# Prueba - 10Leaf

Tiempo: 2 días.
Modalidad: En equipo.

Para hacer la prueba es necesario hacer un fork de este repositorio y crear una nueva rama “nombre1-nombre2-implementation”. El proyecto contiene todos los assets necesarios para desarrollar lo que se pide en la consigna.

Para hacer la entrega, se debe crear un Pull Request desde el fork hacia la rama “implementation” del repositorio original.

Se deben respetar los estándares de programación mencionados al final del documento

## Consigna

En la escena **main_scene.tcsn** se encuentran instancias de **player.tcsn**, **enemy.tcsn** y **flag.tcsn**. Se pide implementar lo siguiente:

### Parte 1:

- **Movimiento del jugador:** puede moverse en las 4 direcciones con w-a-s-d.
- **Ataque del jugador:** Dispara proyectiles que infligen a priori 1 de daño, pero puede que a futuro se quiera probar daños diferentes sin tener que modificar el código.
- **3 enemigos:** Todos deben ser instancias de la escena enemy.tcsn
- **Vida de enemigos:** Deben tener una cantidad de vida ajustable desde el inspector y esta se debe ver reducida cuando un proyectil los alcanza. Por defecto la vida de los enemigos es de 3 puntos.
- **Vida del jugador:** Aunque no se pide implementar nada que dañe al jugador, se pide que también lleve un registro de sus puntos de vida.
- **Muerte:** Cuando los puntos de vida de una entidad llegan a 0, esta debe desaparecer.

*Nota: No es necesario programar IA ni movimiento de los enemigos. Con que sean como “dummies” está bien.*

### Parte 2:

- **Colores:** Asignarle un color distinto a cada enemigo (rojo, azul, amarillo).
- **Flag:** Cuando un enemigo es golpeado por un proyectil, el objeto Flag debe cambiar su sprite, mostrando el sprite que representa la mezcla de los colores de los últimos 2 enemigos golpeados. Empieza en blanco.

*Ejemplo:*
*Si se golpean en este orden: rojo → azul → amarillo → amarillo.*
*Cambia así: blanco → rojo → violeta → verde → amarillo.*

*Los sprites se pueden encontrar en **resources/sprites/flags.***

# Estándares de programación que seguimos:

## Manejo del repositorio:

Es importante trabajar siguiendo **Git Flow** para evitar la mayor cantidad de conflictos posibles. Creando una rama para cada feature. Los nombres de las ramas deben estar en inglés y cumplir snake_case. Los commits también deben ser escritos en inglés y contener una descripción de lo que se hizo.

## Nomenclatura:

- **Clases:** pascal case
- **Nodos:** pascal case
- **Variables:** snake case
- **Funciones:** snake case
- **Escenas:** snake case
- **Scripts:** snake case
- **Recursos:** snake case
- **Nodos globales:** snake case (todo en mayúscula)
- **Constantes:** snake case (todo en mayúscula)
- **Acciones del Input Map:** snake case

Los nombres de las variables y funciones deben ser lo más descriptivas posible. De esta manera, cualquiera puede leer el código y entender fácilmente lo que hace.