**实 验 三 报 告**

**（2020-2021学年第二学期）**

**3D游戏引擎架构设计基础**

**（Foundations of 3D Game Engine Architecture Design）**

|  |  |  |
| --- | --- | --- |
| 学生姓名： | 学号： | 年级，班级： |
| 成绩： | 任课教师签名： | 日期： |
| 实验题目： | | |
| 目录：   1. 实验内容描述: 引擎名称和版本,将分析的引擎模块名称 2. 将分析的引擎模块的主要功能描述 3. 将分析的引擎模块的主要类和类关系描述 4. 类的数据成员和成员函数描述 5. 总结 | | |
| 实验报告：   1. 本次实验分析的是：虚幻4的Actor及ActorComponent，以及一个ActorComponent - SceneComponent 2. 这个部分主要管理虚幻4的关卡中的所有的物件。 3. 主要关系类为Actor及ActorComponent。关系描述：    1. AActor 是可以在关卡中放置或生成的 Object 的基类。AActor 可能包含一组 UActorComponent，可用于控制 Actor 如何移动、如何渲染等。    2. ActorComponent 是定义可重用行为的组件的基类，这些行为可以添加到不同类型的 Actors 中。具有变换的 ActorComponents 称为 SceneComponents，而那些可以渲染的是 PrimitiveComponents。    3. SceneComponent 具有变换和支持附件，但没有渲染或碰撞功能。用作层次结构中的“虚拟”组件。   3.1 组件关系   1. UActorComponent 的关系   Component 不提供嵌套的功能，所有的 Component 只能挂载在 Actor 下。  从功能上来说，UE更倾向于编写功能单一的Component（如UMovementComponent）, 而不是一个整合了其他Component的大管家Component。  而从游戏逻辑的实现来说，UE也是不推荐把游戏逻辑写在Component里面，所以也没什么机会去写一个很复杂的Component.   1. USceneComponent 的嵌套   很多其他游戏引擎，有一种设计思路是“万物皆Node”。Node都带变换。  比如说你要设计一辆汽车，一种方式是车身作为一个Node,4个轮子各为车身的子Node，然后移动父Node来前进。  而在UE里，一种很可能的方式就变成，汽车是一个Actor，车身作为RootComponent，4个轮子都作为RootComponent的子SceneComponent。  UE 里通过组合 Component 来达到组合物体的功能。而这个组合体，再由一个统一的Actor对外提供逻辑功能。从而减轻组合复制物件的开销。  3.2 关键函数：AActor：获取坐标，通过转发至 RootComponent 的方式获取坐标  /\*\* Returns the location of the RootComponent of this Actor\*/  FORCEINLINE FVector GetActorLocation() const  {  return TemplateGetActorLocation(RootComponent);  }  template<class T>  static FORCEINLINE FVector TemplateGetActorLocation(const T\* RootComponent)  {  return (RootComponent != nullptr) ? RootComponent->GetComponentLocation() : FVector::ZeroVector;  }   1. 数据成员和成员函数描述    1. AActor       1. USceneComponent\* RootComponent;   根组件 – SceneComponent   * + 1. TArray<UActorComponent\*> ReplicatedComponents;   复制组件   * + 1. TSet<UActorComponent\*> OwnedComponents;   持有的组件   * + 1. TArray<UActorComponent\*> InstanceComponents;   实例化的组件   * + 1. TArray<UActorComponent\*> BlueprintCreatedComponents;   蓝图组件   * 1. UActorComponent      1. UActorComponent::OnComponentCreated   当一个actor在编辑器中或游戏过程中生成时，它会被任何原生组件调用。对于蓝图创建的组件，它会在该组件的构造过程中被调用。对于从关卡加载的组件，不需要这样做。   * + 1. UActorComponent::RegisterComponent   所有组件都在编辑器中注册，并在运行时创建它们的物理/视觉表示。这些调用可能分布在多个帧上，但总是在 PreRegisterAllComponents 之后。在 UnregisterComponent 调用将其从世界中删除后，这也可能会在稍后被调用。   * + 1. UActorComponent::Activate：   仅当组件设置了 bAutoActivate 时才会调用它。如果手动激活组件，稍后也会调用它。   * + 1. UActorComponent::InitializeComponent   仅当组件具有 bWantsInitializeComponentSet 时才会调用它。这仅在每个游戏会话中发生一次。   * 1. USceneComponent      1. USceneComponent\* AttachParent;   父组件   * + 1. TArray<USceneComponent\*> AttachChildren;   子组件   * + 1. TArray<USceneComponent\*> ClientAttachedChildren;   子组件   * + 1. FVector RelativeLocation;   组件坐标   * + 1. FRotator RelativeRotation;   组件朝向   1. 总结   从UE4的Actor的构建及组合构成中，我了解了一个复杂系统中，功能分解的重要性，面向对象设计的作用，有着优秀的设计，可以令系统有着极大的可拓展性，极大的可维护性。 | | |