# FappyBird总结

1. 如何为对象添加一个组件？

**CreateDefaultSubobject<USceneComponent>(TEXT("Root"));**

此函数只能在构造函数中调用，创建一个组件（如果组件是功能组件则无需设置层级关系。如果是空间组件，则可以设置层级关系，也可以不设置，不设置则直接存在于场景中）

参数：组件名称

这是一个模版函数，模版类型，代表构建组件类型

1. 设置组件层级关系

**arm->SetupAttachment(root);**

此函数**只有空间组件**才能调用，主要目的将一个组件附着到另一个组件下。构造函数中调用，用来调整空间组件的层级关系

参数：附着到的组件指针。

1. 设置Actor的根组件

**SetRootComponent(root);**

将一**个空间组件，只能是空间组件**，设置给一个actor当作根组件。因为在构造函数中使用根组件，根组件不存在，所以需要创建一个新的。进行设置。

1. 在C++中如果需要使用2D内容，需要在 项目名.build.cs文件中引入模块“Paper2D”
2. 设置摄像机组件的透视模式

**camera->ProjectionMode = ECameraProjectionMode::Orthographic;**

1. 设置摄像机的正交窗口大小

**camera->SetOrthoWidth(800);**

1. 设置组件的相对位置，是相对位置和相对旋转

**SetRelativeRotation(FRotator(0, 90, 0));**

**SetRelativeLocation(FVector(0, -60, 0));**

1. 加载资源函数

LoadObjec

LoadClass

1. 开启组件物理模拟（只有空间组件可以开启，并且具备物理资产）

**pFlipBookComponent->SetSimulatePhysics（true）**

1. 锁定组件旋转轴，开启物理才有效果

**//获取到FlipBook组件的刚体对象**

**FBodyInstance\* body = pFlipBookComponent->GetBodyInstance();**

**body->bLockXTranslation = true;**

**//锁定Y轴偏移**

**body->bLockYTranslation = true;**

**//锁旋转**

**body->bLockXRotation = true;**

**body->bLockZRotation = true;**

**//创建transform物理约束器（必须要创建） 默认组件不具备约束器**

**body->CreateDOFLock();**

1. 为组件添加碰撞回调通知

**pFlipBookComponent->OnComponentHit.AddDynamic(this, &ABirdPawn::OnActorHit);**

**注意**：绑定的函数必须使用UFUNCTION宏标记

1. 物体开启物理模拟后，如果希望收到Hit事件，必须添加函数

**body->SetInstanceNotifyRBCollision(true);**

**注意body的获取方式**

1. 在类内获取GameMode

**GetWorld()->GetAuthGameMode();**

**返回GameMode指针，需要用Cast转换**

1. 获取当前操作玩家

**GetWorld()->GetFirstPlayerController()->GetPawn();**

1. 获取当前Playercontroller

**GetWorld()->GetFirstPlayerController()**

1. 绑定按键（只有Pawn和继承自Pawn的类才有下面的函数）

**void ABirdPawn::SetupPlayerInputComponent(UInputComponent\* PlayerInputComponent)**

**{**

**Super::SetupPlayerInputComponent(PlayerInputComponent);**

**PlayerInputComponent->BindAction("DoJump", IE\_Pressed, this, &ABirdPawn::ClickDoJumpEvent);**

**}**

1. 播放声音

**UGameplayStatics::PlaySound2D();**

1. 添加力的函数

**pFlipBookComponent->AddImpulse(FVector(0, 0, 800));**