1. 简述NavMesh的设置方法。

答：在你的场景中构建NavMesh只需要4个步骤：

1. 选择场景中需要生成寻路的几何体-可行走表面和障碍物。

2. 在NavMesh面板中选择需要烘焙寻路的物体，检测是否勾选Navigation Static.

3. 根据你的agent大小来调整bake 面板的设置。

1. Agent Radius : agent可以距离墙体 ，窗户或边缘多近的距离。
2. Agent Height : agent可以通过的最低的空间高度。
3. Max Slope : agent可以直接行走上去的最小坡度。
4. Step Height:  agent可以踩上（走上）的障碍物最高高度。

4. 点击bake按钮烘焙NavMesh。

二、简述NavMeshAgent属性参教的使用方法。

答：agent.updateRotation = false; //不允许NavMesh来旋转角色

agent.updatePosition = true; //允许NavMesh来移动角色

agent.velocity.magnitude 这个也是速度,

GetComponent().SetFloat(“Speed”, agent.velocity.magnitude);speed 移动速度

Angular Speed 转角速度 ,转身速度 角速度： 最高转速（度/秒）。

Acceleration 加速度,启动时的 最大加速度。

Stopping Distance 停止距离 ,制动距离：制动距离。到目的地的距离小于这个值，代理减速。

Auto Traverse OffMesh Link 自动遍历OffMesh链接：自动移动并关闭OffMeshLinks

Auto Repath 自动重新寻路：如果现有的部分已失效，获得新的路径。

Height 高度：代理的高度（用于调试图形）。

Base offset 基本偏移：碰撞几何体相对于实际几何体垂直的偏移。

Obstacle Avoidance Type 障碍躲避类型 ：躲避的质量水平。

NavMesh Walkable 导航网格行走：指定代理可以遍历的导航网格层类型。

三、简述寻路过程中路网烘培过程。

答：1、在window中打开Navigation窗口准备烘焙地形

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

3、在Navigation窗口中进行导航网格的烘焙

Navigation中的主要参数

 //Areas（区域）：相当于障碍物的标签，可以在寻路物体身上的Navmeshagent组件中的AreaMask中屏蔽某一区域

//Bake（烘焙）：生成指定的寻路网格

//Agent Radius： 寻路物体的半径

//Agent Heigh： 驯鹿物体的高度

//Max Slope： 寻路物体可前进的最大坡度（最大为60度）

//Step Heigh：寻路物体前进可跨过的最大高度（最大不能超过寻路物体的身高）//Drop Heigh： 寻路物体下落的最大高度

//Jump Distance 驯鹿物体跳跃的最大距离

//Object（对象，障碍物）：设置指定游戏物体的寻路烘焙属性

//Navigation Static:设置当前游戏物体为寻路静态状态

//Generate offMeshLinks：是否生成网格连接，如果勾选，网格之间会指定跳点烘焙

//Navigation Area：设置当前游戏物体的导航区域

四、对于寻路过程中的障碍物绕行应该怎样处理?

答：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

{

  [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;

    } }

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

{

    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);

        }

    }

}

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;

    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;

    }

}

4、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;

    } }

五、 NavMeshAgent组件的代理器移动到给定目标点需要利用哪个函数?该函数有几个参数？其含义分别是什么？

答：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