1、获取运动数据

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

/// <summary>

/// 运动体 ，职责：定义运动数据

/// </summary>

public class Vehicle : MonoBehaviour

{

    //操控对象容器

    [HideInInspector]

    public Steering[] steerings;

    //操控合力

    protected Vector3 steeringForce;

    //是否在平面

    public bool isplane = true;

    //质量

    public float mass = 1f;

    //加速度

    protected Vector3 acceleratedSpeed;

    //最高速度

    public float maxSpeed;

    //操控力的上线

    public float maxForce;

    [HideInInspector]

    public Vector3 velocity;

    //合力计算的间隔时间

    public float intervalComputerForce = 0.2f;

    public void Start()

    {

        //取得运动体上所有的操控对象

        steerings = GetComponents<Steering>();

        //按时间间隔计算操控合力

        InvokeRepeating("ComputerFinalSteeringForce", 0, intervalComputerForce);

    }

    //计算合力

    public void ComputerFinalSteeringForce()

    {

        steeringForce = Vector3.zero;

        //循环所有的操控组件，产生操控合力(每个单一操控的叠加)

        foreach (var item in steerings)

        {

            steeringForce += item.GetForce() \* item.weight;

        }

        //控制操控力的上限

        steeringForce=Vector3.ClampMagnitude(steeringForce, maxForce);

        if (steeringForce == Vector3.zero)

        {

            velocity = Vector3.zero;

        }

        //根据质量计算运动算需要的加速度

        acceleratedSpeed = steeringForce / mass;

    }

}

2、处理运动数据

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

/// <summary>

/// 运动控制类

/// </summary>

public class LocomtionController : Vehicle

{

    //转向速度

    public float rorationSpeed = 0.5f;

    //移动组件

    [HideInInspector]

    public CharacterController ch;

    //动画组件

    [HideInInspector]

    public Animator anim;

    //位移(操控力)

    public void Start()

    {

        base.Start();

        ch = GetComponent<CharacterController>();

        anim = GetComponent<Animator>();

    }

    public void Update()

    {

        Movement();

        Rotation();

        PlayAnim();

    }

    public void Movement()

    {

        //当前速度+加速度

        velocity += acceleratedSpeed \* Time.deltaTime;

        //当前速度不要超上限

        if (velocity.magnitude > maxSpeed)

        {

            velocity = velocity.normalized \* maxSpeed;

        }

        //是否是平面

        if (isplane) velocity.y = 0;

        //移动

        if (ch != null)

        {

            ch.SimpleMove(velocity);

        }

        else

        {

            transform.position += velocity \* Time.deltaTime;

        }

    }

     //动画

    public void PlayAnim()

    {

        if (anim != null)

        {

        }

    }

    //转向

    public void Rotation()

    {

        if (velocity != Vector3.zero)

        {

            Quaternion targetRoration = Quaternion.LookRotation(velocity, Vector3.up);

            transform.rotation = Quaternion.Lerp(transform.rotation, targetRoration, rorationSpeed);

        }

    }

}

3、运动物体基类

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

/// <summary>

/// 运动体

/// </summary>

public class Steering : MonoBehaviour

{

    //目标

    public Transform target;

    public Vector3? targetPosition = null;

    //希望速度

    protected Vector3 expectationVelocity;

    //最大速度(逃跑，徘徊的速度可能不同)

    public float maxSpeed;

    //运动体

    public Vehicle m\_vehicle;

    //权重

    public float weight=1;

    public void Start()

    {

        m\_vehicle = GetComponent<Vehicle>();

        if (m\_vehicle != null && maxSpeed == 0)

        {

            maxSpeed = m\_vehicle.maxSpeed;

        }

    }

    /// <summary>

    /// 实现具体操控的算法

    /// </summary>

    /// <returns></returns>

    public virtual Vector3  GetForce()

    {

        if (target != null)

        {

            targetPosition = target.position;

        }

         return Vector3.zero;

    }

}

4、运动物体

 using System.Collections;

using System.Collections.Generic;

using UnityEngine;

[RequireComponent(typeof(LocomtionController))]

public class SteeingForColliderObstacle : Steering

{

    //触角的长度

    public float maxSeeAhead = 5f;

    //障碍物所在的层

    public LayerMask mask;

    //推力的放大系数

    public float expandRate = 10f;

    //射线发射点

    public Transform sendPos;

    //碰撞物体的中心

    public Transform PusnPos;

    public void Start()

    {

        base.Start();

        if (m\_vehicle != null)

        {

            if (expandRate > m\_vehicle.maxForce)

            {

                expandRate = m\_vehicle.maxForce;

            }

        }

    }

    public override Vector3 GetForce()

    {

       base.GetForce();

       expectationVelocity = Vector3.zero;

        //检查障碍物(射线)

       RaycastHit hit;

       if (Physics.Raycast(sendPos.position, transform.forward,out hit, maxSeeAhead, mask))

       {

           //产生推力

           expectationVelocity = hit.point - PusnPos.position;

           //放大推力

           expectationVelocity \*= expandRate;

       }

        //形成实际的操控力(实际速度)

       return expectationVelocity;

    }

}