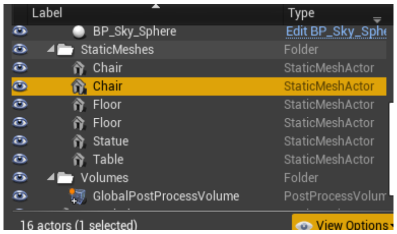
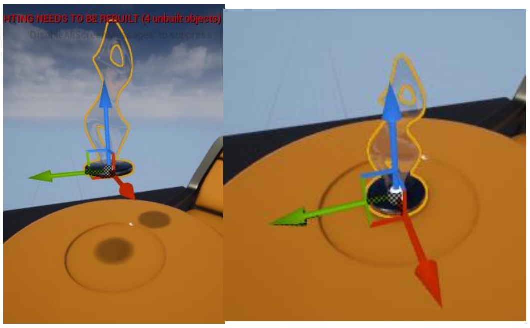
End: selected object will snap to the ground

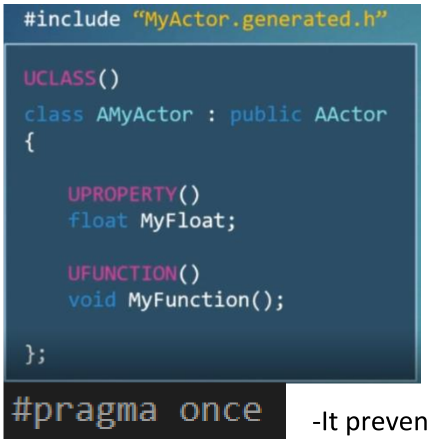


Label	Type
BP_Sky_Sphere	Edit BP_Sky_Sphe
StaticMeshes	Folder
Chair	StaticMeshActor
Chair	StaticMeshActor
Floor	StaticMeshActor
Floor	StaticMeshActor
Statue	StaticMeshActor
Table	StaticMeshActor
Volumes	Folder
GlobalPostProcessVolume	PostProcessVolum

16 actors (1 selected)

F: zoom into the selected object





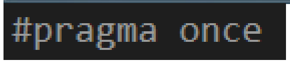
```
#include "MyActor.generated.h"

UCLASS()
class AMyActor : public AActor
{

    UPROPERTY()
    float MyFloat;

    UFUNCTION()
    void MyFunction();

};
```

-Must include special .h file which contain all the code associated with gathering data from the class for the reflection system

-class must be mark with UCLASS macro

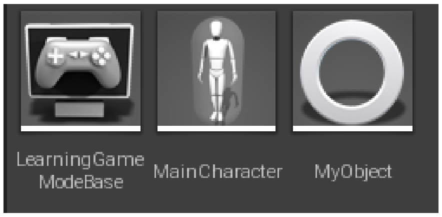
-variable must be mark with UPROPERTY macro

-function must be mark with UFUNCTION macro

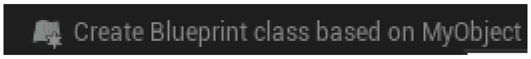


```
#pragma once
```

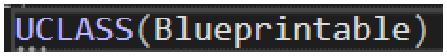
-It prevent u to include the same file twice



-we usually create a blueprint based on the C++ class we created

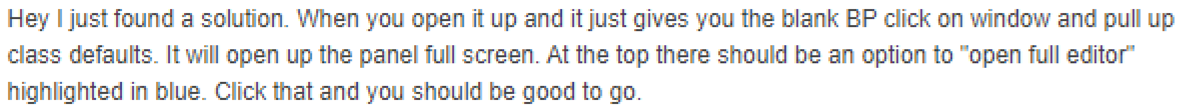


If blueprint cannot be created, then



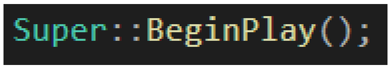
```
UCLASS(Blueprintable)
```

Solution for blank blueprint editor



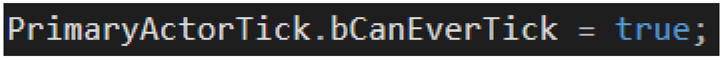
Hey I just found a solution. When you open it up and it just gives you the blank BP click on window and pull up class defaults. It will open up the panel full screen. At the top there should be an option to "open full editor" highlighted in blue. Click that and you should be good to go.

C++ Coding



```
Super::BeginPlay();
```

Super means the parent version of the function will also called



```
PrimaryActorTick.bCanEverTick = true;
```

When this is false means this character ticking functionality no longer works and it will save the resources (减少计算量)

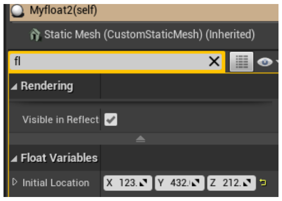
Preprocessor directive(start with #)
#include,#define,#if   #endif…,#pragma

功能是在 process file 之前吧这些 preprocessor directive 取代成 code

MACROS

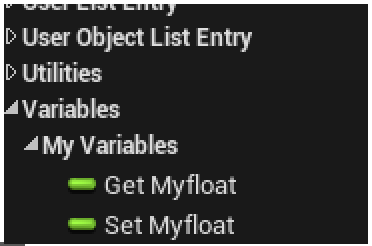
--In order to edit in blueprint:

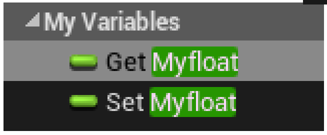
```
UPROPERTY (BlueprintReadWrite)

UFUNCTION (BlueprintCallable)

UPROPERTY(EditInstanceOnly)

UPROPERTY(VisibleInstanceOnly)
```



--To sort the function/variables,use category = "Name"

```
UPROPERTY(Category = "MyVariables")
```





--To print text on unreal engine log

```
UE_LOG(LogTemp, Warning, TEXT("This is an warning text!"));
```

-LogTemp means this type of log is temporary, Warning the text will be yellow and marked as a warning

Define

#define macro-name replacement-text

其实只是从 code 里面找到这个东西（identifier），然后 copy 后面的东西，把 code 里面全部的 identifier replace 成 relacement-text

Component

-a special type of Uobject designed to be used as subobject within actors

-allow parts of an actor to be swapped out to change its functionality

-all objects have a default root component, a scene component by default

C++小知识

Preprocessing stage

-This stage, the processor takes a C++source code file and deals with the preprocessor directives such as #include, #define, all the preprocessor directives will be replaced with code and output a “pure” C++ file without pre-processor directives.

Compiling stage

-When a code is being compiled, it will only go into compiling stage.

-processor will takes the pre-processor output and produces an object file full with binaries which is computer language.

Linking stage

-When a code is being build, it will go into compiling stage and linking stage. It need an entry point which normally it is the main() function, but it can be other than a main function.

-in linking stage, all of the function is linked with their definitions between files or places

-if there are two functions with same name, parameter, blablabla, it will get linking error cuz it doesn’t know which function to link for.

-when the function or variables or whatever is declare as static, it tells that the linking of this thing should only be internal, which means if the file containing this function is included in other file, the linking is only happen in this file but not other file that included this file.

static

-the function or variables with static is treated like private and in linking stage, it is only visible to the file contain the function and won’t go out of scope.

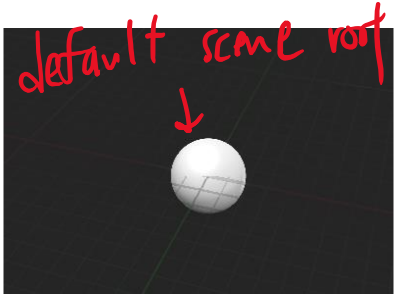
-and it means the functions or variables only has one instance

Difference between struct and class

-the only difference is class is private by default and struct is public by default

Actor Component (base)

1. Scene component



Default scene root component

(derived from scene component)

-it is just a reference, tell us there is an actor, and is not shown in the game

-it doesn’t have any visual, it only has location, rotation and scale

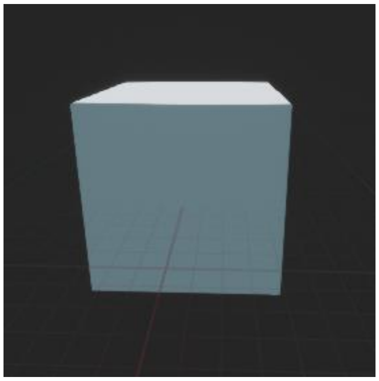
-it doesn’t have a mesh or geometry

!!it can be replaced by other component!!,and other component will be the root component

- it if all component is deleted, the default scene root component come back as an actor must have a default root component

-when setting the actor location, rotation or scale, it is actually the root component being set

2.Primitive component (derived from scene component)



Static mesh component

(derived from primitive component)

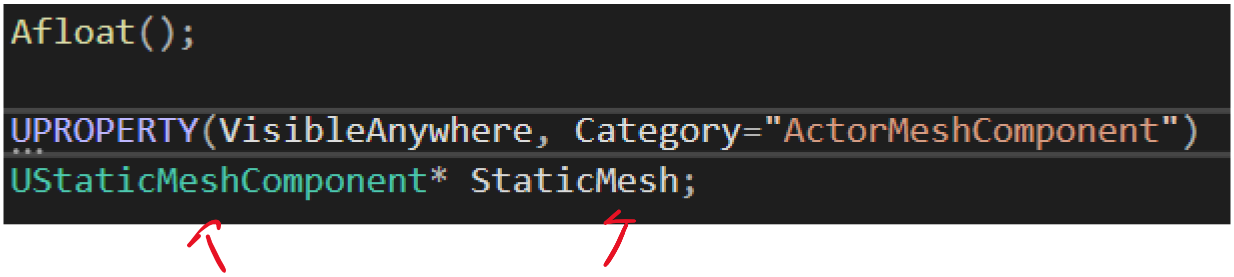
-it can have a mesh or geometry, collision information and etc.

Create static mesh component in c++

-anything that is not primitive type such as an integer or float, we use pointers

As its much more efficient

At the constructor in .h file:

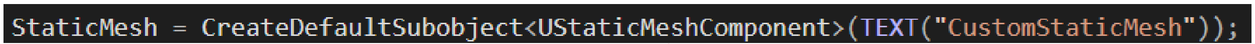


```
Afloat();

UPROPERTY(VisibleAnywhere, Category="ActorMeshComponent")
UStaticMeshComponent* StaticMesh;
```

It is a staticmeshcomponent and its name is StaticMesh

But we haven’t create a Ustaticmeshcomponent, we need to construct by using a special function called CreateDefaultSubobject

At the constructor in .cpp:

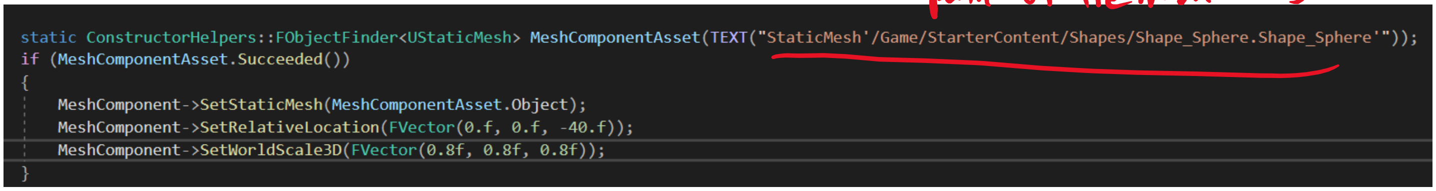


```
StaticMesh = CreateDefaultSubobject<UStaticMeshComponent>(TEXT("CustomStaticMesh"));
```

This is a template function which need the type of the object

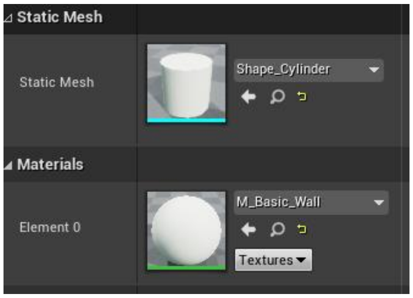
Hard coding adding mesh assets

path of the mesh assets



```
static ConstructorHelpers::FObjectFinder<UStaticMesh> MeshComponentAsset(TEXT("StaticMesh'/Game/StarterContent/Shapes/Shape_Sphere.Shape_Sphere'"));
if (MeshComponentAsset.Succeeded())
{
    MeshComponent->SetStaticMesh(MeshComponentAsset.Object);
    MeshComponent->SetRelativeLocation(FVector(0.f, 0.f, -40.f));
    MeshComponent->SetWorldScale3D(FVector(0.8f, 0.8f, 0.8f));
}
```

Generally do it in blueprint



Static Mesh

Static Mesh: Shape_Cylinder

Materials

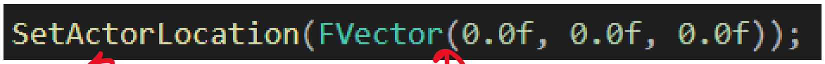
Element 0: M_Basic_Wall

FVector

-It has F prefixes because it do with float value

-It is a struct

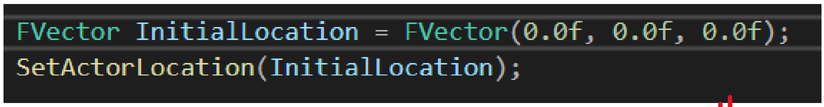
1.Set actor location



```
SetActorLocation(FVector(0.0f, 0.0f, 0.0f));
```

Function takes input parameter type of FVector . This will result the object set to the origin

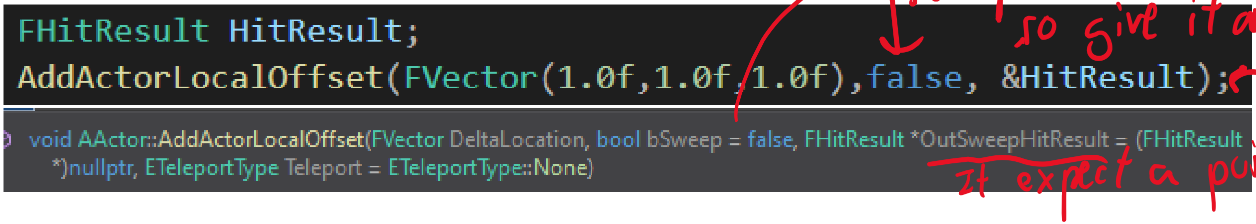
Set a FVector variable



```
FVector InitialLocation = FVector(0.0f, 0.0f, 0.0f);
SetActorLocation(InitialLocation);
```

(sth with collision) do not sweep perform

so give it an address

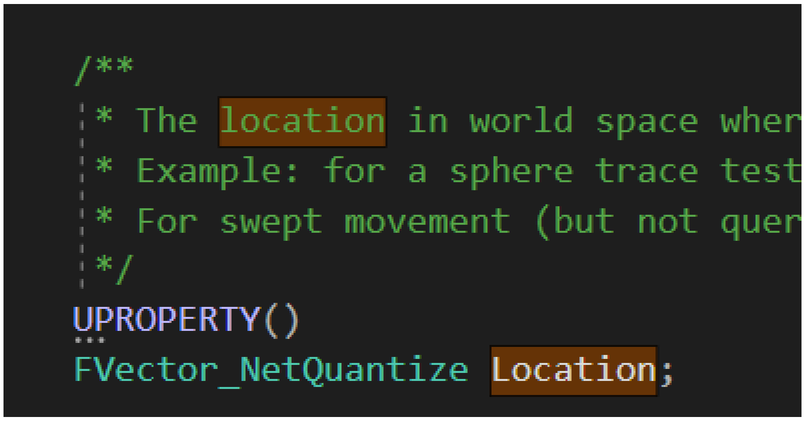


```
FHitResult HitResult;
AddActorLocalOffset(FVector(1.0f,1.0f,1.0f),false, &HitResult);
```

```
void AActor::AddActorLocalOffset(FVector DeltaLocation, bool bSweep = false, FHitResult *OutSweepHitResult = (FHitResult
    *)nullptr, ETeleportType Teleport = ETeleportType::None)
```

It expect a pointer

If sweeping is enable, the object wont clip through other object.

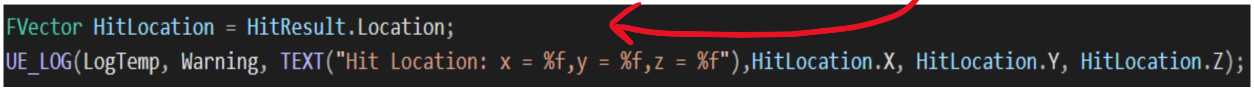


```
/**
 * The location in world space wher
 * Example: for a sphere trace test
 * For swept movement (but not quer
 */
UPROPERTY()
FVector_NetQuantize Location;
```

In definition of FHitResult, we see tis Location variables, this location variables will store the location of the object when collision is happening.

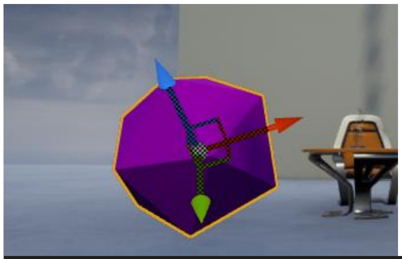
Hence we can save it into a variables using FVector Struct

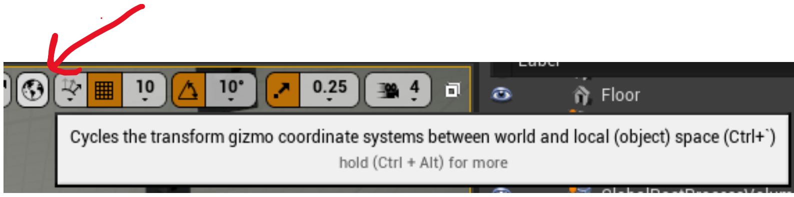
And print it in UE log



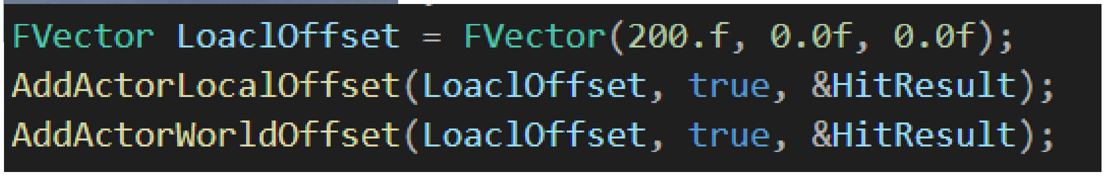
```
FVector HitLocation = HitResult.Location;
UE_LOG(LogTemp, Warning, TEXT("Hit Location: x = %f,y = %f,z = %f"),HitLocation.X, HitLocation.Y, HitLocation.Z);
```

AddActorLocalOffset 使用的 xyz 坐标是那个物体的坐标，可按这个看





Cycles the transform gizmo coordinate systems between world and local (object) space (Ctrl+`)
hold (Ctrl + Alt) for more



```
FVector LoaclOffset = FVector(200.f, 0.0f, 0.0f);
AddActorLocalOffset(LoaclOffset, true, &HitResult);
AddActorWorldOffset(LoaclOffset, true, &HitResult);
```

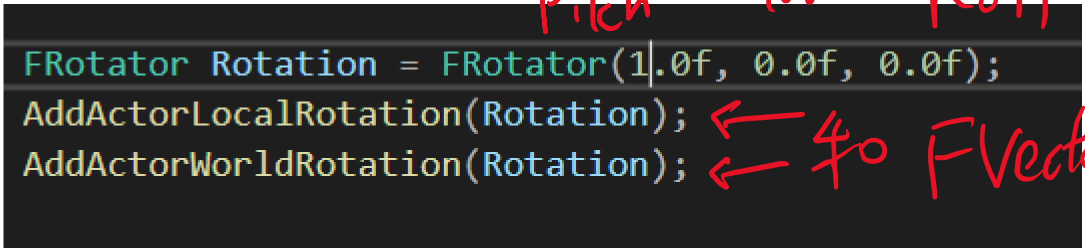
这两个的差别是 local 的会被移到自己的 x 坐标+200 的地方（因为有旋转所以会斜移移）

world 的会被移动到 world 坐标 x +200 的位置（平移）

X Y Z

Pitch Yaw Roll

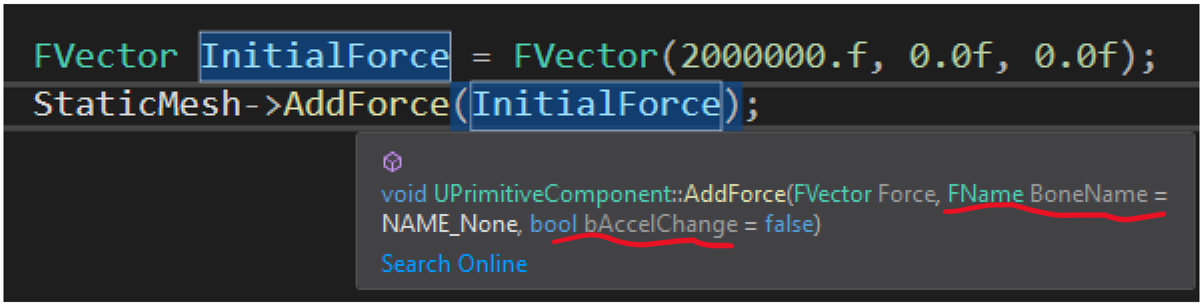
FRotator



```
FRotator Rotation = FRotator(1.0f, 0.0f, 0.0f);
AddActorLocalRotation(Rotation);
AddActorWorldRotation(Rotation);
```

和 FVector 同理

Force and Torque

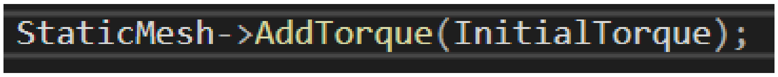


```
FVector InitialForce = FVector(2000000.f, 0.0f, 0.0f);
StaticMesh->AddForce(InitialForce);
```

void UPrimitiveComponent::AddForce(FVector Force, FName BoneName = NAME_None, bool bAccelChange = false)

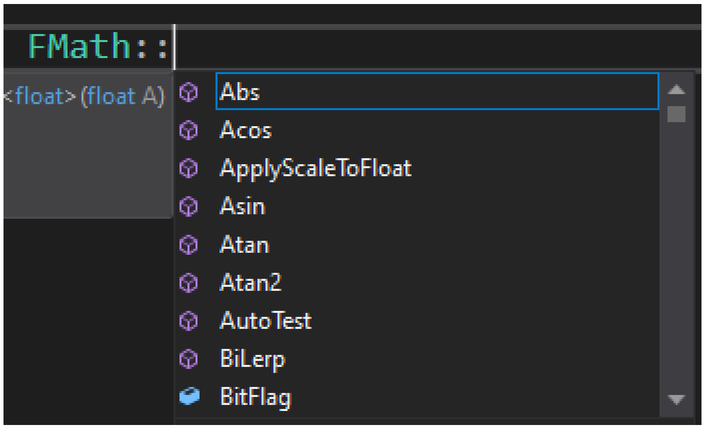
Search Online

There are three parameters u can give to function AddForce, but the second BoneName and bAccelChange If u didn't supply parameter then the things after = which is NAME_None and false will be set by default. So it's okay just give one parameter. Torque is same with Force.



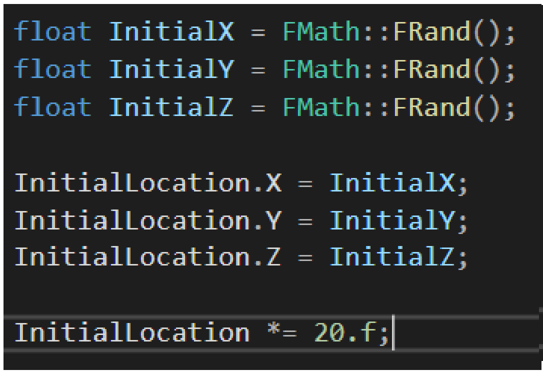
```
StaticMesh->AddTorque(InitialTorque);
```

FMath

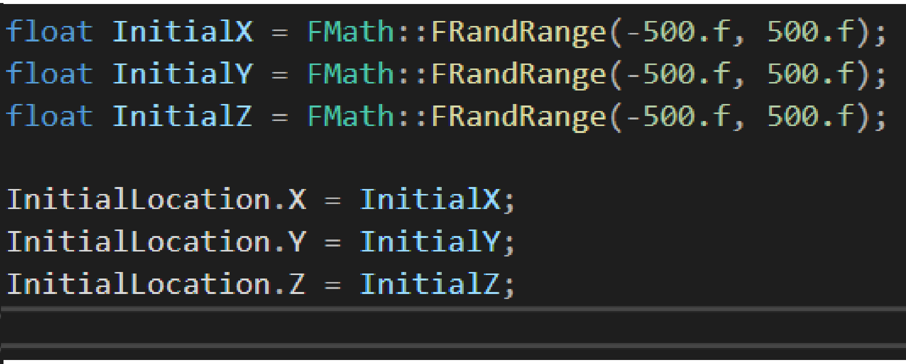


```
FMath::
```

Random number

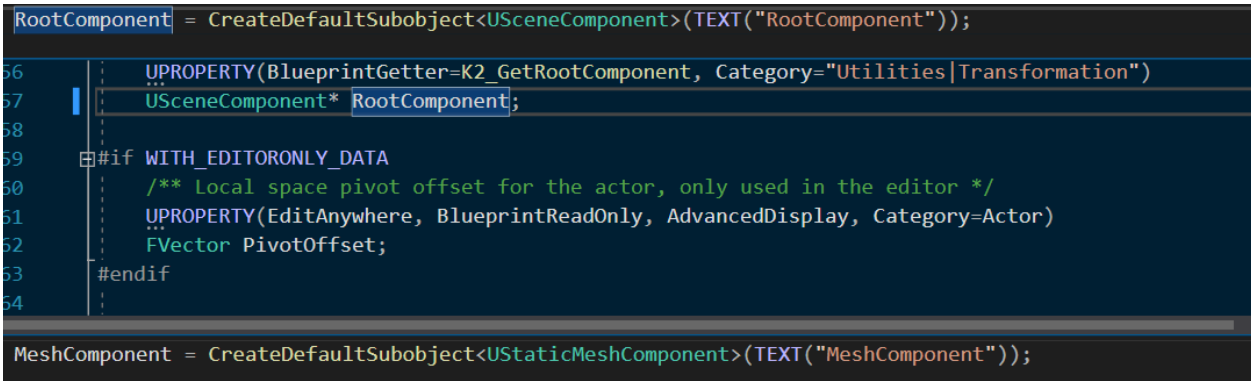


```
float InitialX = FMath::FRand();
float InitialY = FMath::FRand();
float InitialZ = FMath::FRand();

InitialLocation.X = InitialX;
InitialLocation.Y = InitialY;
InitialLocation.Z = InitialZ;

InitialLocation *= 20.f;
```



```
float InitialX = FMath::FRandRange(-500.f, 500.f);
float InitialY = FMath::FRandRange(-500.f, 500.f);
float InitialZ = FMath::FRandRange(-500.f, 500.f);

InitialLocation.X = InitialX;
InitialLocation.Y = InitialY;
InitialLocation.Z = InitialZ;
```

Pawn



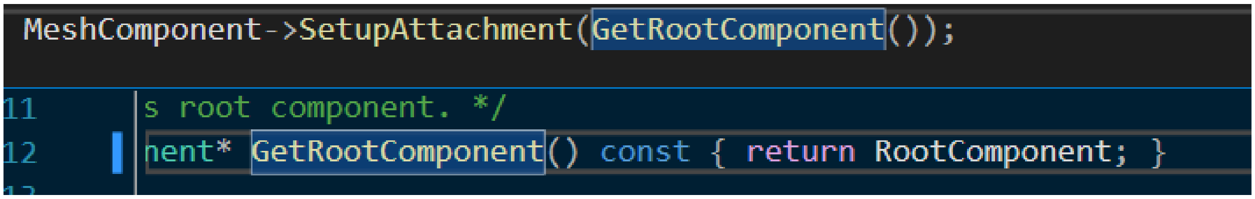
```
RootComponent = CreateDefaultSubobject<USceneComponent>(TEXT("RootComponent"));
```

```
    UPROPERTY(BlueprintGetter=K2_GetRootComponent, Category="Utilities|Transformation")
    USceneComponent* RootComponent;

#if WITH_EDITORONLY_DATA
    /** Local space pivot offset for the actor, only used in the editor */
    UPROPERTY(EditAnywhere, BlueprintReadOnly, AdvancedDisplay, Category=Actor)
    FVector PivotOffset;
#endif
```

```
MeshComponent = CreateDefaultSubobject<UStaticMeshComponent>(TEXT("MeshComponent"));
```

As Pawn inherited actor and it has rootcomponent, no need to create a rootcomponent again.

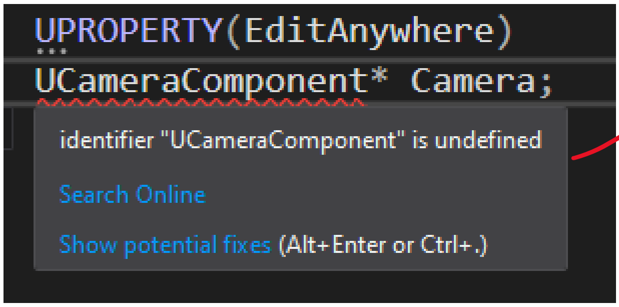
But it doesn’t have rootcomponent variables so it need to be defined in order to attach something on it.



```
MeshComponent->SetupAttachment(GetRootComponent());
```

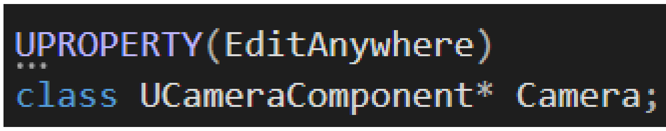
```
s root component. */
nent* GetRootComponent() const { return RootComponent; }
```

Camera:



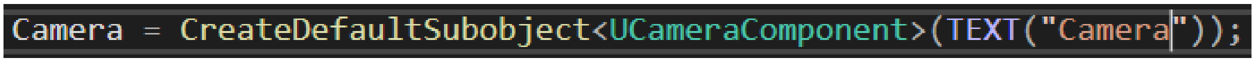
```
UPROPERTY(EditAnywhere)
UCameraComponent* Camera;
```

identifier "UCameraComponent" is undefined

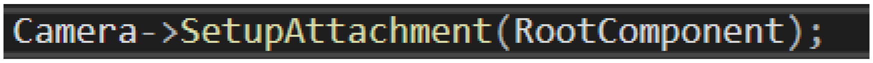


```
UPROPERTY(EditAnywhere)
class UCameraComponent* Camera;
```

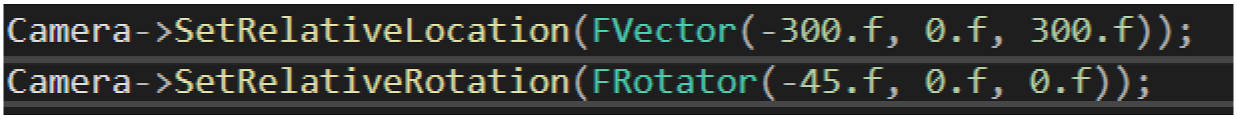
Forward declare and include the UCameraComponent.h file later



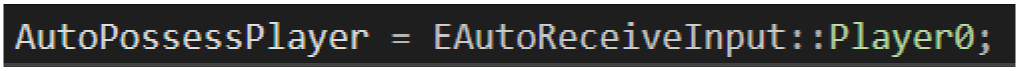
```
Camera = CreateDefaultSubobject<UCameraComponent>(TEXT("Camera"));
```



```
Camera->SetupAttachment(RootComponent);
```

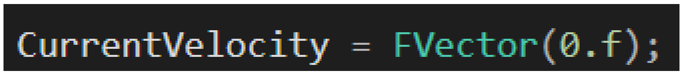


```
Camera->SetRelativeLocation(FVector(-300.f, 0.f, 300.f));
Camera->SetRelativeRotation(FRotator(-45.f, 0.f, 0.f));
```

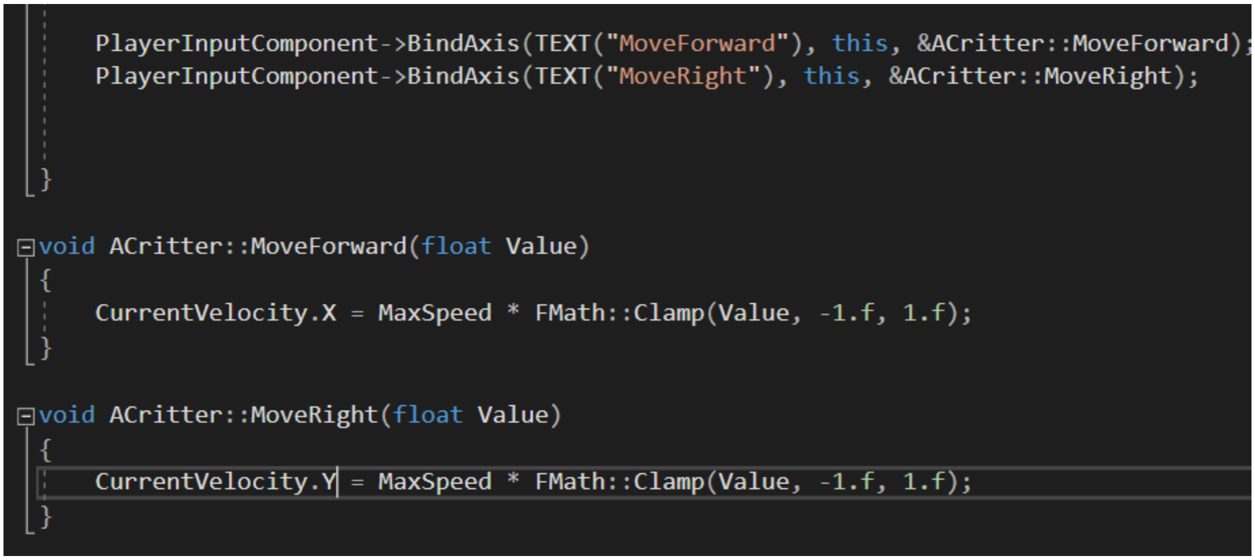


```
AutoPossessPlayer = EAutoReceiveInput::Player0;
```

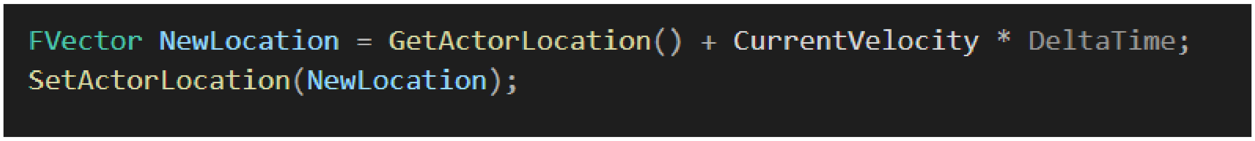
Control



```
CurrentVelocity = FVector(0.f);
```

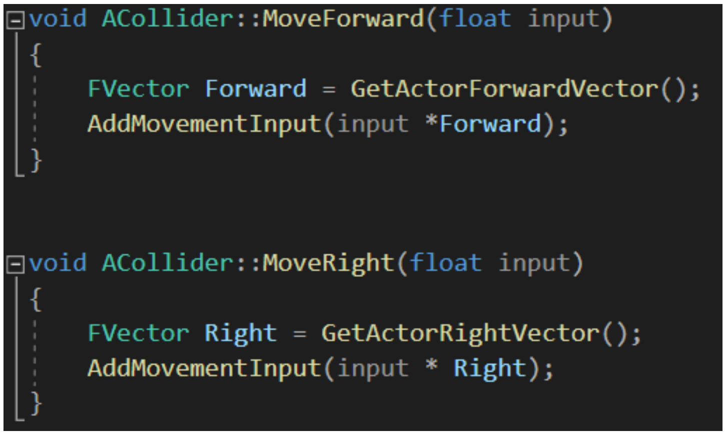


```
    PlayerInputComponent->BindAxis(TEXT("MoveForward"), this, &ACritter::MoveForward);
    PlayerInputComponent->BindAxis(TEXT("MoveRight"), this, &ACritter::MoveRight);



}

void ACritter::MoveForward(float Value)
{
    CurrentVelocity.X = MaxSpeed * FMath::Clamp(Value, -1.f, 1.f);
}

void ACritter::MoveRight(float Value)
{
    CurrentVelocity.Y = MaxSpeed * FMath::Clamp(Value, -1.f, 1.f);
}
```



```
FVector NewLocation = GetActorLocation() + CurrentVelocity * DeltaTime;
SetActorLocation(NewLocation);
```

Or another way



```
void ACollider::MoveForward(float input)
{
    FVector Forward = GetActorForwardVector();
    AddMovementInput(input *Forward);
}


void ACollider::MoveRight(float input)
{
    FVector Right = GetActorRightVector();
    AddMovementInput(input * Right);
}
```