主要业绩

场景物件的制作

1. cry原有物件：无同步，无gameplay，主要是做表现，生成逻辑是在加载关卡的时候生成，逻辑使用flowgraph，每个脚本定义一个物件，无断线恢复逻辑
2. 挑战：需求多样化，物件之间相互引用，解决之道：组件化，在lua中拼装，初始化
3. server分为logic层和CE层封装创立之初，所有物件在逻辑层都有对象对应，逻辑对象都是动态生成的，关卡导出了静态物件数据，通过这些数据动态生成物件，flowgraph是基于静态物件的，即物件的id必须是确定的，动态生成使flowgraph无法使用
4. behavior tree引入，物件无法使用flowgraph，需要一种替代品，怪物使用的是behaviour tree，behaviour tree有很好的调试环境和编辑器，由于属性，行为定义都是物件内部，behaviour tree很容易和flowgraph等同使用
5. 由于flowgraph在表达物件间的相互引用和静态物件生成采用静态生成，动态绑定的机制，CE层的物件是在关卡
6. 解决需求的多样和组合逻辑，比如一个物件既可以加buff，又可以被攻击，还可以被破坏，组件式设计，C++提供基本的组件和行为，以及设置属性的接口（图示）
7. archetype引入，优化工作流程，数据存储在关卡数据外，修改属性甚至重新定义属性，只需要修改lua文件和archetype文件，在editor模式下直接生效，在launcher模式下简单重启server即可，这样关卡设计人员只需要关心物件摆放和相互作用，不用担心逻辑迭代引起的返工问题（画图解释），同时实现静态属性共享，内存节约
8. 断线重连的行为恢复，每个行为有加速模式，比如属性改变或者状态改变触发了行为，当重连恢复的时候，会启用加速模式直接完成，这样虽然效果略有折扣，但是减少了同步压力，逻辑制作也可以统一，只需要设置属性或者改变状态（如果定义状态）
9. 水的功能的抽象，cry原版水是不可修改的渲染节点，绕过raycast方式寻找水面，通过水的物理模块可以快速将一个渲染节点与制作的水面控制逻辑绑定，可以操控水的上升下降，材质更换，颜色等，
10. 引擎改造，植被系统的功能与物件逻辑相结合，物件既具有逻辑，也具有植被的特性，比如玩家进入草的展开和随风摇动

怪物特殊玩法：

1. 特殊移动方式，扩展cry的物理引擎，支持顶部和侧墙的制作流程管线，从编辑器到运行时移动，通过添加物理标志位的方式加快特殊移动区域的raycast检测，针对cry的物理逻辑对顶部进行分块，加快检索速度
2. 怪物行为产生特殊的道具的动态生成，利用archetype实现，逻辑用behaviour tree
3. 怪物与物件的互动，角色，道具，飞行道具系统与物件的整合流程确定

部分优化内容：

1. Sequence系统数据优化，早起写的系统，一些可以共享的静态数据开始只是做了副本基本的共享，提高到一个battleserver级别的共享，按照预期，一共可以节约副本个数\*30m内存，大概15\*40\*30 = 1.8g,引入rapidxml，通过整体加载，字符串弱拷贝，pool加快加载速度，整体了解了怪物猎人的server单进程多副本的数据共享架构
2. 在做怪物特殊移动玩法时候重构了怪物Locomotion系统，早期locomotion做在了behaviour tree中，同步模块需要重写一遍逻辑，做物件的思路推广，同化的功能应该用系统实现，behaviour tree还是做决策，功能划分清晰，简化同步功能，更方便做复杂功能

分享：lua的封装，分析了tolua，cryengine的lua框架，将lua 的ref，metatable，userdata，closure特性结合tolua以及cocos的lua引擎进行分析和举例

解决问题能力：时间差bug，蜘蛛丝bug，引擎可破坏物无法渲染bug，怪物瞬移bug