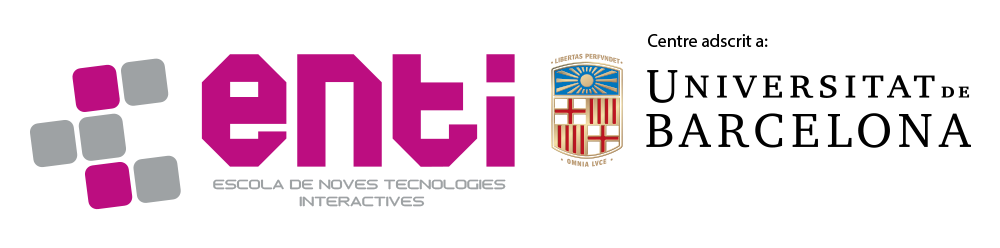
**Colorful War**

**Resumen del juego**

El juego se resume en que el jugador (la entidad azul) se dedica a intentar matar al rojo (con colisiones), mientras la entidad Verde intenta matarlo a él. Básicamente es un juego de 3 en el que uno tiene un objetivo y una amenaza Azul mata Rojo, Rojo mata Verde, Verde mata Azul.

**Steering Behaviors usados:**

* **Pathfinding:** Aunque no es un steering behavior estrictamente, al comenzar la partida tanto el agente verde como el rojo tienen un camino predefinido que repetirán hasta que se encuentren lo suficientemente cerca de su víctima. Si el jugador interactúa con el agente rojo, el agente rojo activará directamente su seek behavior e irá a por el agente verde.
* **Obstacle Avoidance & seek**: Cada uno de los agentes tiene un target diferente, y al activarse este behavior se activarán a su vez obstáculos que están a una mayor prioridad que el seek. En un principio pensé en optar por un wander en lugar de un pathfinding, pero el pathfinding me daba un resultado mucho más sólido y acabado.

**Clases afectadas y funciones:**

* **Clases.**
  + **EcosystemState**: Se encarga de manejar todo el funcionamiento del minijuego colorful war, toda la lógica. Aquí se encuentran la mayoría de variables y funciones necesarias para el funcionamiento del juego.
  + **Player**: Ha sido alterado para recibir nuevas características cómo la vida del jugador. Aquí se encuentran funciones getters y setters.
  + **Boid** : Ha sido alterada simplemente para incluir vida a los agentes, además se han alterado algunos behaviors para que hiciesen cosas extras, por ejemplo el Obstacle Avoidance en lugar de realizar un wander realizan un seek.
* **Funciones.**
  + **CreateBoid:** Función alterada para que el target del segundo agente fuese el primer agente, aparte se ha cambiado el path que seguía para que tuviese uno diferente al primer agente.
  + **IntroGame:** se encarga de comprobar si el jugador ha sido dañado y comprueba si el tiempo de inmunidad señalizado por la constante timeToWait se ha acabado para reiniciar la posibilidad de daño en el jugador. El nombre de la función no es nada intuitivo…
  + **CalculateCollisions:** Comprueba colisiones y aplica la fuerza del empujón que se basa en sumarle a la posición del herido el vector normalizado de la velocidad del agente que impacta por una magnitud en este caso 10. También se encarga de activar el tiempo de inmortalidad del jugador.
  + **seeWhoWins:** Se encarga de revisar si alguno de los jugadores esta con la vida a <= 0 para activar el bool.
  + **StateManager:** revisa si el agente se encuentra a menos de una distancia de la magnitud de 100 para activar el seek.

