【游戏仿真实验】使用Unity仿真电视机光学三原色显示画 面,我是要成为海贼王的男人

原创

林新发 ● 于 2021-08-22 08:08:48 发布 ● 3581 ☆ 收藏 43

分类专栏: Unity3D 文章标签: unity 三原色 电视机 光学 仿真

版权声明:本文为博主原创文章,遵循 CC 4.0 BY-SA 版权协议,转载请附上原文出处链接和本声明。

本文链接: <u>https://blog.csdn.net/linxinfa/article/details/119845923</u>

版权



Unity3D 专栏收录该内容

476 篇文章 689 订阅 订阅专栏 本文效果如下









- 一、前言
- 二、光学三原色
- 三、Unity制作红绿蓝色块
- 四、克隆像素块
- 五、显示图像
- 六、测试
- 七、屏幕后处理
- 七、工程源码
- 八、完毕

一、前言

嗨,大家好,我是新发。

最近我在尝试用 Unity 制作非游戏的场景,做一些数学、物理的仿真实验,比如上一篇文章我做了一个《蒲丰投针仿真实验》 (点击查看),今天,我又要做另一个实验,模拟电视机光学三原色的原理来显示画面,话不多说,我们开始吧~

二、光学三原色

光学三原色分别为红、绿、蓝,这三个颜色通道色各分为 256 阶亮度,在 0 时最弱,而在 255 时最亮。也就是说,一共可以表现 256 x 256 x 256 = 16777216 种颜色。



我们早期的彩色电视机显示器就是利用三原色的叠加来显示不同的色彩的,如果你用放大镜仔细观察电视机的屏幕,就会发现画 面其实是由很多很多红绿蓝的小色块排列组合而成的,





三、Unity制作红绿蓝色块

创建一个 Unity 工程,在 Hierarchy 面板空白处右键鼠标,点击菜单 3D Object / Quad,创建一个四边形,

Cut Copy Paste Paste As Child		
Rename Duplicate Delete		
Select Children		
Set as Default Parent		
Create Empty		
3D Object	>	Cube
Effects	>	Sphere
Light	>	Capsule
Audio	>	Cylinder
Video	>	Plane
UI	>	Quad
Camera		Text - Text JeshPro
Scene Variables		Ragdoll
		Terrain

Wind Zone 3D Text

如下:



调整四边形的 Scale 的 x 缩放为 0.33,



如下,





为它创建一个材质球, shader 使用 Unlit/Color, 颜色设置为红色,



把材质球赋值给四边形,



如下





同理,制作蓝色和绿色,排在一起,如下:



层级结构如下,一个 block 就等价于一个像素,一个像素包括 r、g、b 三个三原色,



可以分别对r、g、b进行调整,



四、克隆像素块

接下来我们只需要进行克隆,排列好,理论上就可以做成一个光学三原色显示器了,



写个函数来执行:

```
创建N个像素块,WIDTH是宽,HEIGHT是高
private IEnumerator CreateBlocks()
   int counter = 0;
   for (int j = HEIGHT; j \ge 0; --j)
       for (int i = 0; i < WIDTH; ++i)
           CreateBlockItem(i, j, Color.white);
           ++counter;
           if (counter > 100)
               counter = 0;
               yield return new WaitForSeconds(0.3f);
// 创建一个像素块
private void CreateBlockItem(int i, int j, Color color)
   var obj = Instantiate(block);
   var trans = obj.transform;
   trans.SetParent(blockRoot, false);
   trans.localPosition = new Vector3(i, j, 0);
   trans.Find("r").GetComponent<MeshRenderer>().material.color = new Color(color.r, 0, 0);
   trans.Find("g").GetComponent<MeshRenderer>().material.color = new Color(0, color.g, 0);
   trans.Find("b").GetComponent<MeshRenderer>().material.color = new Color(0, 0, color.b);
```

比如创建一个宽 100, 高 50 的屏幕, 效果如下:

五、显示图像

我们找一张图片,比如海贼王的图片,



导入到 Unity 工程中,勾选图片可读,



我们可以看到图片的尺寸为 1920×1080,



我们使用 240x135 的分辨率来显示,我们对图像进行采样,可以使用 Texture2D 的 GetPixel 接口,

public Color GetPixel(int x, int y);

blockRoot = block.transform.parent

完整代码如下:

```
using System.Collections;
using UnityEngine;
public class BlockMgr : MonoBehaviour
{
    /// <summary>
    /// 原因像
    /// <summary>
    public Texture2D texture;
    /// <summary>
    public GameObject block;
    /// <summary>
    public GameObject block;
    /// <summary>
    private Transform blockRoot;
    /// %
    /// <summary>
    private const int WIDTH = 240;
    /// <summary>
    private const int WIDTH = 240;
    /// <summary>
    private const int HEIGHT = 135;
    void Start()
    {
```

```
StartCoroutine(CreateCreenBlocks());
private IEnumerator CreateCreenBlocks()
    int counter = 0;
    var width = texture.width;
    var height = texture.height;
    for (int j = HEIGHT; j \ge 0; --j)
        for (int i = 0; i < WIDTH; ++i)</pre>
           Color color = texture.GetPixel(width * i / WIDTH, height * j / HEIGHT);
            ++counter;
            if (counter > 100)
                counter = 0;
private void CreateBlockItem(int i, int j, Color color)
    var obj = Instantiate(block);
    var trans = obj.transform;
    trans.SetParent(blockRoot, false);
    trans.localPosition = new Vector3(i, j, 0);
    trans.Find("r").GetComponent<MeshRenderer>().material.color = new Color(color.r, 0, 0);
    trans.Find("g").GetComponent<MeshRenderer>().material.color = new Color(0, color.g, 0);
    trans.Find("b").GetComponent<MeshRenderer>().material.color = new Color(0, 0, color.b);
```

脚本挂在 screen 节点上,

'≡ Hi	erarchy	a :	Inspector				ć	3 :
+-	् All स्थि SampleScene	2 :	Screen				Stati	c 🔻
	Main Camera		Tag Untagged		Layer Default			
	▼ 🕞 screen ▼ 🛱 block		🔻 🙏 🛛 Transform				0 ∓	::
	- Qr		Position	X -120	Y -67.5	Z 10	3.1	
	(C) b (C) c		Rotation	X O	Y 0	Z 0		
	Φ a		Scale	X 1	Y 1	Z 1		
			🔻 # 🔽 Block Mgr (\$	Script)			9 ∓	± :
				Block N				
			Texture	201				
			Block	⇔ block ■				
				Add Com	oopont			
				Add Collig	ponent			

六、测试

运行 Unity ,执行效果如下:





七、屏幕后处理

亮度有点暗,我们加上屏幕后处理,点击菜单 Window / Package Manager,

C, Mac & Linux Standalone - Unity 2021.1.9f1c1 Personal <D>

ıt	Wind	dow	Help			
Ce		Pane	els		>	
		Next	Window	Ctrl+Tab		
72		Previ	ious Window	Ctrl+Shift+Tab		



搜索 postprocessing,选中 Post Processing,点击 Install,安装后处理插件。



添加一个后处理层 postprocessing,



把 Main Camera 的 Layer 改为 postprocessing,

E Hierarchy a :	Inspector			а:
+ ▼ Q All ▼ SampleScene :	Main Camera		Static	_î
🎧 Main Camera	Tag MainCamera	 Layer postprocessing 		
► 🗗 screen	🔻 👗 🛛 Transform		0 ‡	:
	Position	X 0 Y 1 Z -	10	
	Rotation	x 0 Y 0 Z 0		
	Scale	X 1 Y 1 Z 1		
	🔻 💶 🖌 Camera		0 ≓	:
	Clear Flags	Solid Color		
	Background			ø
	Culling Mask	Mixed		
	Projection	Perspective		
	FOV Axis	Vertical		
	Field of View	•	- 59.2	
	Physical Camera			
	Clipping Planes	Near 0.3		
		Far 1000		
	Viewport Rect	X 0 Y 0		
		W1 H1		
	Depth	-1		

给 Main Camera 添加 Post-process Layer 组件,并设置 Volume Layer 为 postprocessing 层,

🔻 🧔 🗹 Post-process Layer		0 ‡ :
Volume blending		
Trigger	🙏 Main Camera (Transform)	⊙ This
Volume Layer	postprocessing	
Anti-aliasing		
Mode	No Anti-aliasing	
Stop NaN Propagation	~	
Directly to Camera Target		
Toolkit		
Custom Effect Sorting		

给 Main Camera 添加 Post-process Volume 组件,勾选 Is Global,创建一个 Profile 配置文件,添加 Bloom 泛光效果,如下:

🔻 🍪 🗹 Post-process Volume	Ø ‡ :
ls Global	
Weight	• 1
Priority	0
Profile	⇔Main Camera Profile (Post ⊙ New Clone
Overrides	
🔻 🗹 Bloom	:
	On Off
Bloom	
Intensity	15
Threshold	1
 Soft Knee 	● 1
	65472
	O
Color	HDR 🍼

Fast Mode		
Dirtiness		
Intensity	0	
	Add effect	

最终效果,





七、工程源码

本文工程已上传到 CODE CHINA, 刚兴趣的同学可自行下载学习, 地

址: https://codechina.csdn.net/linxinfa/UnityTVScreenSimulation

注: 我使用的 Unity 版本为 Unity 2020.1.14f1c1 (64-bit)

项目设置 代码 Issue 0 合并请求 0 DevOps Wiki 0 分析 项目成员 Pages + ~ 查找文件 Web IDE 克隆 🗸 master 分支 tags 0 历史 1 🔘 linxinfa 40 seconds ago 推送 Unity TV Screen Simulation 6955048e 1次提交 最后提交 名称 最后更新 Unity TV Screen Simulation Assets 40 seconds ago ProjectSettings Unity TV Screen Simulation 40 seconds ago UserSettings Unity TV Screen Simulation 40 seconds ago 🕒 .vsconfig Unity TV Screen Simulation 40 seconds ago 40 seconds ago Assembly-CSharp.csproj Unity TV Screen Simulation VityTVScreenSimulation.sln Unity TV Screen Simulation 40 seconds ago

林新发 / UnityTVScreenSimulation

八、完毕

好了,今天就到这里吧。

我是林新发: https://blog.csdn.net/linxinfa

原创不易,若转载请注明出处,感谢大家~

喜欢我的可以点赞、关注、收藏,如果有什么技术上的疑问,欢迎留言或私信,我们下期见~