# Atajos

**Compile** 🡪 CTRL + ALT + F11 (from visual)

# Clases Instances / Levels

## GameInstance

* **What is it?** A Global Singleton Class**:** tostoredata among levels.
* **Can store**: savegame, game config, assets load…
* To **Access**: World > GetGameInstance()
* Base Class: UGameInstance

## Gamemode

* **What is it?** The level itself (can be Lobby or Menu)
* To Access: World > GetAuthGameMode()
* In Multiplayer Gamemode only exists in server side.
* Base Class: AGameModeBase

## GameState

* **Store**: common global state of all players
* By default it has a **PlayerArray** of players
* Receives events when a player enters or exists the game.
* Lifecycle: the same as Gamemode (it is created by the Gamemode)
* Base Class: AGameStateBase
* In Multiplayer: created in Server Side and replicated to the clients

## PlayerState

* Store: player data independent of its Pawn/Character (score, level, XP)
  + (El Pawn/Character would be the representation, the avatar)

## PlayerController

* Has the input
* Can Own a Pawn
* The Player Controller is always the same (the Pawn can change)

## World

* Tiene **timer** para eventos
* Cada world está asociado a un nivel
* Los actores tienen un GetWorld para acceder a él.

# KEYS

## TSoftObjectPtr

Al definir un puntero para el nivel **UWorld\* MainMenu** yo tengo cargado el nivel en memoria.

En la GameInstance tenemos ptrs a todos los niveles y NO interesa tenerlos todos en memoria por lo que usamos **TSoftObjectPtr<UWorld> MainMenu**;

## MCVGAME\_API

Macro para que la clase se exponga a otros módulos.

Texto

Descripción generada automáticamente

## UStaticMeshComponent

Stores the mesh, its materiales, etc. (.h)

## TSubclassof<>

Type of class that is subclass of the template type

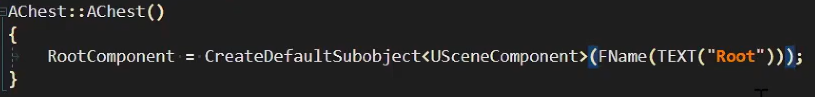
Interfaz de usuario gráfica

Descripción generada automáticamente

# Functions

## CreateDefaultSubobject

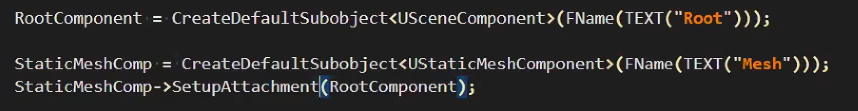
To create a Component. Must be called in the **constructor** of C++ classes (.cpp)



## SetUpAttachment

To parent a component

childComp->SetupAttachment(parentComp);



## OnComponentBeginOverlap

(In MCVGameCharacter.cpp)

// Register event to a function. To do this, the function should be marked as UFUNCTION() in .h

GetCapsuleComponent()->OnComponentBeginOverlap.AddDynamic(this,

&AMCVGameCharacter::OnCapsuleOverlaps);

void AMCVGameCharacter::OnCapsuleOverlaps(

UPrimitiveComponent\* OverlappedComponent,

AActor\* OtherActor,

UPrimitiveComponent\* OtherComp,

int OtherBodyIndex,

bool bFromSweep,

const FHitResult& SweepResult

){

if(UInventory\* OtherInventory =

OtherActor->FindComponentByClass<UInventory>()){

Inventory->TransferAllContentsFrom(OtherInventory)

}

}

En foto:

Texto

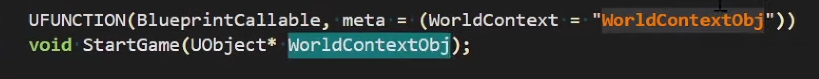
Descripción generada automáticamente



## UGameplayStatics::OpenLevelBySoftObjectPtr

UGameplayStatics::OpenLevelBySoftObjectPtr(this, GameConfig.GameLevel)

* This = const UObject\* WorldContextObject
* GameLevel = const TSoftObjectPtr<UWorld>



Se le puede pasar como meta información el WorldContextObj como parametron.

## GetWorld()->LineTrace

There are several types:

* Multi: a ray can return several collisions
* Single: only the first one
* ByChannel:
* ObjectType: types of collisions (pawn, for example)

Texto

Descripción generada automáticamenteExample:

And QueryParams can Ignore myself Actor

## UE\_LOG

* LogTemp
* Display, Warning, Error
* TEXT(“Interact Actor %s”, ActorName)

## PlayerController->Possess(Pawn) / UnPossess

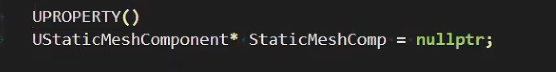
It allows to possess a Character with the input (changes the camera too).

Example: when you look at the other Character and press E, you possess the other one

Texto

Descripción generada automáticamente

# Attributes



## UPROPERTY

When the **UStaticMeshComponent** above is deleted, the UPROPERTY allows the garbage collector to delete and warn us that the object is not usable.

### Tags

Can be added to the UPROPERTY



* **EditAnywhere**
* **BlueprintReadWrite**: can be modified in runtime
* **BlueprintReadOnly**: can NOT be modified in runtime

## UFUNCTION()

This attribute is necessary when we want that a function could be called from blueprints (.h)

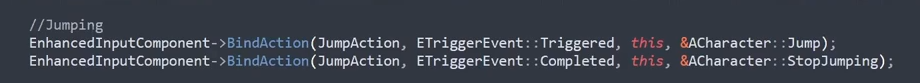
UFUNCTION(BlueprintCallable)

void TransferInventoryContents(UInventoryComponent\* OtherInventory);

# Input

In MCVGameCharacter (SetupPlayerInputComponent)

Graphical user interface, text, application

Description automatically generated

ETriggerEvent::Triggered

ETriggerEvent::Started 🡪 when it started to be pressed

ETriggerEvent::Completed 🡪 when it started to be realesed

# Data Structures

* **TArray<>** …………………… equivalent of **std::vector<>**
* **TMap<>** …………………… equivalent of **std::map<>**
* **FString** ……………………
* **FName** ……………………
* **FText** ……………………