# 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)

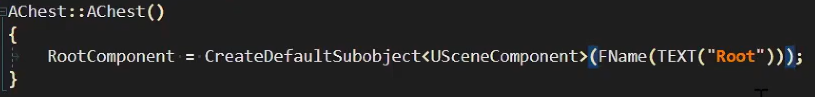
## World

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

# Functions

## CreateDefaultSubobject

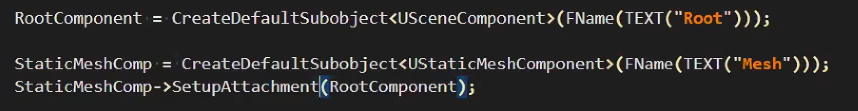
To create a Component. Must be called in the **constructor** of C++ classes (.cpp)



## SetUpAttachment

To parent a component

childComp->SetupAttachment(parentComp);



## UFUNCTION()

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

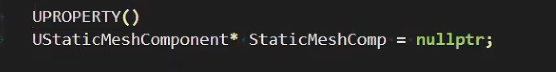
UFUNCTION(BlueprintCallable)

void TransferInventoryContents(UInventoryComponent\* OtherInventory);

# Attributes

## UStaticMeshComponent

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



## 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
* …

# Data Structures

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