Unity AI行为之群组行为

文章来自CSDN博客 2018-09-30 50浏览

场景 JavaScript Unity

群组行为是指多个对象组队同时行进的情况，我们可以坐下来，告诉每一个对象它该如何移动，但这样做的工作量太大。取而代之的是，我们去创建一个群组的领导，让它来为我们做这些，这样我们所要做的就只是设置一些规则，然后群组中的boid就会自行组队。在本篇文章中，我们将学习如何在unity3d中实现这种群组行为。

每个boid都可以应用一下三个基本的规则：

- 分离（seperation）：群组中的每个个体都与相邻个体保持一定距离
- 列队（alignment）：群组以相同速度，向相同方向移动
- 凝聚（cohesion）：群组中心保持最小距离

在本篇文章中我们将创建自己的场景，场景里会有一群对象，并使用C#实现群组行为。有两个主要的组成部分：每个boid行为，以及维持并领导整个群组的主要控制者。我们的场景层级如图，在一个名为UnityFlockController的控制器下面有一些boid实体----UnityFlock。每一个UnityFlock实体都是一个boid对象，它们会引用其父对象UnityFlockController实体作为它们的领导者。UnityFlockController将会在到达目标位置后，随机的更新下一个目标位置。UnityFlock是一个预制体，这个预制体仅仅是一个立方体网格，并拥有UnityFlock脚本。我们可以使用任何更有意思的其他的网格标识这个预制体，比如小鸟。

个体行为boid是Craig Reynold创造的术语，用以表示类似小鸟这样的对象。我们将使用这个术语描述群组中的每个个体对象。UnityFlock这个脚本控制群组中每一个boid的行为。

```
1  using UnityEngine;
2  using System.Collections;
3  public class UnityFlock : MonoBehaviour {
4      public float minSpeed = 20;//最小移动速度
5      public float turnSpeed = 20;//旋转速度
6      public float randomFreq = 20;//用来确定更新randomPush变量的次数
7      public float randomForce = 20;//这个力产生出一个随机增长和降低的速度，并使得群
8      //alignment variables列队变量
9      public float toOriginForce = 50;//用这个来保持所有boid在一个范围内，并与群组的原
10     public float toOriginRange = 100;//群组扩展的程度
11     public float gravity = 2;
12     //seperation variables分离变量
13     public float avoidanceRadius = 50;
14     public float avoidanceForce = 20;//这两个变量可以让每个boid个体之间保持最小距离
15     //cohesion variables凝聚变量，这两个变量可用来与群组的领导者或群组的原点保持最
16     public float followVelocity = 4;
17     public float followRadius = 40;
18     //这些变量控制了boid的移动
19     private Transform origin;//设为父对象，以控制整个群组中的对象。
20     private Vector3 velocity;
21     private Vector3 normalizeedVelocity;
22     private Vector3 randomPush;//更新基于randomFore的值
23     private Vector3 originPush;
24     //以下两个变量存储相邻boid的信息，当前boid需要知道群组中其他boid的信息
25     private Transform[] objects;
26     private UnityFlock[] otherFlocks;
27     private Transform transformComponent;
28     /*
29        */
30     void Start () {
31        randomFreq = 1.0f / randomFreq;
32        //将父类指定为origin
33        origin = transform.parent;
```

本文作者

晓杉枝樟

```
    //Flock transform
    transformComponent = transform;
    //Temporary components临时
    Component[] tempFlocks = null;
    //Get all the unity flock omponents from the parent transform in the group
    if (transform.parent )
     {
       tempFlocks = transform.parent.GetComponentsInChildren&lt;UnityFlock&g
    }
    //Assign and store all the flock objects in this group
    objects = new Transform[tempFlocks.Length];
    otherFlocks = new UnityFlock[tempFlocks.Length];
    for (int i = 0; i &lt; tempFlocks.Length; i++)
     {
       objects[i] = tempFlocks[i].transform;
       otherFlocks[i] = (UnityFlock)tempFlocks[i];
    }
    //Null Parent as the flok leader will be UnityFlockController object
    transform.parent = null;
    //Calculate random push depends on the random frequency provided
    StartCoroutine(UpdateRandom());
  }
  IEnumerator UpdateRandom()
   {
    while (true)
     {
       randomPush = Random.insideUnitSphere * randomForce;
       yield return new WaitForSeconds(randomFreq + Random.Range(-randomFr
    }
  }
  void Update ()  {
    //internal variables
    float speed = velocity.magnitude;//获取速度大小
    Vector3 avgVelocity = Vector3.zero;
    Vector3 avgPosition = Vector3.zero;
    float count = 0;
    float f = 0.0f;
    float d = 0.0f;
    Vector3 myPosition = transformComponent.position;
    Vector3 forceV;
    Vector3 toAvg;
    Vector3 wantedVel;
    for (int i = 0; i &lt; objects.Length; i++)
     {
       Transform transform = objects[i];
       if (transform != transformComponent)
        {
          Vector3 otherPosition = transform.position;
          //Average position to calculate cohesion
          avgPosition += otherPosition;
          count++;
          //Directional vector from other flock to this flock
          forceV = myPosition - otherPosition;
          //Magnitude of that diretional vector(length)
          d = forceV.magnitude;
          //Add push value if the magnitude,the length of the vetor,is less than foll
          if (d &lt; followRadius)
           {
            //calculate the velocity,the speed of the object,based current magnitu
            if (d &lt; avoidanceRadius)
             {
               f = 1.0f - (d / avoidanceRadius);
               if (d &gt; 0)
                {
                  avgVelocity += (forceV / d) * f * avoidanceForce;
               }
            }
```

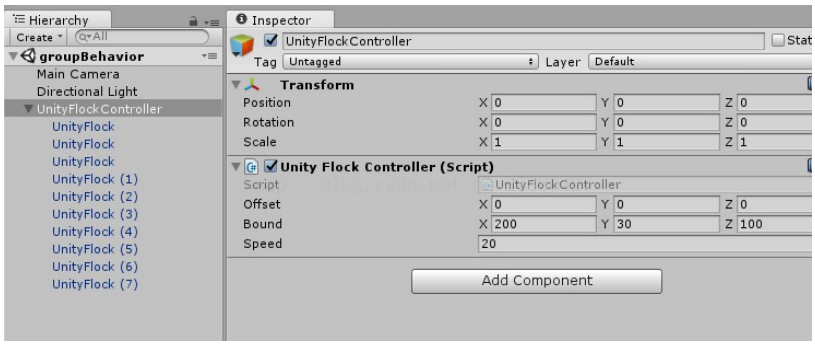
```
                //just keep the current distance with the leader
                f = d / followRadius;
                UnityFlock otherSealgull = otherFlocks[i];
                //we normalize the otherSealgull veloity vector to get the direction of
                avgVelocity += otherSealgull.normalizeedVelocity * f * followVelocity;
            }
        }
    }
    //上述代码实现了分离规则，首先，检查当前boid与其他boid之间的距离，并相应的
    if (count &gt; 0)
     {
       //Calculate the aveage flock veloity(Alignment)
       avgVelocity /= count;
       //Calculate Center value of the flock（Cohesion)
       toAvg = (avgPosition / count) - myPosition;
    }
    else
     {
       toAvg = Vector3.zero;
    }
    //Directional Vector to the leader
    forceV = origin.position - myPosition;
    d = forceV.magnitude;
    f = d / toOriginRange;
    //Calculate the velocity of the flok to the leader
    if (d &gt; 0)
     {
       originPush = (forceV / d) * f * toOriginForce;
    }
    if (speed &lt; minSpeed &amp;&amp; speed &gt; 0)
     {
       velocity = (velocity / speed) * minSpeed;
    }
    wantedVel = velocity;
    //Calculate final velocity
    wantedVel -= wantedVel * Time.deltaTime;
    wantedVel += randomPush * Time.deltaTime;
    wantedVel += originPush * Time.deltaTime;
    wantedVel += avgVelocity * Time.deltaTime;
    wantedVel += toAvg.normalized * gravity * Time.deltaTime;
    //Final Velocity to rotate the flock into
    velocity = Vector3.RotateTowards(velocity, wantedVel, turnSpeed * Time.delta
    transformComponent.rotation = Quaternion.LookRotation(velocity);
    transformComponent.Translate(velocity * Time.deltaTime, Space.World);
    normalizeedVelocity = velocity.normalized;
  }
} <span style="white-space: normal;"> </span>
```

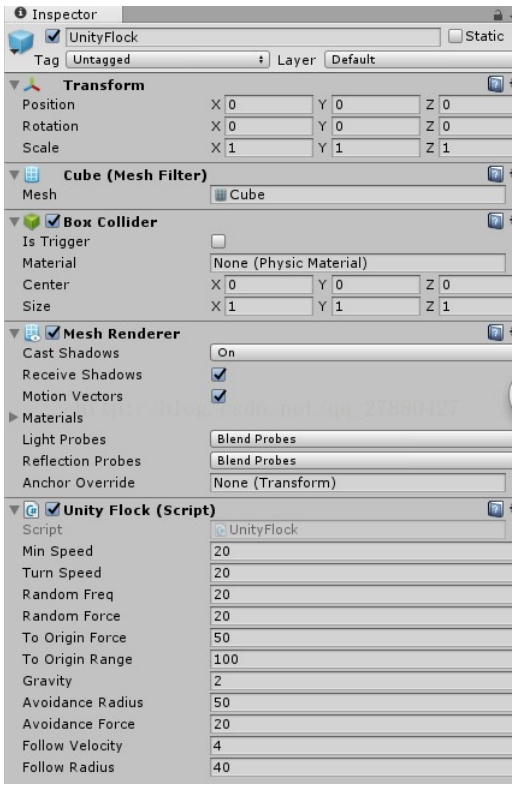
现在是时候创建控制器了，这个类会更新自己的位置，这样其他的boid个体对象就知道该去哪里，这个对象由前面的UnityFlock脚本中的origin变量引用而来。

```
using UnityEngine;
using System.Collections;
public class UnityFlockController : MonoBehaviour  {
   public Vector3 offset;
   public Vector3 bound;
   public float speed = 100.0f;
   private Vector3 initialPosition;
   private Vector3 nextMovementPoint;
   void Start ()  {
      initialPosition = transform.position;
      CalculateNextMovementPoint();
   }
   void Update ()  {
      transform.Translate(Vector3.forward * speed * Time.deltaTime);
      transform.rotation = Quaternion.Slerp(transform.rotation, Quaternion.LookRot
      if (Vector3.Distance(nextMovementPoint,transform.position) &lt;= 10.0f)
```

```
        {
            CalculateNextMovementPoint();
        }
    }
    /*
     在我们的Update（）方法中，检查控制器对象是否在最终目标位置附近，如果在，使用
        */
    void CalculateNextMovementPoint()
     {
        float posX = Random.Range(initialPosition.x - bound.x, initialPosition.x + boun
        float posY = Random.Range(initialPosition.y - bound.y, initialPosition.y + boun
        float posZ = Random.Range(initialPosition.z - bound.z, initialPosition.z + boun
        nextMovementPoint = initialPosition + new Vector3(posX, posY, posZ);
    }
}

```





来自：https://blog.csdn.net/qq_27880427/article/details/72771510

Unity AI系列

1、群组行为之群集动画
3、群组行为CraigReynold算法
4、路径跟随
5、避开障碍物简单实现

0 评论

赶快 登录 发表评论吧

没有更多消息了...

相关推荐

热门

策划

程序

技术前沿

音频

项目管理

游戏运营

游戏测试

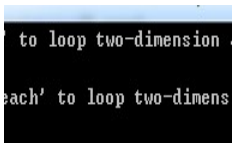
晓杉枝樟 10小时前

格子游戏类型实践（二）

接上篇格子游戏类型实践（一）框架部分，这篇继续给大家介绍格子游戏类型实践的第二篇，我们总共要写这几种格子类型的游戏，像保卫萝卜类型的塔防，其实我个人倾向于一个经典塔防——矢量塔防，贪吃蛇，扫雷，三消（三…

0个赞 0条评论

晓杉枝樟 10小时前



C# foreach循环与for循环对比

C#为我们提供了若干种循环语句，分别适用于不同的情形，其中就包括了foreach循环与for循环，下面要给大家分享下是使用foreach循环和for的区别，给大家做个…

0个赞 0条评论

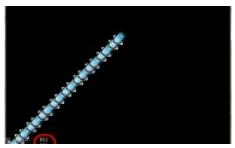
晓杉枝樟 10小时前

C#获取机器唯一识别码

在客户端认证的过程中，我们总要获取客户机的唯一识别信息，曾经以为MAC地址是不会变的，但是现在各种改，特别是使用无线上网卡，MAC地址插一次变一次，所以这样使用MAC就没有什么意义了，怎么办，又开始求助Go…

0个赞 0条评论

晓杉枝樟 10小时前

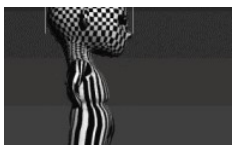


Cocos2dx-SpriteBatchNode与SpriteFrameCache加...

大家都知道一个游戏里面会有大量的图片，每一个图片渲染是需要时间的，以下分析使用SpriteBatchNode与SpriteFrameCache这两个类来加快渲染速度，加快游…

0个赞 0条评论

晓杉枝樟 10小时前



Cocos2dx 将3dmax模型和动画导入游戏

现在手机的硬件有了很大的发展，越来越多的3D手游出现在玩家的视野当中。幸好cocos为我们提供了导入3D模型的方法，下面就来看看是如何将3dmax模型和动…

0个赞 0条评论

晓杉枝樟 10小时前

Directx 11中垂直同步的设置

现在很多游戏特效设置了里边都有"垂直同步"这个选项，本篇要给大家介绍的是在Directx 11中垂直同步的设置方法，如果有不了解的可以看看。1、什么是垂直同步？垂直同步又称场同步（Vertical Hold），从CRT显示器的显示原…

0个赞 0条评论

晓杉枝樟 10小时前

C#中foreach循环原理

在探讨foreach如何内部如何实现这个问题之前，我们需要理解两个C#里边的接口，IEnumerable 与 IEnumerator. 在C#里边的遍历集合时用到的相关类中，IEnumerable是最基本的接口。这是一个可以进行泛型化的接口，比如说IE…

0个赞 0条评论

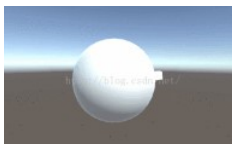
晓杉枝樟　11小时前

Unity5 SkinnedMesh 换装

这篇文章主要给大家介绍下Unity5中的 SkinnedMesh换装。游戏中常见的换装做法：直接更换贴图更换静态Mesh更换SkinnedMesh一、SkinnedMesh原理Skinned…

0个赞　0条评论

晓杉枝樟　2018-10-12



Unity实现围绕星球运动

下面来看看在unity开发过程中如何实现围绕星球运动。一、效果图当然如果是规则的圆使用RotateAround可以轻松实现，但是如果这个球有凹凸部分RotateAround…

0个赞　0条评论

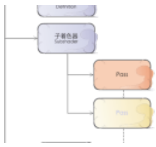
晓杉枝樟　2018-10-12

Unity Socket编程过程

Socket是进程通讯的一种方式，即调用这个网络库的一些API函数实现分布在不同主机的相关进程之间的数据交换。下面就分享下Unity中Socket编程的过程。一、第一步开始连接1、创建socket private Socket socket; socket=newS…

0个赞　0条评论

晓杉枝樟　2018-10-12



Shader基本结构

Shader大体上可以分为两类，简单来说1.表面着色器（Surface Shader） - 为你做了大部分的工作，只需要简单的技巧即可实现很多不错的效果。类比卡片机，上…

0个赞　0条评论

腾讯游戏学院公众号