Unity Hands On

Line of Sight AI

Step 1: Download *Chapter Five/ChasingAI.zip* from the website. Open the chasingAI scene. In it, you will find a large room with a wizard.

Step 2: Open LineOfSight.cs with the script editor and add the code shown in Listing 5.1.

Listing 5.1 Basic line of sight script

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

public class LineOfSight : MonoBehaviour {

     public Transform target; //the enemy
     public Transform head;
     float seeRange = 12.0f; //maximum attack distance -
     //will attack if closer than
     //this to the enemy
     float shootRange = 8.0f;
     float keepDistance = 2.0f; //closest distance to get
     //to enemy
     float rotationSpeed = 4.0f;
     float speed = 0.1f;
     ParticleSystem magicParticles;

     Animator anim;
     GameObject magic;

     void Start()
     {
         anim = gameObject.GetComponent<Animator>();
         magic = GameObject.Find("Magic");
         magicParticles = magic.
             GetComponent<ParticleSystem>();
         magicParticles.Stop();
     }

     void TurnOnMagic()
     {
         magicParticles.Play();
     }

     void TurnOffMagic()
     {
         magicParticles.Stop();
     }
```

```
void Update ()
{
    if(anim.GetCurrentAnimatorClipInfo(0)[0].clip.
        name == "Standing_Walk_Forward")
    speed = 0.1f;
    else
    speed = 0.0f;

    if (CanSeeTarget ())
    {
        if(CanShoot())
        {
            anim.SetBool("isWalking",false);
            anim.SetBool("isAttacking",true);
            Shoot();
        }
        else
        {
            anim.SetBool("isWalking",true);
            anim.SetBool("isAttacking",false);
            Pursue();
        }
    }
    else
    {
        anim.SetBool("isWalking",false);
        anim.SetBool("isAttacking",false);
        //stand around
    }
}

bool CanSeeTarget ()
{
if (Vector3.Distance(head.position, target.position)
    > seeRange)
    return false;

return true;
}

bool CanShoot()
{
    if (Vector3.Distance(head.position, target.
        position) > shootRange)
    return false;

    return true;
}
```

```
    void Pursue()
    {
        Vector3 position = target.position;
        Vector3 direction = position - head.position;
        direction.y = 0;
            // Rotate towards the target
        head.rotation = Quaternion.Slerp (head.rotation,
            Quaternion.LookRotation(direction),
            rotationSpeed * Time.deltaTime);
        transform.eulerAngles = new Vector3(0,transform.
            eulerAngles.y, 0);
            // Move the character
        if(direction.magnitude > keepDistance)
        {
            direction = direction.normalized * speed;
            transform.position += direction;
        }
    }

    void Shoot()
    {
        Vector3 position = target.position;
        Vector3 direction = position - head.position;
        direction.y = 0;
        // Rotate towards the target
        transform.rotation = Quaternion.Slerp(transform.
            rotation, Quaternion.LookRotation(direction),
            rotationSpeed * Time.deltaTime);
        transform.eulerAngles = new Vector3(0, transform.
            eulerAngles.y, 0);
    }
}
```

Step 3: Locate the LineOfSight script in the Inspector attached to the Gatrillian object in the Hierarchy. Drag and drop the RigidBody FPS Controller from the Hierarchy onto the exposed *Target* variable. This sets the object that the NPC will consider its enemy.
Step 4: Drag and drop the Gatrillian > Head object in the Hierarchy onto the exposed *Head* variable. You will find the Head object under Gatrillian > mixamorig:Hips > mixamorig:Spine > :Spine1 > :Spine2 > :Neck. This is used to determine the forward-facing direction of the wizard.
Step 5: Play. The CanSeeTarget() and CanShoot() functions use the variables seeRange and shootRange, respectively, to determine when the player is close enough to pursue and close enough to shoot. If you run away from the NPC backward as fast as possible, you will notice the point at which it gives up chasing you.

Step 6: The current code does not take into consideration the direction that the NPC is facing. If you approach it from behind, it will still sense your presence when you are close enough. For it to use a field of vision angle, modify the AI script as shown in Listing 5.2.

Listing 5.2 Restricting the field of vision of an NPC to an area in front of it

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

public class LineOfSight : MonoBehaviour {

    public Transform target; //the enemy
    public Transform head;
    float seeRange = 12.0f; //maximum attack distance -
    //will attack if closer than
    //this to the enemy
    float shootRange = 8.0f;
    float keepDistance = 2.0f; //closest distance to get to
    //enemy
    float rotationSpeed = 4.0f;
    float sightAngle = 60f;
    float speed = 0.1f;
    Animator anim;
    GameObject magic;
    ParticleSystem magicParticles;

    void Start()
    ...

    bool CanSeeTarget ()
    {
        Vector3 directionToTarget = target.position –
            head.position;
        float angle = Vector3.Angle(directionToTarget,
            head.forward);

        if (Vector3.Distance(head.position, target.
            position) > seeRange || angle > sightAngle)
            return false;

        return true;
    }
    ...
```

Step 7: Play. You will now be able to sneak up behind the NPC without it noticing you.