怪物巡逻状态机

1 人物距离大于10米 每3秒巡逻一次

2 人物距离小于10米 向人物移动

3 人物距离小于2.5米 向人物攻击

怪物状态基类

```
public class EnemyStateBase : MonoBehaviour {

    protected Animator animator;
    protected string animName;
    protected EnemyStateManager manager;
    protected Transform playerTrans;
    public virtual void OnInit()
    {
        animator = GetComponent<Animator>();
        manager = GetComponent<EnemyStateManager>();
        playerTrans = GameObject.FindGameObjectWithTag("Player").transform;
    }
    public virtual void OnEnter() { }
    public virtual void OnExcute() { }
}
```

怪物站立状态

```
public class EnemyStateIdle : EnemyStateBase {

    float time;

    public override void OnInit()
    {
        base.OnInit();
        animName = "Idle";
    }

    public override void OnEnter()
    {
        time = 0;
        animator.SetInteger("State", 0);
    }
    public override void OnExcute()
    {
        if (!animator.GetCurrentAnimatorStateInfo(0).IsName(animName))
            return;

        time += Time.deltaTime;

        if (Vector3.Distance(transform.position, playerTrans.position) < 2.5f)
        {
            manager.ChangeState<EnemyStateAttack>();
            return;
        }

        if (time >= 3.0f||Vector3.Distance(transform.position,playerTrans.position)<10f)
        {
            manager.ChangeState<EnemyStateRun>();
        }
    }
}
```

怪物移动类

```
public class EnemyStateRun : EnemyStateBase {

    CharacterController cc;
    Vector3 centerPoint;
    Vector3 target=Vector3.zero;
    bool moveToPlayer;

    public override void OnInit()
    {
        base.OnInit();
        cc = GetComponent<CharacterController>();
        animName = "pao";
        centerPoint = transform.position;
    }
```

```
public override void OnEnter()
{
    animator.SetInteger("State", 1);

    if (Vector3.Distance(playerTrans.position, transform.position) < 10f)
    {
        target.Set(playerTrans.position.x, transform.position.y, playerTrans.position.z);
        moveToPlayer = true;
    }
    else {
        target.Set(centerPoint.x + Random.Range(-10, 11),
            transform.position.y,centerPoint.z + Random.Range(-10, 11));
        moveToPlayer = false;
    }
}

public override void OnExcute()
{
    if (!animator.GetCurrentAnimatorStateInfo(0).IsName(animName))
        return;

    if (moveToPlayer)
        target.Set(playerTrans.position.x,
                        transform.position.y,
                        playerTrans.position.z);

     transform.LookAt(target);
     cc.SimpleMove(transform.forward * 2.0f);

    float distace = Vector3.Distance(transform.position, target);
```

```
        if (moveToPlayer)
        {
            if (distace < 2.5f || distace > 10f)
                manager.ChangeState<EnemyStateIdle>();
        }
        else
        {
            if (distace < 1.0f)
                manager.ChangeState<EnemyStateIdle>();
            else if (distacePlayer<10f)
                moveToPlayer = true;
        }
    }
}
```

怪物攻击类

```
public class EnemyStateAttack : EnemyStateBase {

    public override void OnInit()
    {
        base.OnInit();
        animName = "attack";
    }

    public override void OnEnter()
    {
        transform.LookAt(playerTrans);
        animator.SetInteger("State", 2);
    }
    public override void OnExcute()
    {
        if (!animator.GetCurrentAnimatorStateInfo(0).IsName(animName))
            return;

        if (animator.GetCurrentAnimatorStateInfo(0).normalizedTime > 0.8f)
            manager.ChangeState<EnemyStateIdle>();
    }
}
```

怪物状态管理类

```
public class EnemyStateManager : MonoBehaviour {

    private EnemyStateBase currentState;
    Dictionary<System.Type, EnemyStateBase> stateDic;
    void Awake () {
        stateDic = new Dictionary<System.Type, EnemyStateBase>();
        AddState<EnemyStateIdle>();
        AddState<EnemyStateRun>();
        AddState<EnemyStateAttack>();

        ChangeState<EnemyStateIdle>();
    }

    void AddState<T>() where T : EnemyStateBase
    {
        EnemyStateBase state= gameObject.AddComponent<T>();
        state.OnInit();
        stateDic.Add(state.GetType(), state);
    }

    public void ChangeState<T>() where T : EnemyStateBase
    {
        EnemyStateBase state = stateDic[typeof(T)];
        currentState = state;
        currentState.OnEnter();
    }
    void Update () {
        if (currentState != null)
            currentState.OnExcute();
    }
}
```