17/02/2023 RECOMENDACIONES (según Jesús en el lab):

- ¡NADA DE RAMAS! No hacen falta.
- INTEGRACIÓN CONTINUA: Commitear y pushear nuestros cambios todos los días mínimo 1 vez.
- CADA CUAL SU ESCENA PARA TEST INTEGRAMOS EN UNA CUANDO FUNCIONEN.
- NUNCA SUBIR ALGO QUE NO COMPILA.

plus: organización de github:
Haremos la arquitectura básica (*gamemanager* + *movimiento*) en la rama main y ya luego cada HU feature según assignees por **escenas propias de cada persona** y **SIN RAMAS** (todos en **main**).

Buenas prácticas y convenciones de programación:
(Esto es para facilitar el trabajo conjunto, no para presentarlo ante nadie)
Disclaimer: esto depende de a quién le preguntes y va por gustos, aunque hay algunas más generales que otras y muchas dependen del lenguaje usado. Usamos la convención camel.

Y **comentarlo!!!** después de cada sesión procurad coger la costumbre de comentar lo que hayáis hecho, vuestro future self y todo el mundo os lo agradecerá (no tiene por que ser de manera exhaustiva tampoco).

Nombres:

guion bajo antes de nombre de variable **privada** en **minúscula**
"*_instance*" variable privada

métodos primera letra **mayúscula**, independientemente de si público o privado (se da por hecho que son descriptivos de lo que hacen)
"*Instance*" método público getter de la variable privada.

si son **varias palabras**, siempre la siguiente empieza por **mayúscula**, nada de separarlo por guión bajo ni nada.
"*_transRights*"
"*TransRights*"

las **constantes** son siempre **mayúsculas**
"PAPOPEPO = 4206669"

a mi personalmente no me podría dar más igual si en inglés o español

Ej:
(SCRIPT GAMEMANAGER)

```
private int _coins = 0;
private GameManager _instance = this;
public GameManager Instance { get { return _instance; } };
lo del get / return es así en c#
public static void AddCoin() { _coins++ };
```

GameManager.Instance permite acceder al GameManager DESDE CUALQUIER SITIO si la instancia de GameManager existe, si no, dará error de referencia.

(SCRIPT COLLISIONMANAGER) // añadir moneda
if(collider.blablabla != null) { GameManager.Instance.AddCoin() };
convendría hacer un if (GameManager.Instance != null) tmb para evitar errores pero vaya

Convenio de llaves:

Las llaves **siempre** así:

```
void papopepo()
{
    papopepo
}
```

nunca así

```
void papopepo() {
    papopepo
}
```

para métodos de una línea como los getters puede ser en la mísma línea tmb aunque es menos legible la mayoría de las veces.

```
void papopepo() { papopepo };
```

Órden de código: metidos en #region #endregion cada uno excepto los de tick

- enums para las state machines
- referencias variables que referencian a otros componentes u objetos
- properties variables internas
- métodos propios que hagamos nosotros
- métodos internos (tick): van en este orden
    - awake
    - start
    - fixedupdate
    - update
    - lateupdate

Órden de carpetas del proyecto: STC pero más o menos el planteamiento general

- Assets:
    - Scripts
    - Scenes
    - Sprites
        - UI
        - Characters
            - …
    - Prefabs
        - …

- Animations (Cuando animaciones)
- Sounds (Cuando sonidos)
  - Background (?)
  - SFX (?)
- … (Según lo que necesitemos, si tenemos varias recursos de un mismo tipo se creará su propia carpeta (fonts por ejemplo si usamos varias), pero por lo general cuanto más simple sea mejor)

Y en general algo **muy útil** es limpiar el código: hace una limpieza más estética y legible que otra cosa, pero siempre viene bien. (ej: (i+4)%2 pasa a ser (i + 4) % 2