

必应

明白了，让我们重新开始。接下来应该讨论什么？

用户

请写一个 unity 用的脚本，这个脚本会持续性的对挂载此脚本的游戏对象随机缩放尺寸，随机旋转和位移，请注意，所有对游戏对象的处理都不会超出相机 FOV。请给出代码。

必应

这是一个简单的脚本，它可以持续性地对挂载此脚本的游戏对象进行随机缩放、旋转和位移。请注意，您需要根据您的相机设置和游戏对象的初始位置来调整代码中的最大/最小值，以确保游戏对象不会超出相机的视野。

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class RandomTransform : MonoBehaviour
{
    public float minScale = 0.5f;
    public float maxScale = 1.5f;
    public float minPosition = -5f;
    public float maxPosition = 5f;
    public float minRotation = -180f;
    public float maxRotation = 180f;

    void Update()
    {
        // 随机缩放
        float randomScale = Random.Range(minScale, maxScale);
        transform.localScale = new Vector3(randomScale, randomScale, randomScale);

        // 随机旋转
        float randomRotationX = Random.Range(minRotation, maxRotation);
        float randomRotationY = Random.Range(minRotation, maxRotation);
        float randomRotationZ = Random.Range(minRotation, maxRotation);
        transform.eulerAngles = new Vector3(randomRotationX, randomRotationY, randomRotationZ);

        // 随机位移
        float randomPositionX = Random.Range(minPosition, maxPosition);
        float randomPositionY = Random.Range(minPosition, maxPosition);
        float randomPositionZ = Random.Range(minPosition, maxPosition);
        transform.position = new Vector3(randomPositionX, randomPositionY, randomPositionZ);
    }
}
```

用户

哈哈啊哈哈，非常棒，但是请加上速度控制，现在变化太迅速了

必应

好的，我们可以通过添加一个变量来控制游戏对象的变化速度。下面是修改后的代码，其中添加了一个 speed 变量来控制游戏对象的变化速度。您可以根据需要调整 speed 的值。

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class RandomTransform : MonoBehaviour
{
    public float minScale = 0.5f;
    public float maxScale = 1.5f;
    public float minPosition = -5f;
    public float maxPosition = 5f;
    public float minRotation = -180f;
    public float maxRotation = 180f;
    public float speed = 0.1f;

    void Update()
    {
        // 随机缩放
        float randomScale = Random.Range(minScale, maxScale);
        transform.localScale = Vector3.Lerp(transform.localScale, new Vector3(randomScale, randomScale, randomScale), speed * Time.deltaTime);

        // 随机旋转
        float randomRotationX = Random.Range(minRotation, maxRotation);
        float randomRotationY = Random.Range(minRotation, maxRotation);
        float randomRotationZ = Random.Range(minRotation, maxRotation);
        transform.rotation = Quaternion.Lerp(transform.rotation, Quaternion.Euler(randomRotationX, randomRotationY, randomRotationZ), speed * Time.deltaTime);

        // 随机位移
        float randomPositionX = Random.Range(minPosition, maxPosition);
        float randomPositionY = Random.Range(minPosition, maxPosition);
        float randomPositionZ = Random.Range(minPosition, maxPosition);
        transform.position = Vector3.Lerp(transform.position, new Vector3(randomPositionX, randomPositionY, randomPositionZ), speed * Time.deltaTime);
    }
}
```

用户

缩放，旋转，位移 应该有各自的 speed

必应

好的，我们可以为缩放、旋转和位移分别添加各自的速度控制变量。下面是修改后的代码，其中添加了 scaleSpeed、rotationSpeed 和 positionSpeed 三个变量来分别控制游戏对象的缩放、旋转和位移速度。您可以根据需要调整这些变量的值。

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class RandomTransform : MonoBehaviour
{
    public float minScale = 0.5f;
    public float maxScale = 1.5f;
    public float minPosition = -5f;
    public float maxPosition = 5f;
    public float minRotation = -180f;
    public float maxRotation = 180f;
    public float scaleSpeed = 0.1f;
    public float rotationSpeed = 0.1f;
    public float positionSpeed = 0.1f;

    void Update()
    {
        // 随机缩放
        float randomScale = Random.Range(minScale, maxScale);
        transform.localScale = Vector3.Lerp(transform.localScale, new Vector3(randomScale, randomScale, randomScale), scaleSpeed * Time.deltaTime);

        // 随机旋转
        float randomRotationX = Random.Range(minRotation, maxRotation);
        float randomRotationY = Random.Range(minRotation, maxRotation);
        float randomRotationZ = Random.Range(minRotation, maxRotation);
        transform.rotation = Quaternion.Lerp(transform.rotation, Quaternion.Euler(randomRotationX, randomRotationY, randomRotationZ), rotationSpeed * Time.deltaTime);

        // 随机位移
        float randomPositionX = Random.Range(minPosition, maxPosition);
        float randomPositionY = Random.Range(minPosition, maxPosition);
        float randomPositionZ = Random.Range(minPosition, maxPosition);
        transform.position = Vector3.Lerp(transform.position, new Vector3(randomPositionX, randomPositionY, randomPositionZ), positionSpeed * Time.deltaTime);
    }
}

```

用户

很不错，但是模型在抖动，应该是所有的位置，旋转获取的随机值应该加上原始位置和旋转值吧？

必应

您说得对，模型的抖动可能是因为每一帧都在获取新的随机值，导致游戏对象的位置、旋转和缩放在不断变化。我们可以通过在每一帧中使用相同的随机值来解决这个问题。

下面是修改后的代码，其中添加了 `targetScale`、`targetRotation` 和 `targetPosition` 三个变量来存储游戏对象的目标缩放、旋转和位移值。这些变量会在游戏对象到达目标值时更新为新的随机值。

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class RandomTransform : MonoBehaviour
{
    public float minScale = 0.5f;
    public float maxScale = 1.5f;
    public float minPosition = -5f;
    public float maxPosition = 5f;
    public float minRotation = -180f;
    public float maxRotation = 180f;
    public float scaleSpeed = 0.1f;
    public float rotationSpeed = 0.1f;
    public float positionSpeed = 0.1f;

    private Vector3 targetScale;
    private Quaternion targetRotation;
    private Vector3 targetPosition;

    void Start()
    {
        // 初始化目标缩放、旋转和位移值
        targetScale = transform.localScale;
        targetRotation = transform.rotation;
        targetPosition = transform.position;
    }

    void Update()
    {
        // 随机缩放
        if (Vector3.Distance(transform.localScale, targetScale) < 0.1f)
        {
            float randomScale = Random.Range(minScale, maxScale);
            targetScale = new Vector3(randomScale, randomScale, randomScale);
        }
        transform.localScale = Vector3.Lerp(transform.localScale, targetScale, scaleSpeed * Time.deltaTime);

        // 随机旋转
        if (Quaternion.Angle(transform.rotation, targetRotation) < 5f)
        {
            float randomRotationX = Random.Range(minRotation, maxRotation);
            float randomRotationY = Random.Range(minRotation, maxRotation);
            float randomRotationZ = Random.Range(minRotation, maxRotation);
            targetRotation = Quaternion.Euler(randomRotationX, randomRotationY, randomRotationZ);
        }
        transform.rotation = Quaternion.Lerp(transform.rotation, targetRotation, rotationSpeed * Time.deltaTime);

        // 随机位移
        if (Vector3.Distance(transform.position, targetPosition) < 0.1f)
        {
            float randomPositionX = Random.Range(minPosition, maxPosition);
            float randomPositionY = Random.Range(minPosition, maxPosition);
            float randomPositionZ = Random.Range(minPosition, maxPosition);
            targetPosition = new Vector3(randomPositionX, randomPositionY, randomPositionZ);
        }
        transform.position = Vector3.Lerp(transform.position, targetPosition, positionSpeed * Time.deltaTime);
    }
}

```

很酷，解决了抖动问题，但是我看到 minPosition maxposition 限定的是一个矩形范围。而往往屏幕都是 16:9 这样的矩形，会导致模型跑出屏幕，这该如何解决呢？

必应

您可以通过调整 minPosition 和 maxPosition 的值来限制游戏对象在屏幕内移动。例如，如果您的屏幕比例为 16:9，那么您可以将 minPosition 和 maxPosition 的值分别设置为 -8 和 8，这样游戏对象就会在一个宽为 16、高为 9 的矩形区域内移动。

此外，您还可以使用相机的 WorldToViewportPoint 方法来检测游戏对象是否在屏幕内。如果游戏对象不在屏幕内，则可以将其移动回屏幕内。下面是一个示例代码，它演示了如何使用 WorldToViewportPoint 方法来检测游戏对象是否在屏幕内，并将其移动回屏幕内。

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class RandomTransform : MonoBehaviour
{
    public Camera mainCamera;
    public float minScale = 0.5f;
    public float maxScale = 1.5f;
    public float minPosition = -5f;
    public float maxPosition = 5f;
    public float minRotation = -180f;
    public float maxRotation = 180f;
    public float scaleSpeed = 0.1f;
    public float rotationSpeed = 0.1f;
    public float positionSpeed = 0.1f;

    private Vector3 targetScale;
    private Quaternion targetRotation;
    private Vector3 targetPosition;

    void Start()
    {
        // 初始化目标缩放、旋转和位移值
        targetScale = transform.localScale;
        targetRotation = transform.rotation;
        targetPosition = transform.position;
    }

    void Update()
    {
        // 随机缩放
        if (Vector3.Distance(transform.localScale, targetScale) < 0.1f)
        {
            float randomScale = Random.Range(minScale, maxScale);
            targetScale = new Vector3(randomScale, randomScale, randomScale);
        }
        transform.localScale = Vector3.Lerp(transform.localScale, targetScale, scaleSpeed * Time.deltaTime);

        // 随机旋转
        if (Quaternion.Angle(transform.rotation, targetRotation) < 5f)
        {
            float randomRotationX = Random.Range(minRotation, maxRotation);
            float randomRotationY = Random.Range(minRotation, maxRotation);
            float randomRotationZ = Random.Range(minRotation, maxRotation);
            targetRotation = Quaternion.Euler(randomRotationX, randomRotationY, randomRotationZ);
        }
        transform.rotation = Quaternion.Lerp(transform.rotation, targetRotation, rotationSpeed * Time.deltaTime);

        // 随机位移
        if (Vector3.Distance(transform.position, targetPosition) < 0.1f)
        {
            float randomPositionX = Random.Range(minPosition, maxPosition);
            float randomPositionY = Random.Range(minPosition, maxPosition);
            float randomPositionZ = Random.Range(minPosition, maxPosition);
            targetPosition = new Vector3(randomPositionX, randomPositionY, randomPositionZ);
        }
        transform.position = Vector3.Lerp(transform.position, targetPosition, positionSpeed * Time.deltaTime);

        // 检测游戏对象是否在屏幕内
        Vector3 viewportPoint = mainCamera.WorldToViewportPoint(transform.position);
        if (viewportPoint.x < 0 || viewportPoint.x > 1 || viewportPoint.y < 0 || viewportPoint.y > 1)
        {
            // 将游戏对象移动回屏幕内
            transform.position = mainCamera.ViewportToWorldPoint(new Vector3(0.5f, 0.5f, viewportPoint.z));
            targetPosition = transform.position;
        }
    }
}
```

用户

将 minPosition 和 maxPosition 的值分别设置为 -8 和 8，这样游戏对象就会在一个宽为 16、高为 9 的矩形区域内移动。这句话后半段应该是 这样游戏对象就会在一个宽为 9、高为 9 的矩形区域内移动。而宽为 16 的屏幕模型就移动的很憋屈吧，所以 minPosition ,maxposition 只能限定一个轴向，要么 height 要么 width，接下来如何更灵活的控制宽高呢

必应

您说得对，我之前的回答有误。如果您希望游戏对象在一个宽为 16、高为 9 的矩形区域内移动，那么您应该分别为 x 和 y 轴设置不同的最小/最大值。例如，您可以将 minPositionX 和 maxPositionX 的值分别设置为 -8 和 8，将 minPositionY 和 maxPositionY 的值分别设置为 -4.5 和 4.5。

下面是修改后的代码，其中添加了 minPositionX、maxPositionX、minPositionY 和 maxPositionY 四个变量来分别控制游戏对象在 x 和 y 轴上的移动范围。

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class RandomTransform : MonoBehaviour
{
    public Camera mainCamera;
    public float minScale = 0.5f;
    public float maxScale = 1.5f;
```

```

public float minPositionX = -8f;
public float maxPositionX = 8f;
public float minPositionY = -4.5f;
public float maxPositionY = 4.5f;
public float minRotation = -180f;
public float maxRotation = 180f;
public float scaleSpeed = 0.1f;
public float rotationSpeed = 0.1f;
public float positionSpeed = 0.1f;

private Vector3 targetScale;
private Quaternion targetRotation;
private Vector3 targetPosition;

void Start()
{
    // 初始化目标缩放、旋转和位移值
    targetScale = transform.localScale;
    targetRotation = transform.rotation;
    targetPosition = transform.position;
}

void Update()
{
    // 随机缩放
    if (Vector3.Distance(transform.localScale, targetScale) < 0.1f)
    {
        float randomScale = Random.Range(minScale, maxScale);
        targetScale = new Vector3(randomScale, randomScale, randomScale);
    }
    transform.localScale = Vector3.Lerp(transform.localScale, targetScale, scaleSpeed * Time.deltaTime);

    // 随机旋转
    if (Quaternion.Angle(transform.rotation, targetRotation) < 5f)
    {
        float randomRotationX = Random.Range(minRotation, maxRotation);
        float randomRotationY = Random.Range(minRotation, maxRotation);
        float randomRotationZ = Random.Range(minRotation, maxRotation);
        targetRotation = Quaternion.Euler(randomRotationX, randomRotationY, randomRotationZ);
    }
    transform.rotation = Quaternion.Lerp(transform.rotation, targetRotation, rotationSpeed * Time.deltaTime);

    // 随机位移
    if (Vector3.Distance(transform.position, targetPosition) < 0.1f)
    {
        float randomPositionX = Random.Range(minPositionX, maxPositionX);
        float randomPositionY = Random.Range(minPositionY, maxPositionY);
        float randomPositionZ = transform.position.z; // 不改变 z 坐标
        targetPosition = new Vector3(randomPositionX, randomPositionY, randomPositionZ);
    }
    transform.position = Vector3.Lerp(transform.position, targetPosition, positionSpeed * Time.deltaTime);

    // 检测游戏对象是否在屏幕内
    Vector3 viewportPoint = mainCamera.WorldToViewportPoint(transform.position);
    if (viewportPoint.x < 0 || viewportPoint.x > 1 || viewportPoint.y < 0 || viewportPoint.y > 1)
    {
        // 将游戏对象移动回屏幕内
        transform.position = mainCamera.ViewportToWorldPoint(new Vector3(0.5f, 0.5f, viewportPoint.z));
        targetPosition = transform.position;
    }
}
}

```