**Gamemode:** a

**GameState**: info global del juego (puntauación del equipo en un juego online por ejemplo)

**PlayerController**: hace de interfaz entre el input y el personaje. Un player controller puede controlar varios personajes.

Diagrama

Descripción generada automáticamente

**Pawn**: avioneta, coche, personaje

**Character**: personajes humanoides (heredan de pawn)

# Estados del juego

**GameInstance**: es para nuestra clase Singleton

* Init: al iniciar el juego
* Shutdown: cuando se cierra

**Gamemode**: gestiona las reglas (al entrar un jugador, spawnea un personaje, o me espero a que estemos todos y luego spawneo a todos a la vez).

**GameState**: es parecido, está mas orientado a datos. En singleplayer lo podríamos meter todo en gamemode, pero es interesante usarlo bien. Podemos guardar puntuación o estado de los cofres si está abierto o cerrado. Cosas así las puede controlar el gameState.

* GameState datos relativos a la partida (cofres, puntuaciones de equipo)
* Gamemode lógica a mas bajo nivel (spawn de players)

# Estados del jugador

**PlayerState**: experiencia del jugador y cosas así.

**Player** **Controller**: mas de mecánicas, input,

Al cargar el nivel se van a perder datos. SI queremos conservarlos lo guardamos en gameinstance

BlueprintReadWrite: puedes modificarlo desde blueprint

BlueprintType: puedes acceder desde vlueprint

EditDefaultsOnly: solo blueprints

EditAnuwhere: todos lodos

Creamos ULevel en la GameInstance

Texto

Descripción generada automáticamente

Creamos la clase

Interfaz de usuario gráfica

Descripción generada automáticamente

MyGameInstance y le hemos puesto las 2 referencias a niveles, por lo que en los detalles vemos esto.

Texto

Descripción generada automáticamente

Llamar siempre al Super a no ser que estemos seguros de que mejor no hacerlo.

Whole Tomato Visual Assist

Visual Assits

UFUNCTION sirve para que se pueda llamar desde Blueprint.

COLISIONES: HABLA A LAS 20:03 o así

2 pawns nunca harán overlap.

Texto

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

Texto

Descripción generada automáticamente

To bind Actions we can do it from here: SetUpPlayerInputComponent

Texto

Descripción generada automáticamente

## GameModes

We have

* **MCVGameGameMode:** game
* **MCVMenuGameMode**: we clone the MCVGameGameMode and we delete the code for spawning the player because we dont need it in the menu.

Each one for each level.

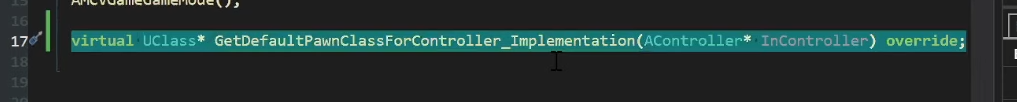
Windows > World Settings: GameMode Override: we specify the gamemode of each Level (we change it manually in each level).

## Virtual functions that can be override

### - GetDefaultPawnClassForController:

We can add logic to override, so the first character is X and the second another one.

BlueprintNativeEvent: needs a **GetDefaultPawnClassForController\_Implementation:** lo hace solo el preprocesador de unreal



Texto

Descripción generada automáticamente

This is a function we can override

### HandleStartingNewPlayer



The implementation of the timer and function not even the professor knows:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente