MW自动化专项

目录

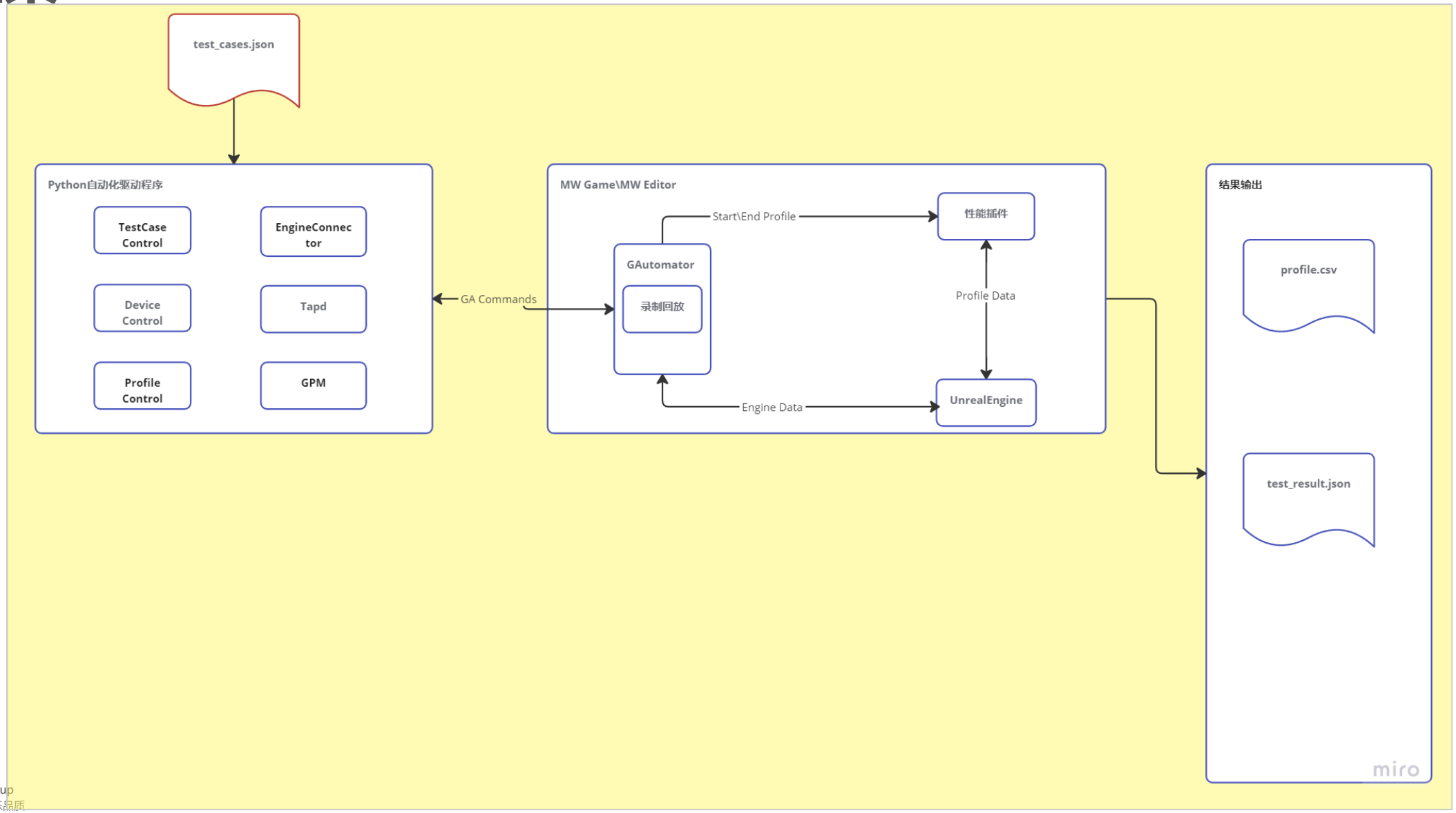
- 冒烟自动化框架
- 性能插件

项目背景介绍

MW是一款基于Unreal 5开发的RPG主机游戏，目前核心的玩法系统包括警匪，载具，战斗和AI。在自动化方面的需求主要在对主角的精准移动控制、驾驶控制和射击控制等。MW业务代码由蓝图和C++实现，没有使用脚本实现业务逻辑。

基于以上特点，利用GAutomator向上兼容Unreal 5开发了一套包含客户端录制回放功能和角色控制功能的冒烟自动化框架，同时适用于编辑器冒烟和性能自动化。

框架

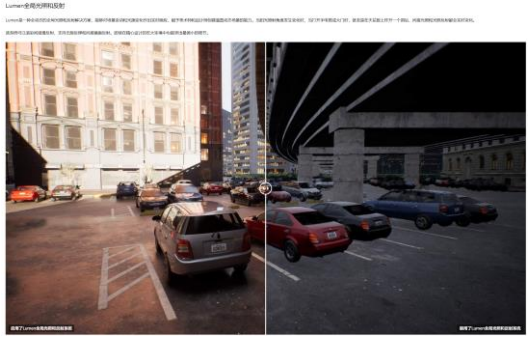

test_cases.json
Python自动化驱动程序
TestCase Control
EngineConnector
Device Control
Tapd
Profile Control
GPM
GA Commands
MW Game\MW Editor
GAutomator
录制回放
Start\End Profile
性能插件
Profile Data
Engine Data
UnrealEngine
结果输出
profile.csv
test_result.json
miro

冒烟自动化-UE5适配

众所周知UE5中新添加了有众多的新功能

如：Lumen全局光照和反射、Nanite虚拟几何体、虚拟阴影贴图(测试版)、时序超分辨率、路径追踪器改进(测试版) 等……

进行UE5项目的自动化工作首先要做的是对工具进行UE4->UE5的适配改造工作



冒烟自动化-UE5适配

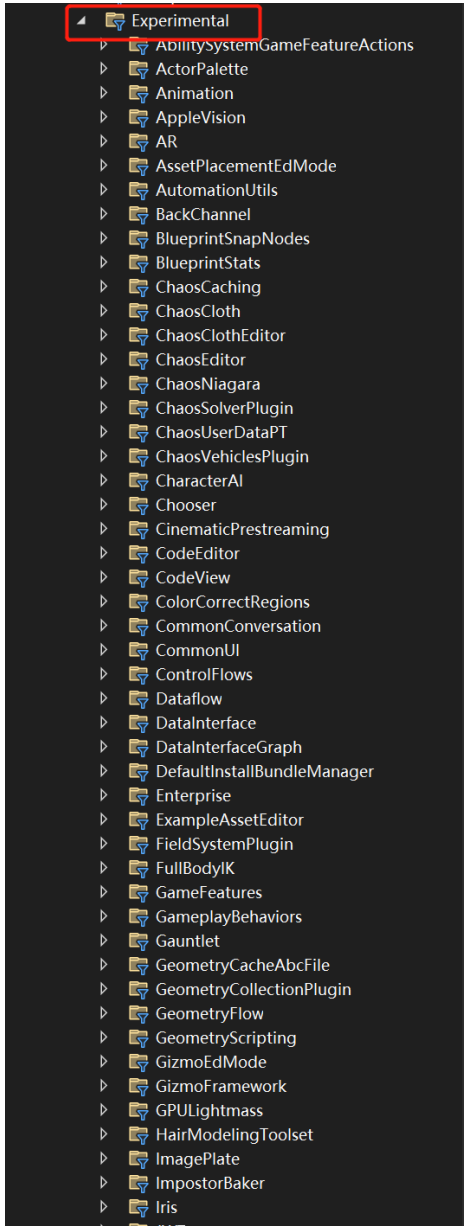
在具体代码中，大部分UE5更新功能以插件形式存放在Plugins/Expermental中

1、新增加类型，添加对应新类型方法

```
#if (ENGINE_MAJOR_VERSION == 5)
    //UE5 新加入类型：层级UI
    if (UCommonActivatableWidgetContainerBase* CommonActivatableWidgetContainer = Cast<UCommonActivatableWidgetContainerBase>(Widget))
    {
        TArray<UCommonActivatableWidget*> CommonActivatableWidgets = CommonActivatableWidgetContainer->GetWidgetList();
        for (auto CommonActivatableWidget : CommonActivatableWidgets)
        {
            UWidget* RootWidget = CommonActivatableWidget->WidgetTree->RootWidget;
            Children.Add(RootWidget);
        }
    }
```

2、依赖路径改变，对应查找新路径并修改

3、旧方法已被废弃，需要寻找新方法进行替代调用



冒烟自动化-录制回放

录制

注册监听InputKey和InputAxis事件

玩家操作同时记录位置信息及转向

输出——玩家操作+玩家人物位置的录制文件

[演示视频]

回放

根据录制文件，解析对应操作和位置信息

对游戏进行InputKey等操作输入，同时对比位置信息，如果差距过大则进行重置；实际运动速度远超回放进度，合并移动输入。

播放结束后返还控制权，通知回放结束

冒烟自动化-主角控制

行走

AddMovementInp
ut
该方法理论支持所有APawn，只要它的`MovementComponent`继承自UE自带的
实际实验可以正常触发行走动画及位置移动，其他方案则不会

如：

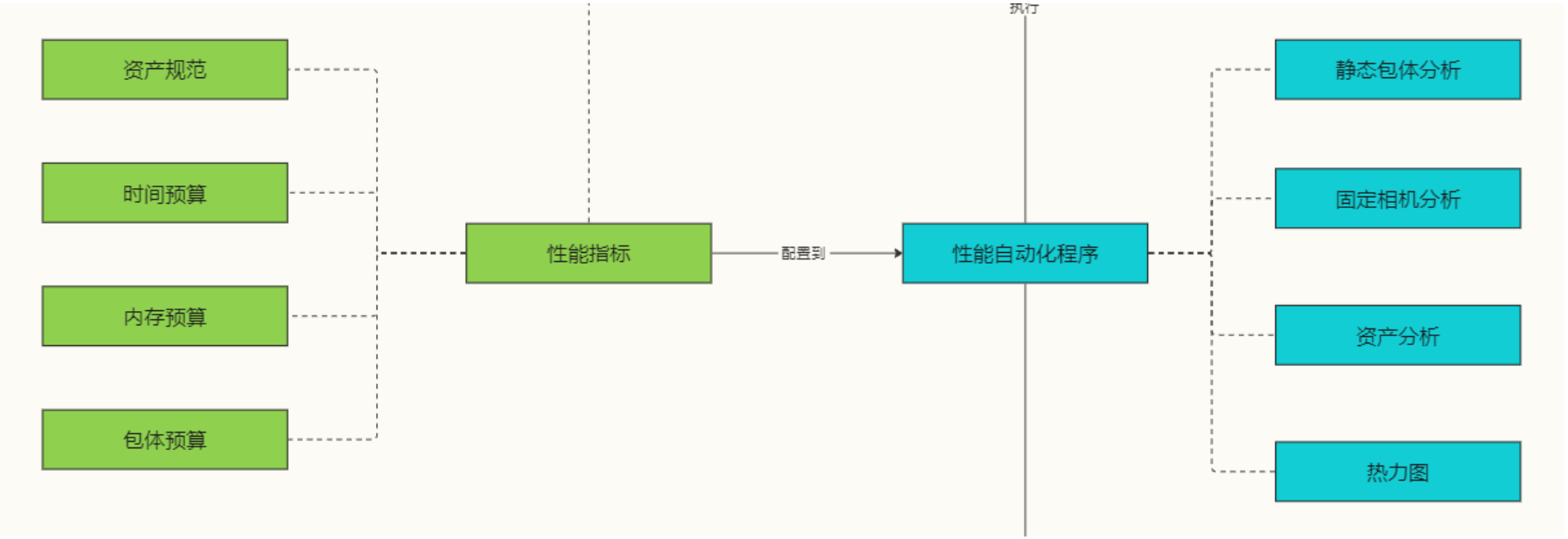
```
ACharacter->GetCharacterMovement()->Velocity += FVector(5.f, 5.f, 0.f);

UKismetSystemLibrary::MoveComponentTo(TopDownCameraComponent, Location, Rotation, false, false, 1.f, true,
EMoveComponentAction::Move, ActionInfo);
```

同时如果目标行进路径上有障碍物，会进行轻微的避障功能

演示视频

性能自动化

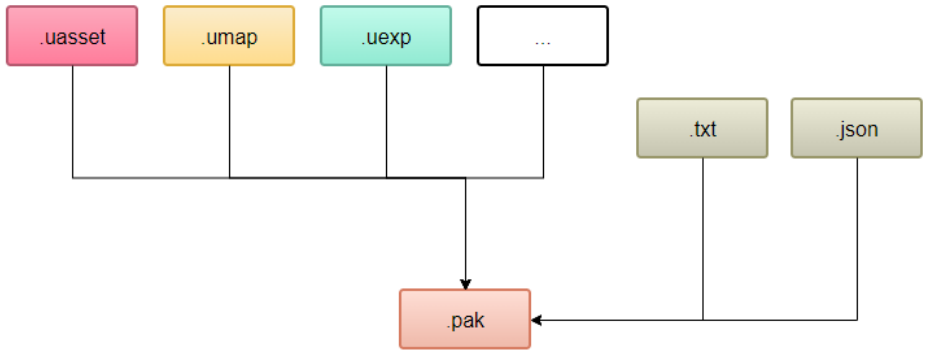

资产规范
时间预算
内存预算
包体预算
性能指标
配置到
执行
性能自动化程序
静态包体分析
固定相机分析
资产分析
热力图

性能自动化-包体分析

因UE5的改版，如果采用新加载的方案进行资源load，原有的UnrealPakViewer之类分析工具不再可用

Zen Loader 是新的下一代运行时加载器。作为大幅缩短下一代游戏机加载时间并更好地利用新的更快硬件的计划的一部分，它基于 EDL 加载器，但经过重构以减少开销，并具有新的 IO 机制，为数据访问提供低开销接口。

原UE4打包方式



新UE5打包方式

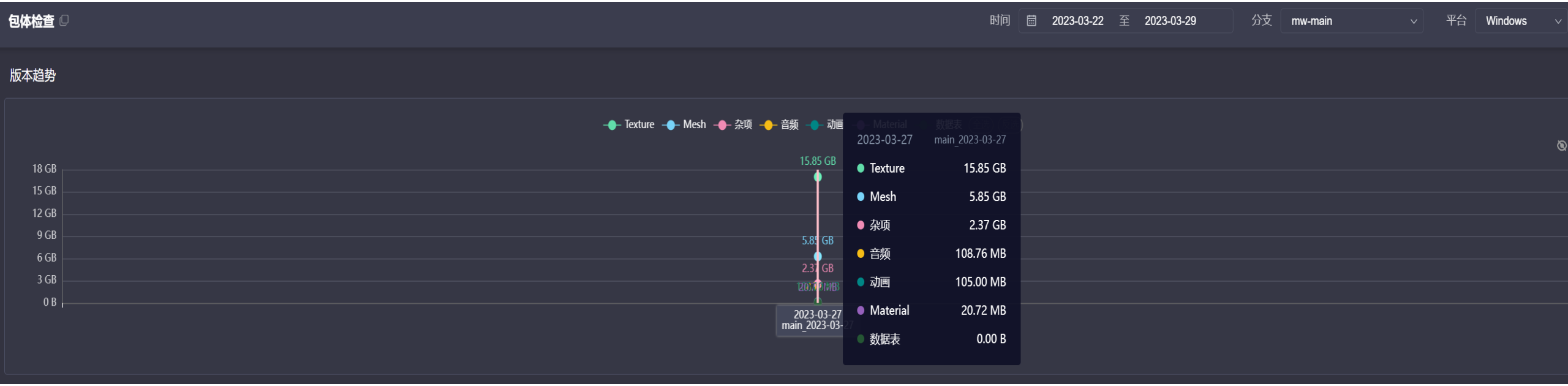
•pak文件：仅保存ini、ushaderbytecode、等没有依赖关系的文件。而将游戏的资产文件（UPackage）移动到ucas中

•utoc：目录文件 = pak的文件索引区 + package的summary

•ucas：存储uasset、umap、ubulk、uexp等。

•ucas仅存储文件内容，且在构建期，由于utoc的信息的存在，ubt会倾向将互相依赖的文件排列在附近，使得形成局部性优势，提高io的效率。

性能自动化-包体分析



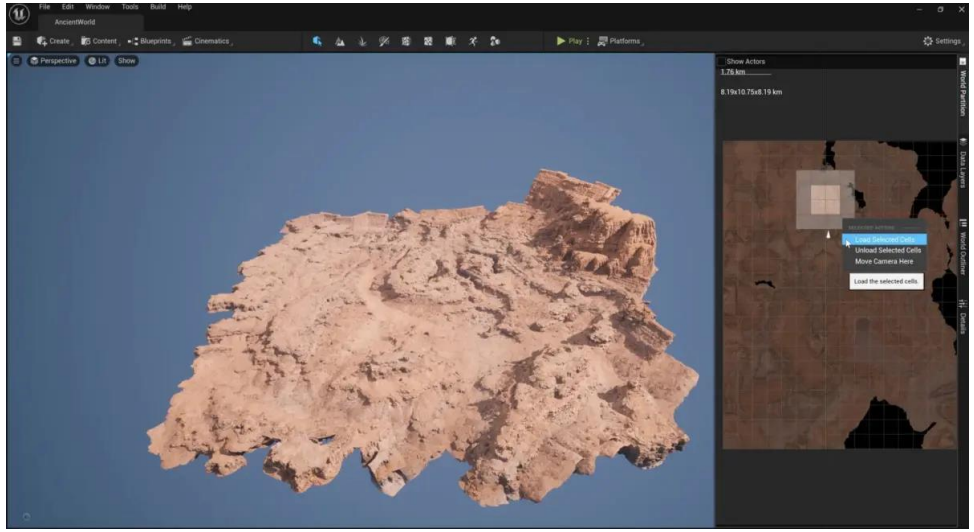
发现UE自带的unrealpak进行了同步升级适配仍可以使用，且可以导出包中的资源列表及大小

输入资源列表，从AssetRegistry中获取对应类型。可以直接复用原UnrealPakViewer中代码

性能自动化-WorldPartition

- 在UE4中，地图是美术同学手动划分成一个个的Level（Level分为Persistent Level以及Streaming Level两类，前者是常驻的，可以理解为地图的初始Level，后者则是在运行时根据需要动态加载卸载的），之后通过World Composition来完成这些Level的管理

- 在UE5中，地图不再需要划分成单个的Level进行编辑，也不需要手动对每个Level的位置进行调整，而是统一在一张大的完整的地图中进行编辑，Unreal会自动按照一定的size将之分割成一个个规整的Cell，这里的Cell就相当于此前UE4中的Level。



性能自动化-WorldPartition

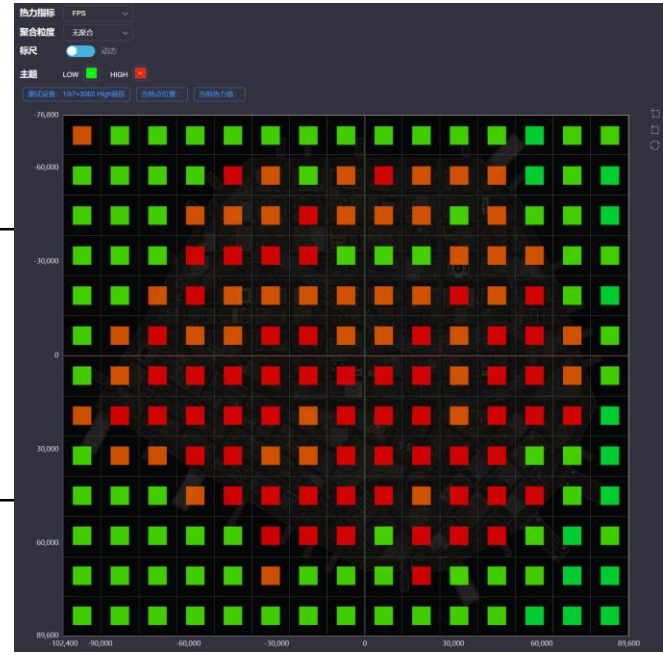
获取Cell大小

RuntimeSpatialHash->GetStreamingGrids()

MainGrid的CellSize

传送人物到下个Cell

TeleportTo(Position)



检查是否更新完

- 每当新增或者移除Actor的时候，并不会立马触发World Partition的更新

IsStreamingCompleted

记录当前Cell性能信息

Perf_Sampling

性能自动化-固定相机位截帧

上传GPM平台展示

性能自动化-热力图

调用性能连续采集指令

开始回放

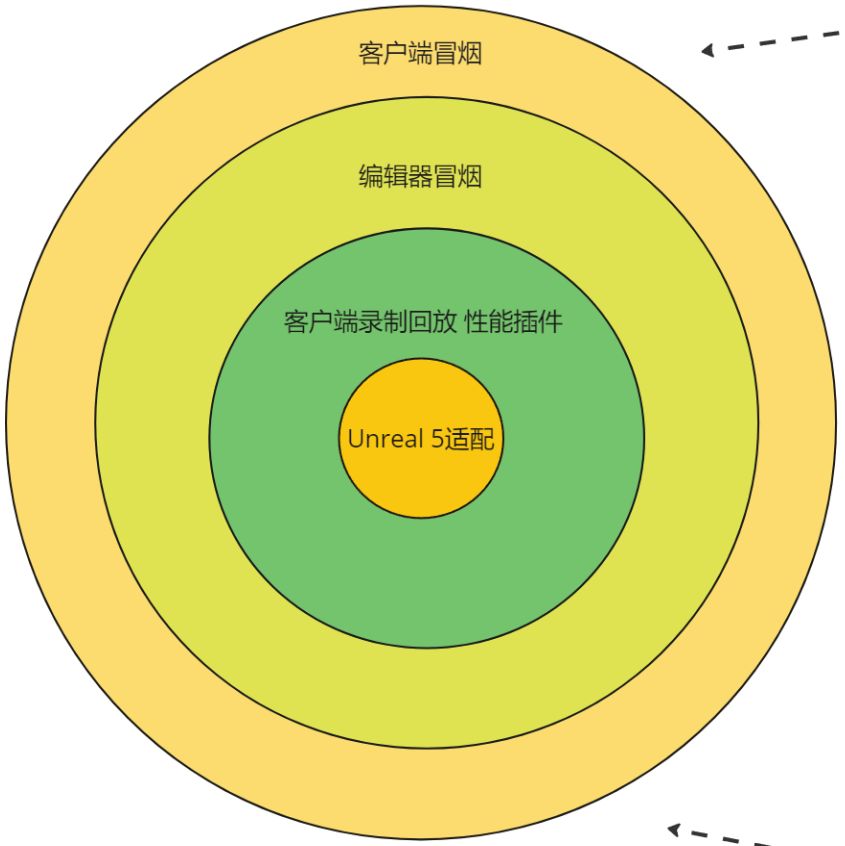
`FLatestGameThreadStatsData`

`RetireveLatestStatUnitData`

`RetrieveLatestFrameStatData`

上传GPM平台展示

总结



包体检测

WorldPartition

StatFrame

HeatMap

版本采集

GPM数据回溯

版本对比

TAPD

蓝盾CI