// EnemyAI.cs

```
using UnityEngine;
using UnityEngine.AI;
using System.Collections;

enum State
{
    // 追いかけ中
    Chasing,
    // 最後に見た場所に移動中
    GoToLastLookPosition,
    // 目的地に移動中
    TurnWalk,
}

public class EnemyAI : MonoBehaviour
{
    [SerializeField]
    private GameObject eye;
    private GameObject player;
    private Transform playerLookPoint;
    private NavMeshAgent agent;
    private State state = State.TurnWalk;
    private Player playerComponent;
    private Animator anim;

    private bool walk = false;
    private bool run = false;
    private bool push = false;
    private bool torituki = false;

    // Use this for initialization
    void Start ()
    {
        player = GameObject.FindGameObjectWithTag("Player");
        playerLookPoint = player.transform.Find("LookPoint");
        agent = GetComponent<NavMeshAgent>();
        playerComponent = player.GetComponent<Player>();
        anim = GetComponent<Animator>();

        state = State.TurnWalk;
        agent.SetDestination(GetNextTargetPosition());
    }

    // Update is called once per frame
    void Update ()
    {
        //Debug.Log(agent.name + ":" + agent.tag);
        if (state == State.Chasing)
        {
            if (CanLookPlayer())
            {
                // 見えている→プレイヤーの位置に向かって進む
                agent.SetDestination(player.transform.position);
                anim.Play("run_27fps");
            }
            else
            {
                // 見失った→最後に見た場所まで移動
                state = State.GoToLastLookPosition;
            }
        }
        else if (state == State.GoToLastLookPosition || state == State.TurnWalk)
        {
            if (CanLookPlayer())
            {
                agent.SetDestination(player.transform.position);

                state = State.Chasing;
            }
            else if (HasArrived())
            {
                // 順番に目的地を選定し、agentに設定する
                agent.SetDestination(GetNextTargetPosition());
                state = State.TurnWalk;
            }
        }
    }

    // プレイヤーが見えるか？
    private bool CanLookPlayer()
    {
        // プレイヤーが遠すぎる場合は、見えないから終了
    if (!IsPlayerInRange())
    {
        return false;
    }

    // 視野角の範囲内にプレイヤーがいるか？
    if (!IsPlayerInFieldOfView())
    {
        return false;
    }

    // プレイヤーが半透明で見えない
    if (playerComponent.PlayerState == PlayerState.InShadow)
    {
        return false;
    }

    // Raycastを使って、障害物に遮られていないかチェック
    RaycastHit hitInfo;
    bool hit = Physics.Raycast(
        eye.transform.position,
        playerLookPoint.position - eye.transform.position,
        out hitInfo, 20);

    Debug.DrawLine(transform.position, player.transform.position, Color.green);

    if (hit && hitInfo.collider.tag == "Player")
    {
        return true;
    }

    return false;
}

public float viewAngle;

/// <summary>
/// プレイヤーが視野角の範囲内にいるかを返却する。
/// 壁の向こうとかは考慮しない
/// </summary>
/// <returns></returns>
private bool IsPlayerInFieldOfView()
```

```
{
    if (state == State.Chasing)
    {
        return true;
    }

    // プレイヤーが前方にいるか？（後方の場合は見えない）
    Vector3 finding = playerLookPoint.transform.position - eye.transform.position;

    return (Vector3.Angle(finding, eye.transform.forward) < viewAngle);
}

public float searchRange;

/// <summary>
/// プレイヤーが探索範囲（距離）内にいるか
/// </summary>
/// <returns></returns>
private bool IsPlayerInRange()
{
    float distance = (playerLookPoint.transform.position - eye.transform.position).magnitude;

    return (distance <= searchRange);
}

// NavMeshAgentの目的地に到着しているか？
private bool HasArrived()
{
    return (agent.remainingDistance < 0.05);
}

public GameObject[] m_TargetPositions;

/// <summary>
/// 現在選択中のTargetPosition
/// </summary>
private int m_NextTargetPositionIndex;

/// <summary>
/// 次に行くべき場所を返却する
/// </summary>
/// <returns></returns>
```

```
    private Vector3 GetNextTargetPosition()
    {
        Vector3 nextTargetPosition = m_TargetPositions[m_NextTargetPositionIndex].transform.position;

        m_NextTargetPositionIndex++;

        if (m_NextTargetPositionIndex >= m_TargetPositions.Length)
        {
            m_NextTargetPositionIndex = 0;
        }

        return nextTargetPosition;
    }
}
```