# **Unity完全自制游戏纸箱战争项目记录（20180711）**

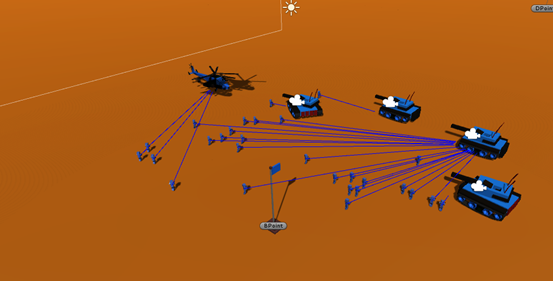
作者：[Pluto](http://gad.qq.com/user/index?id=2267380)

链接：<http://gad.qq.com/article/detail/228401>

今天实现了NPC主动寻找可使用的载具和NPC的驾驶。

效果如下图：



不是Gif图看不出效果，其实图上的飞机和坦克都是由NPC驾驶控制的。

在游戏进行过程中，每隔一秒会在范围内搜寻可用的载具，必要条件为同阵营载具，并且载具中并没有人使用。

检测方法和NPC的射击模式类似，通过射线和球形范围检测来控制条件。

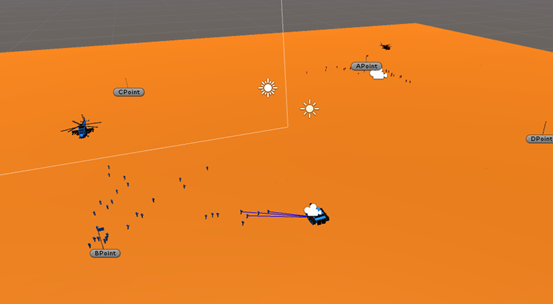
如果蓝色射线照射到的目标和检测到的目标载具不同，NPC就会以寻路状态前往目标载具，直到距离小于设定距离之后才认为“NPC登上了该载具”。

在坦克的控制脚本中加入了AI使用的一个bool选项，用来给其他AI判断该载具是否被使用，也可以用于控制坦克AI的功能。

AI驾驶的坦克同样用到了Unity自带的寻路系统，和士兵的个体AI相同，坦克的AI首先会在游戏中寻获控制管理AI，随后从控制管理AI取到应该进攻的目标点的信息从而对目标发起进攻。

因为坦克和士兵不同，坦克的底座和炮塔并不是所有时候都在一个水平线上，有可能坦克的目标在东面，而需要瞄准的单位在北面。

因此坦克的AI并不能使用到之前封装好的士兵AI方法，坦克所能进行的行为树也不多，所以就又单独写了一条AI，能够让坦克捕获目标，并且向着目标开火，会移动向占领目标点就OK了。

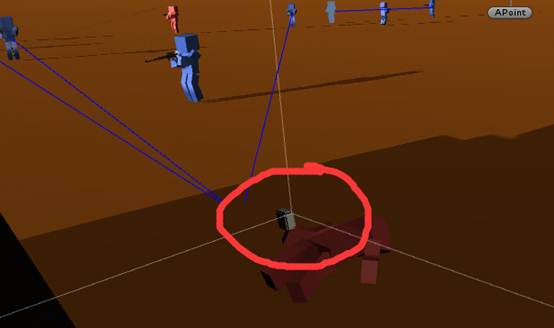


之后又把红色阵营的载具按照蓝色载具处理了一下，更改了阵营识别的数字，这样战场就出具规模了。

游戏运行之后，能看到己方NPC朝着载具蜂拥而去，驾驶载具之后朝着目标点进攻，士兵在奔跑的过程中被子弹击中，倒地身亡。



其实中间还抽空完善了一些细节，就像是NPC手中拿的东西，不再单一的是AK47了。



就像是这个死掉的红色士兵，手里拿了个医疗包。

当然了，他们拿的东西也不是随机的，看当时他想干什么，可能这个死掉的NPC当时正想要加血，然后被打死了……



另外还做了个日出日落系统，本来想实现白天过得慢一点，夜晚过的快一点，现在还没有找到好的判断白天黑夜的办法，如果时间充裕的话可以研究一下。

最后贴上一段坦克AI的代码冲字数。

using UnityEngine;

using System.Collections;

using UnityStandardAssets.Characters.FirstPerson;

public class Tank\_NPC\_AI : MonoBehaviour

｛

public bool AIEnable = false;

//寻路导航目标点

public Transform moveTargetPoint;//移动目标点

public NavMeshAgent agent;

//寻路导航炮塔看向目标点

public GameObject TankGameObject;//坦克炮塔

public Transform attackGameObject;//攻击目标点

public GameObject AI\_Human;//驾驶员

public GameObject Computer;//主机获取

public int Team=1;//小队获取

void OnEnable()

｛

GetComponent<NavMeshObstacle>().enabled = false;

GetComponent<NavMeshAgent>().enabled = true;

GetComponent<Tank\_PlayerMove>().enabled = false;

GetComponentInChildren<RigidbodyFirstPersonController>().enabled = false;

｝

void OnDisable()

｛

GetComponent<NavMeshAgent>().enabled = false;

GetComponent<Tank\_PlayerMove>().enabled = false;

GetComponentInChildren<RigidbodyFirstPersonController>().enabled = false;

｝

void Start()

｛

ComputerFind();//主机寻获

TargetFind();//目标寻获

agent = GetComponent<NavMeshAgent>();//寻路组件获取

InvokeRepeating("AttackTargetFind",0,1);//重复目标获取

｝

void Update()

｛

AttackTargetFind();//坦克攻击目标寻获

｝

private void TemporaryTargetPointFind()//目标点寻获修正

｛

if (moveTargetPoint == null || moveTargetPoint.gameObject.GetComponent<TargetPoint>().PointValue == 1)

｛

TargetFind();//目标点寻获,保证始终存在目标点

｝

｝//OK

void AttackTargetFind()//坦克攻击目标寻获

｛

agent.SetDestination(moveTargetPoint.position);

TankGameObject.transform.LookAt(new Vector3(moveTargetPoint.transform.position.x, TankGameObject.transform.position.y, moveTargetPoint.transform.position.z));

//TankGameObject.transform.Rotate(new Vector3(0, 180, 0));

TemporaryTargetPointFind();//目标点寻获修正

if (AIEnable)

｛

Camp thisGameObjectCamp = gameObject.GetComponent<Camp>();//自体组件获取

thisGameObjectCamp.AI\_Enable = true;

Collider[] RangeDetection = Physics.OverlapSphere(transform.position, 50, 1 << 9);//只检测第九层的游戏物体,场景破坏物不再遍历范围之内

foreach (var hit in RangeDetection)

｛

if (RangeDetection.Length != 0)

｛

Camp hitGameObjectScripts = hit.gameObject.GetComponent<Camp>();

if (hitGameObjectScripts != null)

｛

if (hitGameObjectScripts.Type != 0 && hitGameObjectScripts.CampNumber != thisGameObjectCamp.CampNumber)

｛

attackGameObject = hitGameObjectScripts.gameObject.transform;

return;

｝

｝

｝

｝

if (attackGameObject != null)

｛

TankGameObject.transform.LookAt(new Vector3(attackGameObject.transform.position.x, TankGameObject.transform.position.y, attackGameObject.transform.position.z));

TankGameObject.transform.Rotate(new Vector3(0, 180, 0));

｝

｝

｝

private void ComputerFind() //主机寻获

｛

if (GetComponent<Camp>().CampNumber == 1)

｛

Computer = GameObject.Find("AI\_Red\_Administration");

｝

else if (GetComponent<Camp>().CampNumber == 2)

｛

Computer = GameObject.Find("AI\_Blue\_Administration");

｝

｝//OK

private void TargetFind()//目标寻获

｛

if (Team == 1)

｛

moveTargetPoint = Computer.GetComponent<AI\_Management>().ATeam\_AccidentalTransform;

｝

if (Team == 2)

｛

moveTargetPoint = Computer.GetComponent<AI\_Management>().BTeam\_AccidentalTransform;

｝

if (Team == 3)

｛

moveTargetPoint = Computer.GetComponent<AI\_Management>().CTeam\_AccidentalTransform;

｝

｝//OK

｝

using UnityEngine;

using System.Collections;

using UnityStandardAssets.Characters.FirstPerson;

public class Tank\_NPC\_AI : MonoBehaviour

｛

    public bool AIEnable = false;

    //寻路导航目标点

    public Transform moveTargetPoint;//移动目标点

    public NavMeshAgent agent;

    //寻路导航炮塔看向目标点

    public GameObject TankGameObject;//坦克炮塔

    public Transform attackGameObject;//攻击目标点

    public GameObject AI\_Human;//驾驶员

    public GameObject Computer;//主机获取

    public int Team=1;//小队获取

    void OnEnable()

    ｛

        GetComponent<NavMeshObstacle>().enabled = false;

        GetComponent<NavMeshAgent>().enabled = true;

        GetComponent<Tank\_PlayerMove>().enabled = false;

        GetComponentInChildren<RigidbodyFirstPersonController>().enabled = false;

    ｝

    void OnDisable()

    ｛

        GetComponent<NavMeshAgent>().enabled = false;

        GetComponent<Tank\_PlayerMove>().enabled = false;

        GetComponentInChildren<RigidbodyFirstPersonController>().enabled = false;

    ｝

    void Start()

    ｛

        ComputerFind();//主机寻获

        TargetFind();//目标寻获

        agent = GetComponent<NavMeshAgent>();//寻路组件获取

        InvokeRepeating("AttackTargetFind",0,1);//重复目标获取

    ｝

    void Update()

    ｛

        AttackTargetFind();//坦克攻击目标寻获

    ｝

    private void TemporaryTargetPointFind()//目标点寻获修正

    ｛

        if (moveTargetPoint == null || moveTargetPoint.gameObject.GetComponent<TargetPoint>().PointValue == 1)

        ｛

            TargetFind();//目标点寻获,保证始终存在目标点

        ｝

    ｝//OK

    void AttackTargetFind()//坦克攻击目标寻获

    ｛

        agent.SetDestination(moveTargetPoint.position);

        TankGameObject.transform.LookAt(new Vector3(moveTargetPoint.transform.position.x, TankGameObject.transform.position.y, moveTargetPoint.transform.position.z));

        //TankGameObject.transform.Rotate(new Vector3(0, 180, 0));

        TemporaryTargetPointFind();//目标点寻获修正

        if (AIEnable)

        ｛

            Camp thisGameObjectCamp = gameObject.GetComponent<Camp>();//自体组件获取

            thisGameObjectCamp.AI\_Enable = true;

            Collider[] RangeDetection = Physics.OverlapSphere(transform.position, 50, 1 << 9);//只检测第九层的游戏物体,场景破坏物不再遍历范围之内

            foreach (var hit in RangeDetection)

            ｛

                if (RangeDetection.Length != 0)

                ｛

                    Camp hitGameObjectScripts = hit.gameObject.GetComponent<Camp>();

                    if (hitGameObjectScripts != null)

                    ｛

                        if (hitGameObjectScripts.Type != 0 && hitGameObjectScripts.CampNumber != thisGameObjectCamp.CampNumber)

                        ｛

                            attackGameObject = hitGameObjectScripts.gameObject.transform;

                            return;

                        ｝

                    ｝

                ｝

            ｝

            if (attackGameObject != null)

            ｛

                TankGameObject.transform.LookAt(new Vector3(attackGameObject.transform.position.x, TankGameObject.transform.position.y, attackGameObject.transform.position.z));

                TankGameObject.transform.Rotate(new Vector3(0, 180, 0));

            ｝

        ｝

    ｝

    private void ComputerFind() //主机寻获

    ｛

        if (GetComponent<Camp>().CampNumber == 1)

        ｛

            Computer = GameObject.Find("AI\_Red\_Administration");

        ｝

        else if (GetComponent<Camp>().CampNumber == 2)

        ｛

            Computer = GameObject.Find("AI\_Blue\_Administration");

        ｝

    ｝//OK

    private void TargetFind()//目标寻获

    ｛

        if (Team == 1)

        ｛

            moveTargetPoint = Computer.GetComponent<AI\_Management>().ATeam\_AccidentalTransform;

        ｝

        if (Team == 2)

        ｛

            moveTargetPoint = Computer.GetComponent<AI\_Management>().BTeam\_AccidentalTransform;

        ｝

        if (Team == 3)

        ｛

            moveTargetPoint = Computer.GetComponent<AI\_Management>().CTeam\_AccidentalTransform;

        ｝

    ｝//OK

｝