|  |  |
| --- | --- |
| SCRIPTS EN USO COMO COMPONENTES | |
| CATEGORÍA | DESCRIPCIÓN |
| A | - Define variable “instancia”.  - Usa patrón Singleton para asegurar un único componente script aunque haya cambios de escena.  - Navega entre escenas. (Con el uso de “DontDestroyOnLoad (gameObject);” - Define enumerador con estados o fases por las que pasará el juego durante toda la simulación.  - Al establecerse, activa su primer hijo “OtrosGestores” (Gestores complementarios de Cat. B).  - Define eventos que se irán invocando para que reciben la notificación aquellos scripts componente que se hayan querido subscribir. |
| B | - Define variable “instancia”, pero sin uso de Singleton.  - Al estar emparentado en el GameManager como nieto, también navega entre distintas escenas.  - Se subscribe a diferentes eventos para recibir notificaciones.  - Se encarga de definir variables y procesos que guarden relación con un cometido concreto.  - Debe estar activo en un único componente script. |
| C | - Define variable “instancia” sin uso de Singleton.  - No navega entre escenas, por lo que solo se usa en una escena de Unity concreta.  - Se encarga de definir variables y procesos que guarden relación con su cometido.  - - Debe estar activo en un único componente script. |
| D | - No definen variable “instancia”, tampoco uasn singleton.  - Se subscriben a diferentes eventos para recibir notificaciones.  - Se encargan de definir variables y procesos que guarden relación con un cometido concreto.  - Se suelen utilizar múltiples veces de manera simultanea. |

Scripts usados en la práctica “PatosLocos!”:

* Escena de Unity: “MenuInicio”  
  + GameManager.cs (A) => Define variable “static GameManager instancia” (Dentro del Singleton)
    - OtrosGestores (inicialmente desactivado)
      * MusicM.cs (B)
      * SoundM.cs (B)
      * TransitionM.cs (B)
      * PlayerDataManager.cs (B)
  + MainMenuM.cs (C)
* Escena de Unity: “Gameplay”  
  + HudM.cs (C)
  + PauseM.cs (C)
  + Rifle.cs (C)
  + Pato.cs (D)
  + Oleaje.cs (D)