FCTUC FACULDADE DE CIÊNCIAS E TECNOLOGIA

Relatório da Meta 1 do Trabalho Prático Nº1

PL3 / Prof. João Correia
Dário Félix, Nº 2018275530, dario@student.dei.uc.pt
João Calhau, Nº 2016255704, uc2016255704@student.uc.pt
Tatiana Simões, Nº 2018285812, 2018285812@student.uc.pt
Coimbra, 7 de março de 2021

❖ Problema

Expandir o código fornecido de forma a suportar o sensor de obstáculos.

❖ Código Alterado

```
// Update is called once per frame
Unity Message | 0 references
void FixedUpdate()
{
    ObjectInfo anObject;
    anObject = GetClosestObstacle();
    if (anObject != null)
    {
        angleToClosestObj = anObject.angle;
        strength = 1.0f / (anObject.distance + 1.0f);
    }
    else
    { // no object detected
        strength = 0;
        angleToClosestObj = 0;
    }
}
```

Fig. 1 – BlockDetectorScript.cs: FixedUpdate()

```
1 reference
public ObjectInfo GetClosestObstacle()
{
    ObjectInfo[] a = (ObjectInfo[])GetVisibleObstacle("Wall").ToArray();
    if (a.Length == 0)
    {
        return null;
    }
    return a[a.Length - 1];
}

1 reference
public List<ObjectInfo> GetVisibleObstacle(string objectTag)
{
    RaycastHit hit;
    List<ObjectInfo> objectsInformation = new List<ObjectInfo>();

    for (int i = 0; i * angleOfSensors < 360f; i++)
    {
        if (Physics.Raycast(this.transform.position, Quaternion.AngleAxis(-angleOfSensors * i, initialTransformUp) * initialTransformFwd, out hit, rangeOfSensors))
        {

            if (hit.transform.gameObject.CompareTag(objectTag))
            {
                if (debugMode)
                {
                    Debug.DrawRay(this.transform.position, Quaternion.AngleAxis((-angleOfSensors * i), initialTransformUp) * initialTransformFwd * hit.distance, Color.green);
                }
                ObjectInfo info = new ObjectInfo(hit.distance, angleOfSensors * i + 90);
                objectsInformation.Add(info);
            }
        }
    }

    objectsInformation.Sort();

    return objectsInformation;
}
```

Fig. 2 – BlockDetectorScript.cs: GetClosestObstacle() e GetVisibleObstacle()

```
Unity Message | 0 references
void Update()
{

    // get sensor data
    resouceAngle = resourcesDetector.GetAngleToClosestResource();

    resourceValue = weightResource * resourcesDetector.GetLinearOuput();

    obstacleAngle = blockDetector.GetAngleToClosestObstacle();

    obstacleValue = weightWall * blockDetector.GetLinearOuput();

    // apply to the ball
    applyForce(resouceAngle, resourceValue); // go towards
    applyForce(obstacleAngle, -1.0f * obstacleValue); // go the opposite direction of a wall


}
```

Fig. 3 – LinearRobotUnitBehaviour.cs: Update()

Para suportar o sensor de obstáculos baseou-se substancialmente nos métodos do *ResourceDetectorScript.cs* para desenvolver os métodos do *BlockDetectorScript.cs (fig.1 e fig. 2)*. Por fim, no *LinearRobotUnitBehaviour.cs*, aplicam-se agora duas forças ao invés de uma, com a particularidade da nova força ter o valor de *obstacleValue* a ser multiplicado por -1 para provocar um deslocamento na direção oposta aos obstáculos *(fig. 3)*.

❖ Conclusões

Verificamos que a velocidade do *robô D31* tinha efeito sobre a eficácia dos sensores (em velocidades mais altas o robô acabava por bater contra os obstáculos e o movimento também não era tão coordenado aquando na tentativa de aproximação aos recursos).

Os objetivos foram alcançados sem dificuldades.