1. 简述NavMesh的设置方法。
2. 选择场景中需要生成寻路的几何体-可行走表面和障碍物。
3. 在NavMesh面板中选择需要烘焙寻路的物体，检测是否勾选Navigation Static.
4. 根据你的agent大小来调整bake 面板的设置。
5. Agent Radius : agent可以距离墙体 ，窗户或边缘多近的距离。
6. Agent Height : agent可以通过的最低的空间高度。
7. Max Slope : agent可以直接行走上去的最小坡度。
8. Step Height:  agent可以踩上（走上）的障碍物最高高度。
9. 点击bake按钮烘焙NavMesh。
10. 简述NavMeshAgent属性参数的使用方法。

Speed：物体寻路过程中的移动速度。

Angular Speed：物体旋转的角速度（物体在寻路过程中，Z轴会旋转至移动方向）。

Acceleration：加速度。

Stopping Distance：离终点停止的距离。

Auto Braking：自动刹车。

Radius：寻路碰撞体的半径。

Height：寻路碰撞体的高度。

Quality：躲避质量（越高则会躲避前方的寻路物体）。

AreaMask：寻路遮罩，可以指定该寻路走的路线。

1. 简述寻路过程中路网烘焙过程。

打开Navigation窗口准备烘焙地形

在inspector界面中将场景中的障碍物勾选为静态（navigation static）

在Navigation窗口中进行导航网格的烘焙Navigation中的主要参数。设置完成后，直接点击Bake场景会生成导航网格

生成导航网格后，在需要自带导航的物体上添加NavMeshAgent组件

1. 对于寻路过程中的障碍物绕行应该怎样处理？
2. 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

{

  [HideInInspector]

    public Steering[] steerings;

    protected Vector3 steeringForce;

    public bool isplane = true;

    public float mass = 1f;

    protected Vector3 acceleratedSpeed;

    public float maxSpeed;

    public float maxForce;

    [HideInInspector]

    public Vector3 velocity;

    public float intervalComputerForce = 0.2f;

    public void Start()

    {

        steerings = GetComponents<Steering>();

        InvokeRepeating("ComputerFinalSteeringForce", 0, intervalComputerForce);

    }

    public void ComputerFinalSteeringForce()

    {

        steeringForce = Vector3.zero;

        foreach (var item in steerings)

        {

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

        steeringForce=Vector3.ClampMagnitude(steeringForce, maxForce);

        if (steeringForce == Vector3.zero)

        {

            velocity = Vector3.zero; }

        acceleratedSpeed = steeringForce / mass;

    } }

1. 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

{

    public float rorationSpeed = 0.5f;

    [HideInInspector]

    public CharacterController ch;

    [HideInInspector]

    public Animator anim;

    public void Start()

    {

        base.Start();

        ch = GetComponent<CharacterController>();

        anim = GetComponent<Animator>();

    }

    public void Update()

    {

        Movement();

        Rotation();

        PlayAnim();

    }

    public void Movement()

    {

        velocity += acceleratedSpeed \* Time.deltaTime;

        if (velocity.magnitude > maxSpeed)

        {

            velocity = velocity.normalized \* maxSpeed; }

        if (isplane) velocity.y = 0;

        if (ch != null)

        {

            ch.SimpleMove(velocity); }

        else

        {

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

        } }

    public void PlayAnim()

    {

        if (anim != null)

        { }

    }

    public void Rotation()

    {

        if (velocity != Vector3.zero)

        {

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

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

        }

    }

}

1. 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;

    protected Vector3 expectationVelocity;

    public float maxSpeed;

    public Vehicle m\_vehicle;

    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;

    }

}

1. 运动物体

 using System.Collections;

using System.Collections.Generic;

using UnityEngine;

[RequireComponent(typeof(LocomtionController))]

public class SteeingForColliderObstacle : Steering

{

    public float maxSeeAhead = 5f;

    public LayerMask mask;

    public float expandRate = 10f;

    public Transform sendPos;

    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;

        RaycastHit hit;

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

       {

          expectationVelocity = hit.point - PusnPos.position;

           expectationVelocity \*= expandRate;}

       return expectationVelocity;

    } }

1. NavMeshAgent组件的代理器移动到给定目标点需要利用那个函数？该函数有几个参数？其函数有几个参数？其含义分别是什么？

NavMeshAgent.Move 移动

NavMeshAgent.acceleration 加速度

NavMeshAgent.angularSpeed 角速度

NavMeshAgent.areaMask 区域遮挡

NavMeshAgent.autoBraking 自动制动

NavMeshAgent.autoRepath 自动重新获取路径

NavMeshAgent.desiredVelocity 需求速度

NavMeshAgent.destination 目的地

NavMeshAgent.FindClosestEdge 寻找最近边缘

NavMeshAgent.hasPath 有路径

NavMeshAgent.height 高度

NavMeshAgent.isOnNavMesh 是否在导航网格上

NavMeshAgent.steeringTarget 转向目标

NavMeshAgent.hasPath 有路径

NavMeshAgent.obstacleAvoidanceType 障碍逃避类