

基于 WebGL 的三维落叶场景仿真

张文娟, 吴 琼, 曹欣然

(西安石油大学 计算机学院, 陕西 西安 710000)

摘 要: 基于 WebGL 的第三方开源库 Three.js, 结合粒子系统, 以 WebStorm 11.0.1 为开发环境, 分析落叶的物理运动过程, 建立了落叶粒子系统三维模型。通过对系统内各个粒子位移的控制, 简化了粒子的消亡过程。结合 Three.js 的简便性, 模拟叶粒子近大远小的透视投影效果, 不再需要在笛卡尔三维坐标系的 Z 轴上引入额外的控制, 只需要使用特定的相机。同时, 通过对粒子系统内粒子数量的控制, 可以模拟出在不同自然条件下叶子落下的场景。使用 Three.js 可以直接在网页上渲染三维场景, 而不需要额外的插件, 同时也较好地利用了硬件加速, 提高了图形渲染速度。实验结果表明, 该方法具有真实的三维仿真效果以及较快的图形渲染速度。

关键词: 粒子系统; 落叶模拟; Three.js; 纹理映射

中图分类号: TP301

文献标识码: A

文章编号: 1673-629X(2018)06-0165-05

doi: 10.3969/j.issn.1673-629X.2018.06.037

Three-dimensional Simulation of Fallen Leaves Based on WebGL

ZHANG Wen-juan, WU Qiong, CAO Xin-ran

(School of Computer Science, Xi'an Shiyou University, Xi'an 710000, China)

Abstract: Based on the third party open source library Three.js of WebGL and used the WebStorm 11.0.1 as a development environment, we establish a three-dimensional model of leaf particle system after the analysis of the physical movement trails of fallen leaves and simplified the extinction process of particles by controlling the displacement of each particle in the system through the combination of particle system. Because of the simplicity of Three.js, the excess control on Z axis in Cartesian three-dimensional coordinate system is no longer needed to simulate the leaf particle's perspective effect that the objects look small in the distance and big on the contrary. Meanwhile, the scenes of falling leaves in different natural environment can be simulated through the quantity control of particle in the system. By using Three.js, the three-dimensional scenes could be rendered in the webpage directly without extra plug-ins, also the graphic rendering speed can be improved due to the better use of the hardware acceleration function. The experiment indicates that this method has realistic effect in three-dimensional simulation and faster speed in graphic rendering.

Key words: particle system; fallen leaves simulation; Three.js; texture mapping

0 引 言

粒子系统是迄今为止模拟不规则物体最成功的算法之一, 目前国内外已有许多成功使用粒子系统与其他技术相结合的案例, 以模拟现实生活中雨雪、火焰、爆炸、喷泉等场景。与之相结合的技术主要有 OpenGL、WebGL, 使用的编程语言也多为 C++ 或者 C#。Latta 等采用 GPU 进行粒子系统的模拟, 能实时处理超过 100 万个粒子, 使得大规模粒子模拟成为可能, 也使越来越多的人开始关注并研究基于 GPU 的粒子系统动画模拟。李晓萍对基于 GPU 的粒子系统进行了深入研究, 利用基于 GPU 的粒子系统模拟了喷泉, 并

在 CPU 中处理了喷泉粒子的产生和消亡, 在 GPU 中更新粒子的属性, 显示粒子^[1]。袁霞等对粒子系统的方法及应用做了较为系统详尽的研究, 并利用 3D MAX 模拟了水泡上升的自然现象^[2]。罗维佳等使用 C++ 语言, 将雨粒子产生区域定义为一个视图体顶部的外接长方体, 用像素点和直线作为对雨粒子的形状、降落过程的重力作用的模拟, 完成了对降雨过程的仿真^[3]。徐利明等使用将粒子系统与 OpenGL 相结合的方式, 把模拟雨、雪的粒子在一个新的视口中视线有效区域内进行绘制, 然后与原视口中的场景一起显示于窗口中, 实现了对雨雪效果的模拟^[4]。

收稿日期: 2017-06-06

修回日期: 2017-10-09

网络出版时间: 2018-02-24

基金项目: 陕西省科技攻关项目 (2016GY132)

作者简介: 张文娟 (1994-), 女, 硕士研究生, 研究方向为计算机仿真、人工智能。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20180224.1516.036.html>

随着互联网技术的不断发展,网页制作技术起到了越来越重要的作用,在网页制作的过程中,3D 技术具有十分明显的优势,它可以使用户对所需浏览的内容的感知更加真实。目前实现 3D 网页的主要技术有 Flash3D、java3D、Unity3D 等等^[5]。但这些技术在网页上的实现均需要安装特定的插件,而插件的安装则会对网页的稳定性和跨平台性有所影响。其次,很多技术只针对特定的行业,不具备较强的通用性,例如 Unity3D 只适合游戏开发。而 WebGL 的产生则解决了这些问题^[6]。

WebGL 是一项可以在浏览器中绘制、显示三维计算图形并与之交互的技术。曾经只有高端的计算机或专门的游戏终端才能渲染三维图形,因为这需要大量复杂的编程才能实现^[7]。然而随着个人计算机以及浏览器性能的提高,使用网页技术渲染三维图形也成为可能。和 OpenGL 和 Direct3D 不同,WebGL 程序可存在于网页中并在浏览器中执行,而不必安装任何其他的插件和库^[8]。然而,直接使用 WebGL 编程是十分复杂的,需要了解 WebGL 的内部细节,学习复杂的着色器语法^[8],因此文中使用一个基于 WebGL 的开源框架 Three.js。

Three.js 基于 WebGL,封装了底层的图形接口,使得使用者不需要掌握冗杂的图形学知识,就能用简单的代码完成三维场景的渲染。理论上,更高的封装程度往往意味着灵活性的牺牲,但 Three.js 在这方面做得很好,几乎不存在 WebGL 支持而 Three.js 不能实现的情况。目前,国内主要将这一技术应用于物体的真实化展示。如高辰飞将其应用于海洋样品的三维可视化研究,实现了对海洋样品的虚拟可视化^[9]。

综上,在研究粒子系统的基础上,利用 WebStorm 开发工具在 Windows 平台下进行降雨场景的模拟,建立粒子系统和雨滴系统模型。

1 粒子系统的基本原理

粒子系统由 Reeves W T 于 1983 年提出,并成功模拟了灰尘、喷泉等效果。其基本思想是利用形状简单的微小粒子作为元粒子来描述不规则的模糊物体。粒子系统并不是一个简单的静态系统,而是动态变化的。每个元粒子具有大小、形状、位置、颜色、速度以及生命周期等属性^[10-11]。与其他描述不规则物体的方法相比,粒子系统具有三个主要特点:

(1) 对物体的描述是通过一组定义在空间的原始粒子,而不是利用原始的具有边界的面元^[12]。

(2) 粒子系统不是一个静态系统,每个粒子的属性均随时间的变化而变化。

(3) 粒子系统所描述的物体不是预先定义好的,

其相关属性均可使用随机过程来描述^[12]。

从应用的角度可将粒子系统分为三类,分别是随机粒子系统、结构化粒子系统、方向粒子系统。随机粒子系统通过可控制的随机过程控制粒子属性的变化,用来生成灰尘、烟、爆炸等场景图像^[13]。结构化粒子系统可以用来模拟具有一定结构的现象或物体,如树、草等。方向粒子系统则考虑了粒子间的相互影响,粒子除具有位置和速度等动态属性之外,还应该具有方向属性,用来模拟可变物体。由于落叶场景没有固定的结构,在模拟时亦不需要考虑叶粒子之间的相互作用,因此适合采用随机粒子系统进行模拟。

在随机粒子系统中,随着时间的变化,粒子的运动状态及形式不断发生变化,元粒子状态的改变引起粒子群整体的变化。在这一过程中,旧粒子不断死亡,新粒子不断产生,粒子生命周期及其变化都可以使用随机过程进行模拟^[3]。通常,粒子系统实现的主要步骤如下:

- (1) 分析粒子的静态特性,定义新粒子。
- (2) 引入随机函数,建立粒子属性动态变化特征。
- (3) 根据粒子动态特性更新粒子属性^[14]。
- (4) 删除生命周期结束的粒子。
- (5) 渲染产生新粒子。

2 Three.js 搭建三维空间

随着 HTML5 标准的颁布以及主流浏览器功能的日益强大,直接在浏览器中展示三维图形和动画已经变得越来越容易,也越来越受关注。但是,三维图形和动画本身就比较复杂,不仅需要具备丰富的数学、图形学等方面的基础知识,还需要了解材质、贴图等各种创建三维场景所必须的要素。除此之外,直接使用 WebGL 在浏览器中创建动画也十分烦琐^[15]。

Three.js 的出现解决了这个矛盾,它将 WebGL 的强大功能融汇在其中,同时语法简单、易于使用,即使使用者对 WebGL 的底层细节并不清楚,也能借助 Three.js 创建出绚丽多姿的三维场景和动画。Three.js 支持多种渲染器进行场景渲染并提供了点、线、面、向量、矩阵等创建三维物体所必须的要素,同时可以使用十分简单的语法将照相机(镜头)、光线、物体等对象添加到场景当中。

用 Three.js 创建三维物体并在网页上显示的主要步骤如下:

(1) 场景设置:场景实质上就是一个三维空间,后续创建的物体均需要添加到场景中。在 Three.js 中,创建三维场景的代码为:var scene = new THREE.Scene()。场景可以理解成一个巨大的容器,在程序最开始执行的时候完成场景的实例化,然后通过 add

() 方法向场景中添加对象。

(2) 相机设置: Three.js 中有两种不同的相机, 分别是正投影相机和透视相机。透视相机可以模拟出最自然的视图, 在同一场景中如果选用透视相机, 则距离相机越远的物体会被渲染的越小, 而如果采用正投影相机, 则同一场景中物体距离相机的远近不会影响该物体的大小^[13]。在进行自然现象的模拟中通常使用透视投影, 因为这更接近人眼的观察效果, 图 1 分别表示同一场景在两种相机下的显示情形。



图 1 透视相机(左)与正投影相机(右)

可以看到, 透视相机照射出的物体明显具有近大远小的效果。而在传统的模拟中, 需要对 Z 轴进行一系列控制, 才能渲染出近大远小的效果。如周强等的基于粒子系统的三维降雪场景仿真中, 在 Z 轴上需要对雪花进行深浅的分类才能模拟出此效果^[13]。而使用 Three.js 只需选择合适的相机并对所渲染物体的参数设置合理, 就可以很容易地模拟出这种效果, 而不需要添加函数控制或者引入其他粒子。

(3) 光源设置: Three.js 中提供了多种光源, 每种光源都有特定的用途, 一个场景中可以设置多个光源, 基本上都是将环境光源和其他光源组合起来, 根据三维场景中显示物体的不同, 可以选择不同的光源组合。

(4) 物体模型的设置: Three.js 本身提供了大量的几何体, 例如球、长方体等。也可选择合适的库来加载其他格式的三维模型, 如 obj、json 等格式的模型, 通过导入外部三维模型, 可以达到更加真实的仿真效果。

(5) 渲染器设置: 渲染器的工作是将三维场景中的物体映射到二维平面上^[16], 即映射到电脑的显示器上。在场景中添加了物体、设置好相机之后, 就可以调用渲染器的渲染函数来渲染整个场景。

通过以上五个步骤, 就可以搭建一个简单的三维场景。图 2 为 Web 上显示三维场景的基本结构模块。

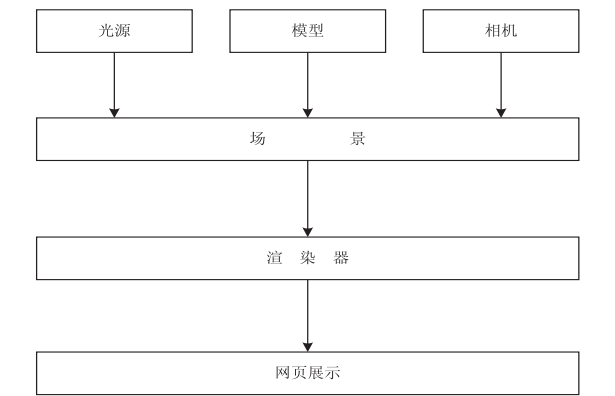


图 2 三维场景的构成

3 落叶粒子系统模型

Three.js 中使用 ParticleBasicMaterial(基础粒子材质)创建和设计粒子, 使用 ParticleSystem(粒子系统)创建一个粒子集合。通常, 使用 Three.js 创建粒子系统的步骤如下:

- (1) 搭建三维场景;
- (2) 定义一个三维几何体;
- (3) 对所需创建的粒子系统中的粒子定义材质;
- (4) 利用双层循环为每个粒子创建一个定点, 并利用 push 方法将其添加到几何体中;
- (5) 创建 ParticleSystem 对象, 将三维几何体和材质进行融合;
- (6) 利用 scene.add() 方法将 ParticleSystem 添加到场景中。

图 3 是一个简单的粒子系统。在该例中, 首先创建了一个 THREE.Geometry 对象, 之后利用循环语句在随机的位置上创建粒子, 并将其添加到几何体中, 最后再使用 ParticleSystem 类来显示粒子。ParticleSystem 类的构造函数接收两个参数, 一个是几何体、一个是材质。材质用来给粒子上色和添加纹理, 而几何体用来指定将粒子放在哪里, 每个顶点, 即定义几何体的各个点, 将会以粒子的形态展示出来。

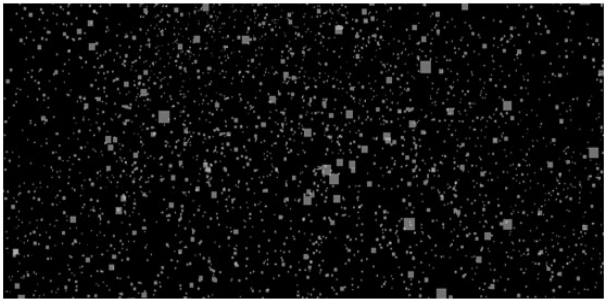


图 3 简单粒子系统

3.1 落叶粒子属性

对叶粒子降落的运动过程进行分析, 确定叶粒子的属性。叶子在下降过程中, 会受到风力、重力等因素的影响, 在视觉上并不是沿直线降落, 同时, 加速度和速度也在不断变化。在 X 轴上, 落叶受风力影响位置左右飘动, 在 Y 轴上落叶滴受空气浮力以及重力的作用做下降运动且各时刻速度均不相同。利用 WebGL 的第三方库 Three.js 模拟落叶效果与传统模拟方法的主要区别在于不需要对 Z 轴进行额外的控制。在传统方法中, 要模拟出远处的物体看上去比近处的物体小一些的效果, 需要对 Z 轴上粒子的深浅程度进行控制。而利用 Three.js 除了可以使用透视相机来达到这一目的外, 还可以在创建粒子时对 sizeAnnotation 属性进行设置, 将该属性设置为 false 时, 系统中的粒子无论距相机多远, 都将具有相同的尺寸, 而如果设置为 true,

则粒子的大小取决于其距离相机的远近。通过相机与 sizeAnnotation 属性的配合使用,就能很好地模拟出远处的物体看上去比近处小的效果,而不需要引入额外的控制,这就简化了实现步骤,并不影响模拟效果的真实性。当雨粒子下降到地面时,重新对其设置位置属性,如此循环模拟降雨场景。

3.2 落叶粒子系统模型初始化

由于在实验中,对相机所能渲染的范围设置为 1 000,即屏幕上的物体均可被渲染出来。因此,在叶子飘落的过程中,可以将显示器顶部作为粒子的初始位置。

在此次实验中,采用加载外部图片的方法来格式化粒子系统中的粒子。Three.js 中可以使用 THREE.ImageUtils.loadTexture() 方法加载外部图片,使用外部图片来格式化粒子的优点是使得所模拟物体更加形象逼真。文中采用 128 * 128 的 png 图片来格式化落叶粒子系统,如图 4 所示。



图 4 落叶粒子纹理

加载完外部图形之后,需要将图片应用于粒子的材质中,Three.js 中实现这一效果的主要代码如下:

```
var material=new THREE. ParticleBasicMaterial( {
size:10,
transparent:true,
opacity:0.3,
map:texture,
blending:THREE. AdditiveBlending,
sizeAttenuation:true,
color:0xffffffff
});
```

其中,map 属性就指向加载的外部图片。

在完成粒子的格式化之后,需要对粒子运动时的速度进行控制。为了简化落叶粒子运动过程,提高系统的实时性,对每个落叶粒子的速度都使用随机控制函数,使得每个粒子的速度状态都不一致,通过大量速度状态不一致的落叶叶粒子,营造出落叶的效果。第 i 个粒子的速度描述公式如下:

```
particle. velocityX=( Math. random() -0.5)/800
particle. velocityY=0.1 + Math. random()/1 000;
velocityX 用来定义粒子以多快的速度横向运动,
velocityY 是定义粒子以多快的速度下降。至此,粒子的
```

初始属性设置完毕。通过 for 循环中参数的设置,可以产生多个雨粒子,通过改变这个参数,可以实现模拟下雨量。

3.3 落叶粒子运动更新

传统的模拟方法中,需要对超出生命周期的粒子进行删除,增加了复杂度。在此次实验中,通过对粒子位置的控制以避免粒子的删除,其主要实现代码如下:

```
If( v. y<=0) v. y=80;
If( v. x <= -50 || v. x>= 50) v. velocityX= v. velocityX * -1;
```

这两行代码保证了粒子处在被定义的范围之内,如果 y 方向的位置低于零,就将落叶粒子放回顶部,如果 x 方向的位置超出边界,将横向运动速度去反,让落叶粒子反弹。这样,系统中粒子的数目其实是固定的,只是通过对系统内粒子位置的调整来模拟出不断有新的粒子产生和旧粒子消亡的过程。这样不仅简化了实现方法,也节约了渲染时间。

4 实验结果

实验的硬件环境为主频 2.5 GHz 的 i5CPU,显卡为 NVIDIA GeForce GT 740M,运行内存为 4G 的 PC 机,选用 WebStorm 11.0.1 开发工具在 Windows 平台下,利用 WebGL 的第三方库 Three.js 进行模拟。实验结果如图 5 和图 6 所示。

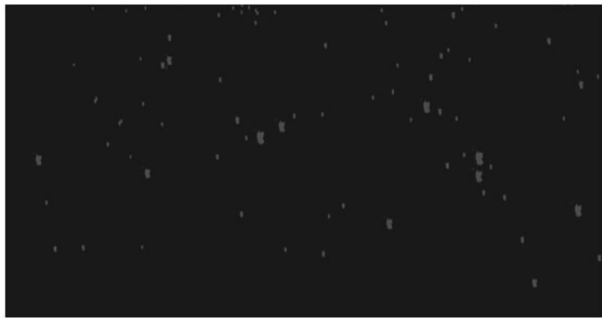


图 5 系统内粒子数量为 1 000 时落叶的模拟



图 6 系统内粒子数量为 2 000 时落叶的模拟

5 结束语

在传统的粒子系统算法的基础上,通过对粒子系统内每个粒子的位置状态的控制,简化了粒子消亡过

程的控制。结合 WebGL 的第三方开源库 Three.js,避免了为实现粒子近大远小的视觉效果而需要对 Z 轴上添加控制因子。结合这两种优势,大大简化了实现方法。同时,通过对粒子系统内粒子数量的控制,可以根据需要模拟出落叶多少的效果,在简化实现方法的同时也具有逼真的三维仿真效果。

文中代码具有良好的扩展性,在此基础上,可以加载其他外部图片作为粒子纹理,模拟不同的自然现象,例如降雨、烟花等。也可以对粒子运动速度进行控制,模拟不同天气情况下的降雨效果。

参考文献:

[1] 李晓萍. 基于 GPU 的粒子系统的研究与应用[D]. 长春: 吉林大学,2009.

[2] 袁霞,张玉琢. 粒子系统方法及其应用[J]. 云南师范大学学报:自然科学版,2003,23(3):14-16.

[3] 罗维佳,都金康,谢顺平. 基于粒子系统的三维场地降雨实时模拟[J]. 中国图象图形学报,2004,9(4):495-500.

[4] 徐利明,姜昱明. 基于粒子系统与 OpenGL 的实时雨雪模拟[J]. 计算机仿真,2005,22(7):242-245.

[5] 荣艳冬. 基于 WebGL 的 3D 技术在网页中的运用[J]. 信息安全与技术,2015(8):90-92.

[6] XU Zhao,ZHANG Yang,XU Xiayan. 3D visualization for building information models based upon IFC and WebGL integration[J]. Multimedia Tools & Applications, 2016, 75(24):17421-17441.

(上接第 164 页)

[2] 刘琦. 道路交通安全管理研究[D]. 青岛:中国海洋大学,2012.

[3] WHEELER T A, KOCHENDERFER M J. Factor graph scene distributions for automotive safety analysis[C]//International conference on intelligent transportation systems. Rio de Janeiro, Brazil:IEEE,2016:1035-1040.

[4] 梁春疆,段发阶,杨毅,等. 车辆外廓尺寸计算机视觉动态测量[J]. 光电工程,2016,43(1):42-48.

[5] 李怀泽,沈会良,程岳. 基于旋转多视角深度配准的三维重建方法[J]. 计算机应用,2012,32(12):3365-3368.

[6] 孔颖乔,赵健康,夏轩. 基于立体视觉的高精度标定与测量方法[J]. 计算机应用,2017,37(6):1798-1802.

[7] 郭政业,胡雯蔷,朱李瑾. 基于眼球重建界面直线模型的双目视线跟踪算法[J]. 计算机应用研究,2016,33(4):1249-1252.

[8] 姜庆昌. 汽车轮廓尺寸测量机的研究[D]. 哈尔滨:哈尔滨工业大学,2006.

[9] 苏建,翟乃斌,刘玉梅,等. 汽车整车尺寸机器视觉测量系统的研究[J]. 公路交通科技,2007,24(4):145-149.

[10] LI Shuaijun, JIANG Xinyu, QIAN Huihuan. Vehicle 3-dimension measurement by monocular camera based on license plate[C]//International conference on robotics and biomim-

[7] 魏云申. 基于 WebGL 的全景 3D 漫游系统的设计与实现[D]. 南京:南京大学,2016.

[8] 顿儒源. 基于 WebGL 的织物三维展示系统[D]. 杭州:浙江大学,2016.

[9] 高辰飞. 基于 WebGL 的海洋样品三维可视化的研究[D]. 青岛:中国海洋大学,2014.

[10] REEVES W T. Particle systems—a technique for modeling a class of fuzzy objects [C]//Seminal graphics. [s. l.]: ACM,1998:203-220.

[11] REEVES W T,BLAU R. Approximate and probabilistic algorithms for shading and rendering structured particle systems[C]//Proceedings of the 12th annual conference on computer graphics and interactive techniques. [s. l.]:ACM,1985:313-322.

[12] 王润杰,田景全,倪政国. 基于粒子系统的实时雨雪模拟[J]. 系统仿真学报,2003,15(4):495-496.

[13] 周强,汪继文. 基于粒子系统的三维降雪场景仿真[J]. 计算机技术与发展,2017,27(1):130-133.

[14] DANCHILLA B. Three.js framework [M]//Beginning WebGL for HTML5. [s. l.]:[s. n.],2012:173-203.

[15] 朱丽萍,李洪奇,杜萌萌,等. 基于 WebGL 的三维 WebGIS 场景实现[J]. 计算机工程与设计,2014,35(10):3645-3650.

[16] HUANG Youliang,ZHOU Mingquan. Design and development of the virtual acupuncture training using WebGL[J]. Advanced Materials Research,2013,756-759:2076-2080.

etics. Qingdao,China:IEEE,2016:800-806.

[11] SURAL S,QIAN Gang,PRAMANIK S. Segmentation and histogram generation using the HSV color space for image retrieval [C]//International conference on image processing. Rochester,NY,USA:IEEE,2002:589-592.

[12] BENNETT K P. Combining support vector and mathematical programming methods for classification [M]. Cambridge, MA:MIT Press,1999:307-326.

[13] 范伊红,彭海云,张元. 基于 SVM 的车型识别系统的设计与实现[J]. 微计算机信息,2007,23(3-1):296-297.

[14] WEI S,LAI Shanghong. Fast template matching based on normalized cross correlation with adaptive multilevel winner update[J]. IEEE Transactions on Image Processing,2008,17(11):2227-2235.

[15] DAI Huadong,WANG Yang. A new method for detecting rectangles and triangles[C]//2015 IEEE advanced information technology, electronic and automation control conference. Chongqing,China:IEEE,2015:321-327.

[16] LIU Dong,WANG Yongtao,TANG Zhi,et al. A robust circle detection algorithm based on top-down least-square fitting analysis[J]. Computers & Electrical Engineering,2014,40(4):1415-1428.