Walkers Knowledge

Pablo Amorín Valle.
Gabriel Ballesteros Melero.
Daniel Coronado Martín.
Carlos Gregorio Ruiz.

INDEX

01 Player Knowledge

02 Minion Knowledge

01

Player knowledge

Objectives

- Pick as much pick-ups as you can

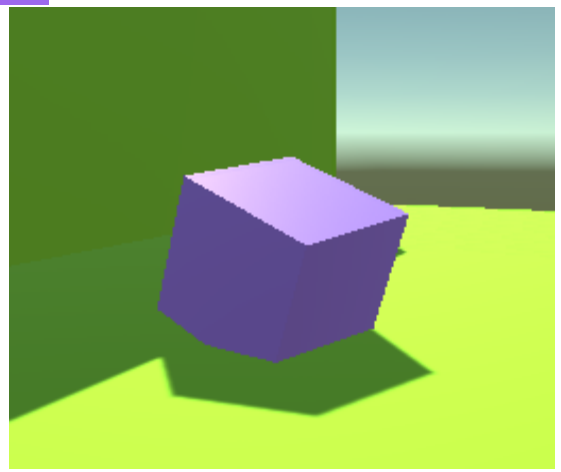
- Don’t collide with minions while going for the pick-ups

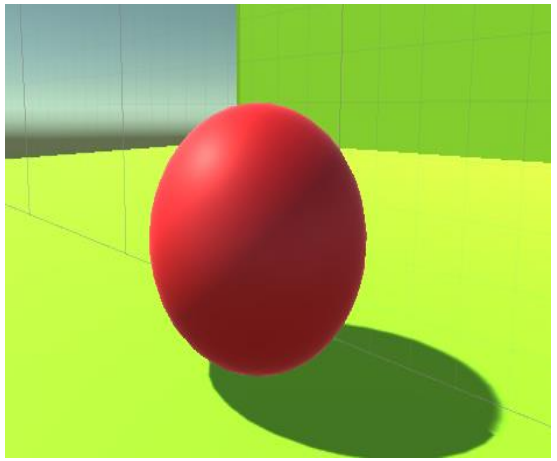
Dependencies

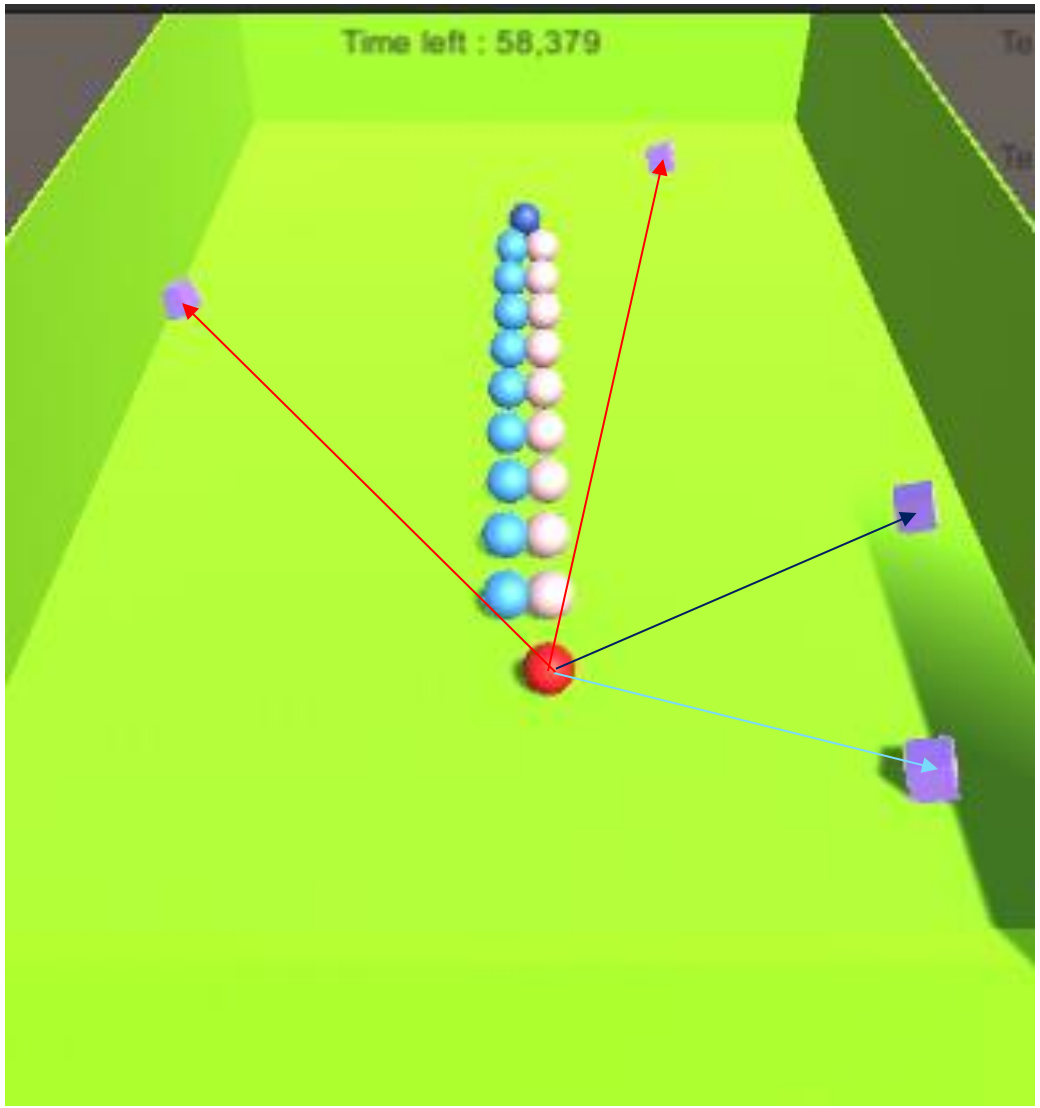
- Pick as much pick-ups as you can
  - Position of the pick-ups(V3)
  - Position of the player(V3)
  - Debug line to the pick-up

- Don't collide with minions while going for the pick ups
  - Position of the player(V3)
  - Position of the minions(V3)
  - Debug line to the pick-up and collisions with the minions

Considering Profit position, we select the one with no minions in the middle which is the closest one to the player subtracting the distance of the player and the profit.







```
void CalculaMinimo()
{
    //metemos a el player a la layer de ignore raycast para que no se detecte a si misma en las colisiones
    int LayerIgnoreRaycast = LayerMask.NameToLayer("Ignore Raycast");
    Pepe.layer = LayerIgnoreRaycast;
    Debug.Log("Current layer: " + gameObject.layer);
    //iniciamos el closest pick up infinito para ir comparando cual esta mas cerca
    //tambien inicializamos un array con todos los profits que se pueden coger
    float distanceToClosestPickUp = Mathf.Infinity;
    float distanceToPickup;
    GameObject closestProfit = null;
    GameObject[] alltargets = GameObject.FindGameObjectsWithTag("Profit");
    //recorremos el array de proffits
    foreach (GameObject currentProfit in alltargets)
    {
        //metemos al proffit en la layer que no detecta colisiones para luego
        currentProfit.layer = LayerIgnoreRaycast;
        if (currentProfit.activeInHierarchy)
        {
            //hacemos la comparacion para poder comparar cual esta mas cerca
            distanceToPickup = (currentProfit.transform.position - this.transform.position).sqrMagnitude;
            //si el que esta mas cerca notiene ningun minion en medio comprobando las colisiones con las layers lo guardamos como objetivo
            if (distanceToPickup < distanceToClosestPickUp && !Physics.Linecast(this.transform.position, currentProfit.transform.position))
            {
                distanceToClosestPickUp = distanceToPickup;
                closestProfit = currentProfit;
            }
        }
    }
    // si ningun target esta libre de minions por el medio vamos al primer target del array de profits para que no se quede quieto
    if((alltargets[0].transform.position - this.transform.position).sqrMagnitude== Mathf.Infinity)
    {
        distanceToPickup = (alltargets[0].transform.position - this.transform.position).sqrMagnitude;
        distanceToClosestPickUp = distanceToPickup;
        closestProfit = alltargets[0];
    }

    Debug.DrawLine(this.transform.position, closestProfit.transform.position);
    playersMovUnits = playersMovUnits * ScGameGlobalData.maxPlayersMovUnits;
    GetComponent<ScPlayerControl>().moveOn((closestProfit.transform.position - this.transform.position), playersMovUnits);
}

// Fin de - public class ScPlayerAI_Near : MonoBehaviour {
```

02

Minion knowledge

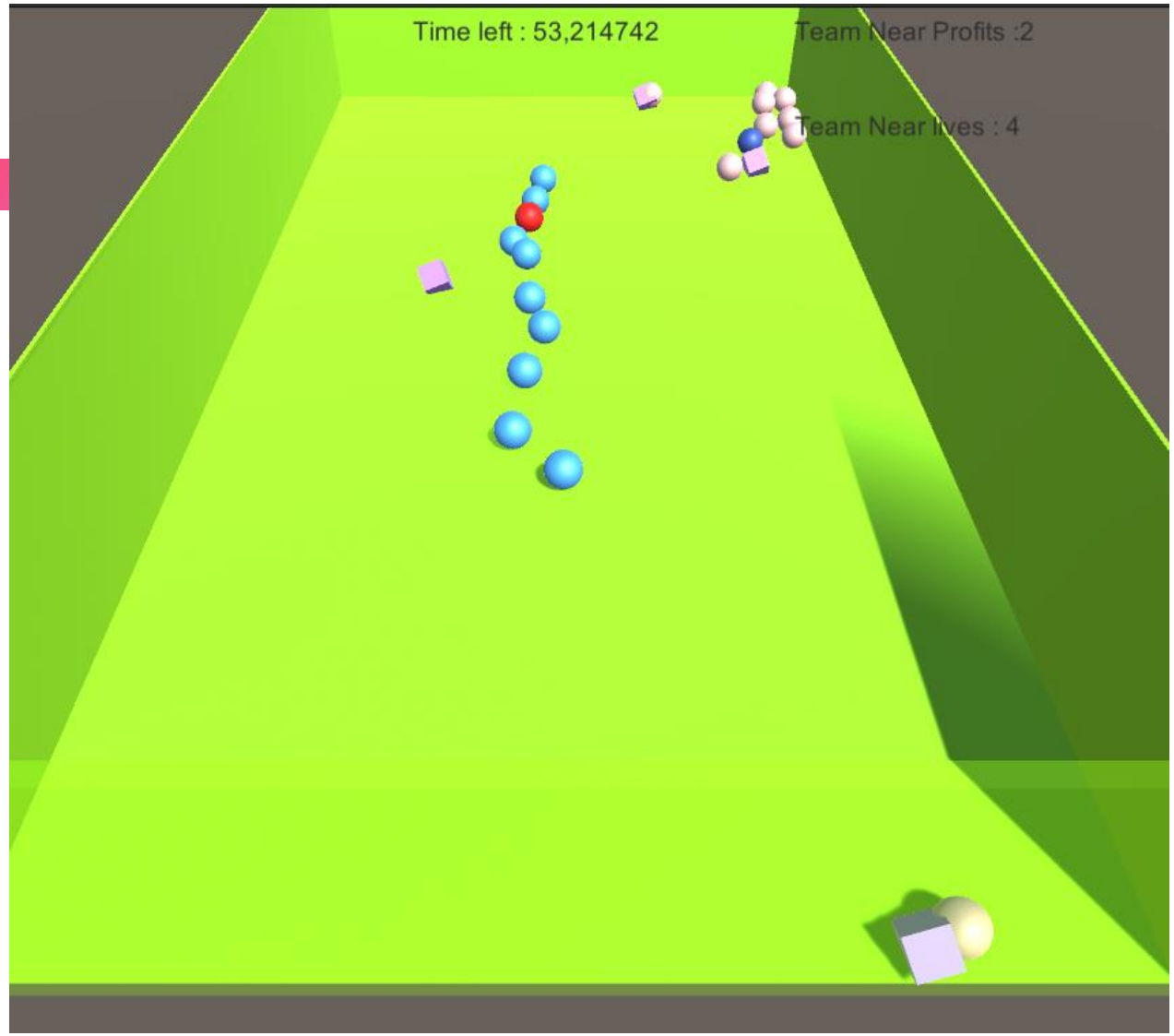
Objectives

- Defeat the opponent player.
- Defend a pickup to avoid the other team to chase it.

Dependencies

- Defeat the opponent player:
  - Where is the opponent player?

- Defend a pickup to avoid the other team to chase it:
  - Where are the pickups?
  - Is there a minion already defending it?



The minion will move depending on a condition:

+ If he is really close to the player, he will start chasing it and try to defeat him.

+ If the player is far from the minion, the minion will play in a defensive way, defending a profit and staying there.

```
void FixedUpdate()
{

    // Si la distancia al Player es menor que 5, los Minions van a perseguir al Player
    if (Vector3.Distance(this.transform.position, Player.transform.position) < 0.5)
    {
        perseguirPlayer();
    }

    // En caso de que el Minion se encuentre más lejos,
    else
    {
        custodiarProfit();
    }

}// Fin de - void FixedUpdate()
```

```
void perseguirPlayer()
{
    // Se puede dibujar una línea para comprobar
    //Debug.DrawLine(this.transform.position, Player.transform.position);

    // Hacemos que los Minions miren hacia el Player
    var lookPos = Player.transform.position - this.transform.position;
    //lookPos.y = 0;

    // Otra forma de realizar el movimiento
    //var rotation = Quaternion.LookRotation(lookPos);
    //transform.rotation = Quaternion.RotateTowards(transform.rotation, rotation, 2);
    // Hacemos que, mirando hacia el Player, se mueva hacia delante a por él
    //transform.Translate(Vector3.forward * 3 * Time.deltaTime);

    // Guarda el vector hacia donde tiene que ir y lo multiplica hacia la velocidad que puede ir el Minion
    movement = lookPos;
    minionsMovUnits = minionsMovUnits * ScGameGlobalData.maxMinionsMovUnits;
    GetComponent<ScMinionControl>().moveOn(movement, minionsMovUnits);
}
```

```
void custodiarProfit(){

    float distanceToClosestPickUp = Mathf.Infinity;
    GameObject closestPickup = null;
    GameObject[] alltargets = GameObject.FindGameObjectsWithTag("Profit");
    GameObject[] allminions = GameObject.FindGameObjectsWithTag("Minion");
    bool near = true;

    //Calcula el Profit mas cercano y compruba si hay algun minion cerca de ese Profit
    foreach (GameObject currentPickup in alltargets)
    {
        if (currentPickup.activeInHierarchy)
        {
            fl
            if (closestPickup == null){
                //Perseguir player
                perseguirPlayer();
            }else{
                //Custiodar Profit
                var lookPos = closestPickup.transform.position - this.transform.position;
                //lookPos.y = 0;

                movement = lookPos;
                minionsMovUnits = minionsMovUnits * ScGameGlobalData.maxMinionsMovUnits;
                GetComponent<ScMinion
            }
        }
    }
}// Fin
```

(Campo) float ScMinionAI_Near.minionsMovUnits