Brief:
Hit a Moving Target
A 2D top down game with enemy AI needs a targeting system to shoot at the player. The shots are not instant so the system will need to aim ahead of the player's current position. Shots aimed at the player should hit if the player is not moving, or if the player is moving at a constant speed. To avoid hits, the player must need to change while the shot is travelling. As input to the system you will be given the player's relative location and velocity. You must return the relative clockwise angle in degrees needed to hit the player.

Step 1:
Creating the assets:
You need to create 3 2D objects. The player, the shooter, the bullet.

Step 2:
Adding materials:
In order to have a clear view of what object is what, add 3 different colours to them. To do this, select Assets, Material, select which colour you prefer and drag this onto the object.

Step 3:
Creating the scripts. We have to create 3 scripts, so let's go through them one at a time, starting with the player:

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;


public class Pla : MonoBehaviour {


    [SerializeField]

    float moveSpeed = 5f; //what speed to move at


    float dirX , dirY; //which direction

    Rigidbody2D rb; //refering to the rigidbody attached to the object
```

```
    // Use this for initialization
    void Start () {
        rb = GetComponent<Rigidbody2D> ();
    }

    // Update is called once per frame
    void Update () {
        dirX = Input.GetAxis ("Horizontal") * moveSpeed; //as unitys go to controls are the key arrows, we do not have to set these, but this part of code shows which direction to travel in and when.
        dirY = Input.GetAxis ("Vertical") * moveSpeed;
    }

    void FixedUpdate()
    {
        rb.velocity = new Vector2 (dirX, dirY);
    }
}
```

We also need to add a tag to the player, add a tag named “Pla”, you can name the tag whatever you prefer but ensure you change this in the next script.

Bullet:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

```
public class Bullet : MonoBehaviour {

    float moveSpeed = 7f; //what speed the bullet moves at

    Rigidbody2D rb; //refering to the rigidbody attatched to the player

    Pla target;
    Vector2 moveDirection; //what direction the bullet will travel in

    // Use this for initialization
    void Start () {
        rb = GetComponent<Rigidbody2D> ();
        target = GameObject.FindObjectOfType<Pla>(); //what object are you aiming the
bullets towards
        moveDirection = (target.transform.position - transform.position).normalized *
moveSpeed; // move towards the targets location at the given speed
        rb.velocity = new Vector2 (moveDirection.x, moveDirection.y); //the new direction in
which the bullet will move

    }

    void OnTriggerEnter2D (Collider2D other){


        if(other.gameObject.tag.Equals ("Pla"))
            {

                Destroy(other.gameObject); //destroy player
            }
        }
    }
```

Enemy:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Enemy : MonoBehaviour {
```

```
    [SerializeField]
    GameObject Bullet; // what object will be the bullet, this iwll be se tin the inspector.

    float fireRate; //how often it fires
    float nextFire; // how much time between the fires

    // Use this for initialization
    void Start () {
        fireRate = 1f;
        nextFire = Time.time;
    }

    // Update is called once per frame
    void Update () {
        CheckIfTimeToFire ();
    }

    void CheckIfTimeToFire()
    {
        if (Time.time > nextFire) {
            Instantiate (Bullet, transform.position, Quaternion.identity);
            nextFire = Time.time + fireRate; //check if its enough time to fire
        }

    }

}
```

Step 4:
We now need to set up the inspectors to ensure our code will work. In the player's inspector, change the Move speed to your desired speed.
We need to make the bullet a Prefab, as it will not start on screen, but appear once the game is running. In order to do this, simply go onto assets, create prefab and drag the Bullet object we have been previously using onto the prefab. It is now a prefab. Ensure it is renamed Bullet.
For the enemy inspector, ensure you have dragged the bullet prefab into the Bullet option.

Date	Estimate Start	End	Interruptions	Task
21/03/2019	2.17pm	2.30pm	2 minutes	Creaitng assets

21/03/2019	2.32pm	2.38pm	0	Adding Materials
21/03/2019	3.00pm	4.32pm	5 minutes	Writing the code
21/03/2019	4.44pm	4.50pm	0	Setting up inspector