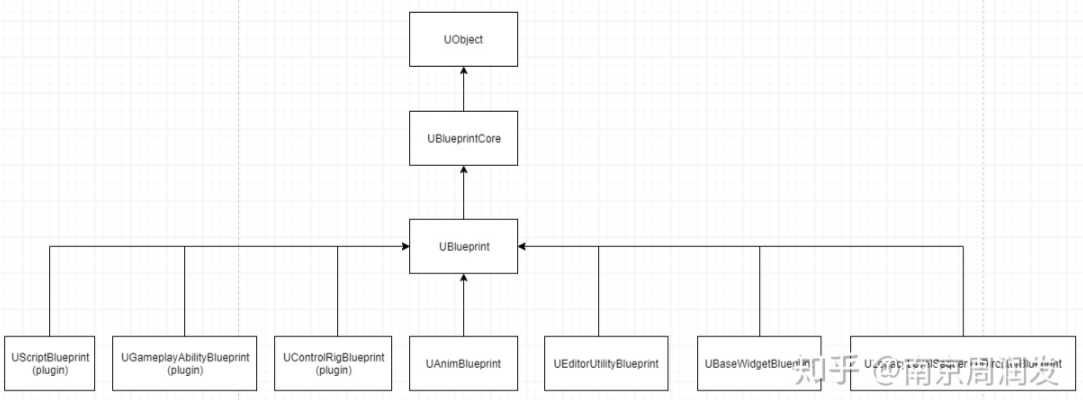
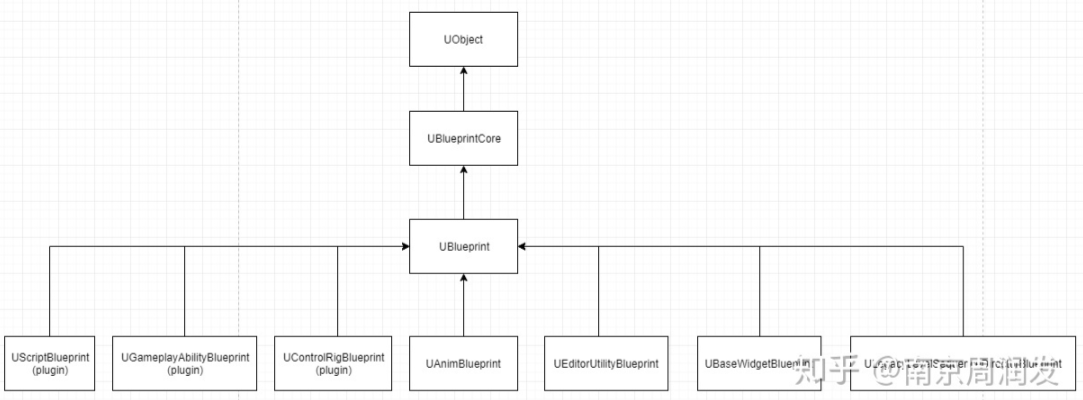
一、关于蓝图

蓝图运行中专门的蓝图虚拟机上，对蓝图做出修改后不需要像C++一样重新编译，而只需要重新生成一下蓝图的字节码，就可立即运行

二、UBlueprint



UBlueprintCore:作为蓝图的基类

TSubclassOf<class UObject> SkeletonGeneratedClass：指向Skeleton

TSubclassOf<class UObject> GeneratedClass:指向全编译class的指针，通常都是使用这个class，可以完整的描述一个蓝图

UBlueprint:

负责实现蓝图的主要功能，包括蓝图属性、蓝图函数、节点连接关系等。

主要成员变量： TSubclassOf<class UObject> ParentClass：指向蓝图类父类的UClass。会注册到AssetRegistryTags中，以支持蓝图预览。

TArray<UEdGraph\*> UbergraphPages：所有ubergraphpages，一个ubergraph就是编辑器左侧显示的一个eventgraph，里面会包含所有event的实现，在蓝图编译时，它们会被复制到一个大的ubergraph中。

TArray<UEdGraph\*> FunctionGraphs：所有函数graph，ConstructionScript也是一个函数。

TArray<UEdGraph\*> DelegateSignatureGraphs：所有事件代理Graph。

TArray<UEdGraph\*> MacroGraphs：所有这个蓝图实现的宏。

TArray<UEdGraph\*> EventGraphs：所有的Event声明。

TArray<class UTimelineTemplate\*> Timelines：蓝图中的Timeline，timeline需要被特别处理。

TArray<struct FBPVariableDescription> NewVariables：这个蓝图自己创建的变量，不包含从父类继承的，在编译时会添加到BlueprintGeneratedClass中。

TArray<class UBreakpoint\*> Breakpoints：断点集合，用于调试

其实，蓝图编译过程就是把UBlueprint描述的信息转换为BlueprintGeneratedClass的过程。

三、蓝图虚拟机

蓝图在编译后会生成对应字节码，引擎运行时会读取字节码，并交由蓝图虚拟机动态解释执行。

虚拟机的字节码为EExprToken枚举，在Script.h文件中定义。

enum EExprToken

{

    // Variable references.

    EX\_LocalVariable        = 0x00,    // A local variable.

    EX\_InstanceVariable        = 0x01,    // An object variable.

    EX\_DefaultVariable        = 0x02, // Default variable for a class context.

    ...

}

字节码中的每个指令都有相应的C++函数进行实现，它们在ScriptCore.cpp中定义

如EX\_LocalVariable的字节码的C++函数实现为

DEFINE\_FUNCTION(UObject::execLocalVariable)

{

    checkSlow(Stack.Object == P\_THIS);

    checkSlow(Stack.Locals != NULL);

    FProperty\* VarProperty = Stack.ReadProperty();

    if (VarProperty == nullptr)

    {

        FBlueprintExceptionInfo ExceptionInfo(EBlueprintExceptionType::AccessViolation, LOCTEXT("MissingLocalVariable", "Attempted to access missing local variable. If this is a packaged/cooked build, are you attempting to use an editor-only property?"));

        FBlueprintCoreDelegates::ThrowScriptException(P\_THIS, Stack, ExceptionInfo);

        Stack.MostRecentPropertyAddress = nullptr;

    }

    else

    {

        Stack.MostRecentPropertyAddress = VarProperty->ContainerPtrToValuePtr<uint8>(Stack.Locals);

        if (RESULT\_PARAM)

        {

            VarProperty->CopyCompleteValueToScriptVM(RESULT\_PARAM, Stack.MostRecentPropertyAddress);

        }

    }

}

IMPLEMENT\_VM\_FUNCTION( EX\_LocalVariable, execLocalVariable );

字节码对应函数：因此如果我们想加入新的字节码，主要任务就是提供一个对应的C++函数用于实现指令，当然，编译阶段也要生成对应的字节码

主要的一些实现类：

FFrame:栈帧，一个对蓝图UFUNCTION的调用就会创建一个FFrame