 旧请求不污染新请求
第 5 天	C# GC 网络 算法与 LRU	每题 60 秒 编码有边界测试
第 6 天	P10 图形学 SDK 发布 针对 JD 补弱	说清全局与局部 合批与遮挡
第 7 天	两轮模拟 HR 谈薪和复盘	每轮记录三个明确卡点并修正

面试前需要你补齐的事实

实际团队分工和个人决策范围；每个项目上线状态及可公开展示内容；第一段工作时间；P15 基准记录能否公开；目标设备真实表现；Go 与 Node/Koa 各自对应的项目和职责；自己能现场讲清的一次真实故障。

资料准备

准备个人网站、可公开演示和一页项目关系图。公司私有源码与未获授权的资产不直接外发。把项目代号解释为公开展示用名称，面试官需要的是你的角色与解决过程。

开场前最后十分钟

只看自我介绍、一个主项目、P15 的测试条件和该岗位补强项。确认网络与屏幕共享；关闭无关窗口；现场编码先问输入和边界。遇到不熟悉的问题，明确已知部分再推导，不用术语填满空白。

资料完成后的使用方式

先按第 1 天试练一次，给每题打分；以后根据面试反馈更新 Word 版。基础题是检索与练习材料，真正的目标是能独立解释取舍，并用自己的项目证明判断。

岗位与个人材料来源

核查日期 2026 年 10 月 28 日 职位状态与招聘要求可能后续变化

S01 个人材料：BOSS 已上传的张文杰简历及以下个人网站。项目实现细节取自网站的 P7、P10、P15、共用框架和工程实践专题；本次没有访问公司私有代码仓库或重新运行 Unity 项目基准。

个人作品集

P7 技能与战斗专题

P10 流场专题

P15 属性专题

共用客户端框架

S02 波克城市 客户端开发工程师（unity）

S03 Garena 客户端程序-系统

S04 上海鹰角网络 高级客户端开发-系统向（明日方舟）

S05 淘米 高级 Unity 客户端开发（回合制战斗方向）

S06 游之趣 unity 游戏开发工程师（甲方自研）

S07 莉莉丝游戏 高级 U3D 客户端开发工程师

S08 莉莉丝游戏 Unity 客户端开发工程师（欧美 SLG）

S09 莉莉丝游戏 客户端储备主程

投递状态说明：截至本次筛选记录，8 个岗位已收藏，成功开聊 0，附件发送 0。平台的简历审核限制尚未确认解除，收藏不能记作已经投递。手册不依赖解除该限制即可使用。

岗位相关问题是根据职责推演的模拟题；没有宣称掌握公司的保密题库。展示薪资不代表录用薪资、固定奖金或可保证的月度税后收入。

技术原文一

按你的主要项目版本准备 API 与行为应回到具体包版本核实

S11 Entities 1.0 结构变化

S12 Entities 1.0 命令缓冲

S13 Entities 可启用组件

S13 补充 Burst 1.8 类型支持

S14 Addressables 1.21 内存管理

S15 Addressables 操作句柄

S16 HybridCLR AOT 泛型

S17 HybridCLR 代码裁剪

S18 Microsoft 装箱与拆箱

S19 Microsoft 异步编程模型

S20 Microsoft 线程安全集合

Unity 核心以 2022.3、Entities 以 1.0 为复习基线；P15 已有记录明确使用 Entities 1.0.16。
Addressables 1.21、Burst 1.8 和 HybridCLR 文档用于原理核对，不代表所有项目都使用该文档当前版本。

技术原文二

来源用于核对原理 示例设计与练习代码不是原项目源码

S21 Unity 2022.3 GPU Instancing

S22 ORCA 作者项目与论文入口

S23 Unity 2022.3 生命周期顺序

S24 Unity 2022.3 垃圾收集器

S24 补充 增量垃圾回收

S25 Unity 2022.3 CPU Profiler

S26 Unity 2022.3 UI 自动布局

S27 Unity 2022.3 旋转与方向

S28 Unity 2022.3 Root Motion

S29 IETF RFC 9293 TCP

S30 IETF RFC 6455 WebSocket

配套 C# 练习将计数、来源贡献、异步租约、空间网格、事件和 LRU 放在一个独立文件中，便于先遮住答案再手写。它是用于面试练习的小型实现，没有替代生产项目的线程、性能、平台和数据校验要求。