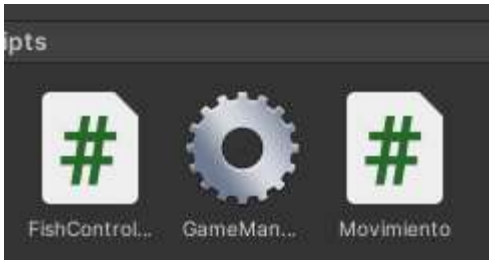
By: Adrián Ontiveros Cruz

PECERA STEERING BEHAVIOR

GENERAL

Para este proyecto he utilizado principalmente 3 scripts y 2 prefabs.
A parte de diferentes componentes en la escena, incluyendo la UI.



FUNCIONALIDAD DE LOS PECES

Los peces por defecto están constantemente haciendo wander y comprobando si van a chocar con muros mediante raycast.

```
// Fuerza de evitación de los walls
1 reference | Qodo Gen: Options | Test this method
private Vector2 CollisionAvoidance()
{
    Vector2 steeringForce = Vector2.zero;
    RaycastHit2D[] hit = Physics2D.RaycastAll(transform.position, velocity.normalized, raycastDetectionDistance);
    for (int i = 0; i < hit.Length; i++)
    {
        if (hit[i].collider.CompareTag("Wall"))
        {
            // Debug.Log("avoid Wall");


            // Punto de impacto del raycast
            Vector2 hitPoint = hit[i].point;

            // Calcular la dirección opuesta al punto de impacto
            Vector2 avoidanceDirection = ((Vector2)transform.position - hitPoint).normalized;

            // Cambiar la dirección del pez drásticamente en el lado contrario
            Vector2 desiredVelocity = avoidanceDirection * maxSpeed;

            // Forzar un giro completo hacia la dirección opuesta
            velocity = desiredVelocity;
            steeringForce = Vector2.ClampMagnitude(desiredVelocity - velocity, maxForce);

            break;
        }
    }

    return steeringForce;
}
```

By: Adrián Ontiveros Cruz

```
void Update()
{
    Vector2 avoidanceForce = CollisionAvoidance();
    // Debug.Log(avoidanceForce);

    var emission = particles.emission;
    // var main = particles.main;

    if (target != null && hunter == null)
    {
        // Si hay un objetivo, persigue el objetivo
        ApplySteering(Seek(target.position) + avoidanceForce);
        maxSpeed = 15;
    }
    else if (hunter != null)
    {
        target = null;
        // Si hay un cazador, huye del cazador
        ApplySteering(Flee(hunter.position) + avoidanceForce);
        maxSpeed = 20;

        // Cambia la cantidad de partículas emitidas en base a la speed del pez
        emission.rateOverTime = maxSpeed -5;
        // main.startSpeed = 8;
    }
    else
    {
        // Si no hay objetivo ni cazador, vagar
        Wander(avoidanceForce);

        // Reiniciar valores base
        emission.rateOverTime = isBigFish ? 10 : 8;
        maxSpeed = isBigFish ? 10 : 12;
        // main.startSpeed = isBigFish ? 1 : 1.5f;
    }

}
```

En caso de encontrarse con otro pez reacciona en función a si es más pequeño o más grande persiguiéndolo o escapando respectivamente.

Cuando se encuentra con el player siempre escapará.
Las detecciones son mediante triggers.

By: Adrián Ontiveros Cruz

```
0 references | Qodo Gen: Options | Test this method
private void OnTriggerEnter2D(Collider2D other)
{
    // Detectar otros peces y decidir si son targets o hunters
    if (other.CompareTag("Fish")) // podria hacerlo detectando su script/clase
    {
        FishController otherFish = other.GetComponent<FishController>();
        if (otherFish != null)
        {
            if (isBigFish && !otherFish.isBigFish)
            {
                target = other.transform;
            }
            else if (!isBigFish && otherFish.isBigFish)
            {
                hunter = other.transform;
            }
        }
    }

    if (other.CompareTag("Player")) hunter = other.transform; // el player siempre será hunter
}

0 references | Qodo Gen: Options | Test this method
private void OnTriggerExit2D(Collider2D other)
{
    // Limpiar objetivo/cazador al salir del rango
    if (other.CompareTag("Fish"))
    {
        FishController otherFish = other.GetComponent<FishController>();
        if (otherFish.isBigFish)
        {
            hunter = null;
        }
        else if (!otherFish.isBigFish)
        {
            target = null;
        }
    }

    if (other.CompareTag("Player"))
    {
        hunter = null;
    }
}
```

Al dejar de estar en el rango de persecución (se ha/ se le ha escapado), reiniciamos el target/ hunter, para que siga haciendo wander.

By: Adrián Ontiveros Cruz

INTERACCIÓN GAMEPLAY / UI

Mediante la UI, se pueden añadir o quitar peces grandes y pequeños, además se puede ver un contador para saber cuántos hay de cada.

A través de las funciones del GameManager gestionamos el spawneo/despawneo de peces. Estas funciones se llaman directamente desde los botones de la UI.

```
// Instancia un pez (big o small) y actualiza los contadores
0 references | Qodo Gen: Options | Test this method
public void InstanceFish(bool isBig){
    GameObject newFish;
    if(isBig){
        newFish = Instantiate(bigFish, bigFishSpawn.position, bigFishSpawn.rotation);
        newFish.transform.SetParent(bigFishSpawn);
        bigFishesCount++;
    }
    else{
        newFish = Instantiate(smallFish, smallFishSpawn.position, smallFishSpawn.rotation);
        newFish.transform.SetParent(smallFishSpawn);
        smallFishesCount++;
    }

    smallFishCounter.text = smallFishesCount.ToString();
    bigFishCounter.text = bigFishesCount.ToString();
}

// Elimina un pez (big o small) y actualiza los contadores
0 references | Qodo Gen: Options | Test this method
public void RemoveFish(bool isBig){
    GameObject toDelete = null;
    if(isBig && bigFishSpawn.childCount > 0){
        toDelete = bigFishSpawn.GetChild(0).gameObject;
        bigFishesCount--;
    }
    else if (!isBig && smallFishSpawn.childCount > 0){
        toDelete = smallFishSpawn.GetChild(0).gameObject;
        smallFishesCount--;
    }
    Destroy(toDelete);
    smallFishCounter.text = smallFishesCount.ToString();
    bigFishCounter.text = bigFishesCount.ToString();
}
```

By: Adrián Ontiveros Cruz

ASPECTO VISUAL

El juego tiene sprites para los “peces”, un bg, la “pecera” que es una TV antigua y efectos de particle system que se modifican en función a la speed de los “peces”. Speed que cambia en función a si son small, big o si están persiguiendo/siendo perseguidos.

Por ejemplo:

```
else if (hunter != null)
{
    target = null;
    // Si hay un cazador, huye del cazador
    ApplySteering(Flee(hunter.position) + avoidanceForce);
    maxSpeed = 20;

    // Cambia la cantidad de partículas emitidas en base a la speed del pez
    emission.rateOverTime = maxSpeed -5;
    // main.startSpeed = 8;
}
```

By: Adrián Ontiveros Cruz

EXTRA

He ampliado una función del juego, en vez de que eviten el ratón, directamente puedes interactuar como un player dentro de la pecera, el cual puedes mover mediante los comandos típicos de movimiento WASD o flechas.

```
0 references | Qodo Gen: Options | Test this class
public class Movimiento : MonoBehaviour
{
    1 reference
    public float velocidad = 5f;

    0 references | Qodo Gen: Options | Test this method
    void Update()
    {
        float movimientoHorizontal = Input.GetAxis("Horizontal"); // A y D
        float movimientoVertical = Input.GetAxis("Vertical"); // W y S

        Vector3 movimiento = new Vector3(movimientoHorizontal, movimientoVertical, 0) * velocidad * Time.deltaTime;

        transform.Translate(movimiento);
    }
}
```

Link a GitHub

https://github.com/McWally/SteeringBehavior