计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(12):78−83 [doi: 10.15888/j.cnki.csa.006116]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 WebGL 的大型团体舞虚拟编排系统①

童基均[1], 邹永胜[1], 余明权[2]

[1](浙江理工大学 信息学院, 杭州 310018)
[2](浙江理工大学 体育教研部, 杭州 310018)

摘　要: 随着中国演艺市场规模的不断发展, 演出市场的演出场次和观众人数不断增加, 对演出排练的质量要求逐渐提高. 开发一个虚拟编排系统具有重要的价值和意义. 本文研究开发一个适用于中大规模演出的虚拟编排系统, 通过对基于 WebGL 的虚拟编排系统关键技术的研究, 提出了模型文件压缩、预加载、内嵌 server、优化 v8 引擎等方法, 使得大规模虚拟排练系统得以流畅的运行, 同时利用 WebGL 技术实现三维模型的创建、渲染, 使得用户能够直接在本系统中完成大型排练的所有过程, 具有良好的应用前景.

关键词: 虚拟编排; Canvas; WebGL; V8 引擎优化

引用格式: 童基均,邹永胜,余明权.基于 WebGL 的大型团体舞虚拟编排系统.计算机系统应用,2017,26(12):78–83. http://www.c-s-a.org.cn/1003-3254/6116.html

Virtual Choreography System of Group Dancing Based on WebGL

TONG Ji-Jun[1], ZOU Yong-Sheng[1], YU Ming-Quan[2]

[1](School of Informatics, Zhejiang Sci-Tech University, Hangzhou 310018, China)
[2](Department of Physical Education and Teaching, Zhejiang Sci-Tech University, Hangzhou 310018, China)

Abstract: With the booming of performing arts in China, which is obviously embodied in the increasing number of performances and audiences, and the requirement of performance quality is increasing. Therefore, a virtual choreography system becomes particularly important with great value and significance. This paper aims to explore and develop a virtual choreography system for medium-scale and large-scale performances. By doing research on the key technology of the virtual choreography system based on WebGL, we put forward a series of methods including file compression, preloading, embedded server and v8 engine optimization, which can help assure the smooth operation of the system. Meanwhile, we adopt WebGL to achieve the creation and rendering of three-dimensional models so that users can directly complete the whole process of large-scale rehearsals in the system. It has great prospect.

Key words: virtual choreography; Canvas; WebGL; V8 engine optimization

团体操是一种体育和艺术高度结合、以体操动作为主的群众性体育表演项目[1]. 它由几十人、几百人以至成千上万人, 在大的场 (馆) 中通过以体操为主体的各色各样的体育、文艺形式和队形变化、图案造型, 配以音乐、道具、服装, 以至背景、场景 (舞台) 和灯光等艺术装饰所构成的体育艺术性的集体表演项目[2]. 团体操设计和排练是一件非常烦琐、耗时的工作. 虚拟现实技术在体育仿真中具有很好的应用前景[3], 将虚拟人群仿真技术应用于团体操设计和演练, 可以提高设计和排练质量, 提高设计人员和排练人员的工作效率. 团体操编排和演练仿真涉及到大量的虚拟人群运动、动作及队列编排, 其关键技术为大规模场景的实时绘制、人物的建模、路径输入等.

目前关于虚拟人群仿真[4,5]主要是通过一些软件或

① 基金项目: 浙江省重点研发计划 (2015C03023)
收稿时间: 2017-03-13; 修改时间: 2017-04-05; 采用时间: 2017-04-13

者插件来制作, 比如美国的 Vital Idea 公司的 Crowd; Discreet 3D Max 的插件 Crowdit; Avid 的 SoftImage Behavior. 但是这些软件均有一些不足, 无法串联整个团体舞的编排过程, 仅仅能完成其中的一个步骤. 因此本文从编排的数据输入出发, 串联了整个虚拟编排的过程, 极大地提高整个编排的效率.

WebGL[6](Web graphics library) 是专门为 Web 设计的底层图形编程 API, 将浏览器和 GPU 紧密的联系在了一起. 这样开发人员即可借助 GPU 在浏览器里更流畅地展示 3D 场景和模型, 创建复杂的 3D 展示系统.

针对虚拟排练用户群体数量大、排练路径数据多、实时响应性要求高的需求, 目前国内关于 WebGL 相关的产品较少, 资源相对匮乏. 本文采用 WebGL 将传统的大型排练与计算机相结合[7], 极大地提高了排练的效率, 它能将演员的走位效果实时的展现出来, 更加方便导演及时调整演员的走位信息, 使得导演能够充分发挥自己的想象力, 对于自己的编排有一个直观的认识, 具有很好的研究意义和应用前景.

1 系统简介

本系统主要主要借助 HTML5[8]中的 Canvas 元素, 主要分为 2D 路径输入系统以及 3D 虚拟展示系统. 2D 路径输入系统主要职责就是收集所有演员在所有帧的数据, 然后导出至指定格式的文件, 将文件传递到 3D 平台, 3D 平台会自动生成对应的 3D 模型以及动画, 此外用户还可以修改表演场景的舞台、天空、特效、以及音乐.

1.1 整体介绍

系统会根据用户输入的配置信息自动生成 2D 舞台, 随后导演就可以放置关键帧上所有的演员, 当所有关键帧的数据全部输入完毕后, 即可形成 2D 平台的动画. 此外, 用户如果不满意当前设计方案, 可以重新开始设计路线.

关键用户数据生成后, 3D 平台根据这些数据完成 3D 人物模型的建模, 生成路径动画. 此外用户还能够添加上灯光、烟雾、地面、天空、音乐等控制, 增强用户对自己设计方案的感官认识. 图 1 为虚拟编排系统总体框图.

1.2 2D 路径输入系统

本子系统主要负责收集导演的灵感, 采用技术为 HTML5 中 Canvas 元素的 2D context. 系统采用棋盘的方式模拟表演的场景, 采用带颜色的正方形方格来模拟模型人物. 当用户输入场景的宽度、高度、人物的个数等基本信息后, 本子系统会动态生成一张网格, 然后用户就可以在此背景上进行布局, 此外本系统还拥有较好的错误提示, 当用户放置的人数超过预设人数或者是在移动人物位置过程中产生重叠, 会提示用户输入有误请重新调整.

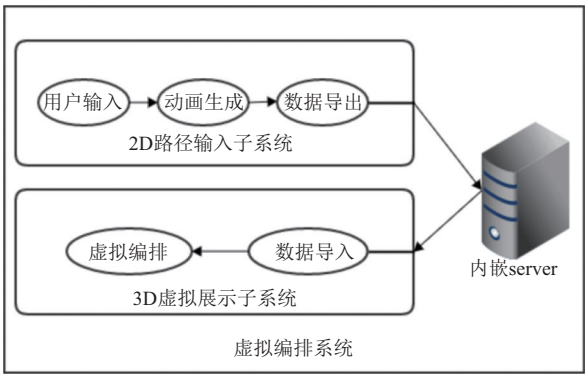


图 1 虚拟编排系统整体框架图

1.3 3D 虚拟展示系统

本系统负责将前面的 2D 平台数据转化为 3D 人物模型, 并进行 3D 动画展示. 采用的技术是 HTML5 中 Canvas 元素的 3D context-WebGL. 每次只绘制一次人物模型, 然后再从此基础上克隆出指定数量的人物模型, 同时在加载过程中采用懒加载技术, 这样有效的提高了整体的加载效率. 对于灯光、地面、天空、音乐等非核心部件的加载则采用按需加载, 只有在用户打开相应的开关时才会去加载相应的素材. 这种处理逻辑能够有效的将原本比较大的模型素材分解为一个个小步骤, 有效的提高了系统整体运行的流畅性.

1.4 系统模块划分

本系统主要分为如下模块: 2D 场景绘制、2D 人物绘制、2D 事件处理模块、2D 动画生成、2D 错误提示、2D 数据导出、3D 模型加载、3D 数据导入、3D 模型克隆、3D 事件处理模块[9]、3D 定时器同步、3D 背景音乐控制. 各个模块之间相互独立, 但是又保持一定的联系性. 系统各模块的的关系图如图 2 所示.

2 实现关键技术

2.1 2D 绘制

2D 绘制主要分为表演舞台绘制、动画背景绘制以及人物模型绘制. 整个 2D 场景绘制采用分层思想. 整体的绘制分为 stage、layer、group、shape. 整个

2D 场景有且仅有一个 stage, 其下可以有若干 layer, 每个 layer 下可以有若干个 layer 或者是若干个 group, 每个 group 下又可以包含若干个 group 或者是 shape. 这种绘制思想极大地提高了场景绘制的可伸缩性、可扩展性, 有效的提高了整个系统的稳定性. 图 3 所示为 2D 绘制的结构层次图.

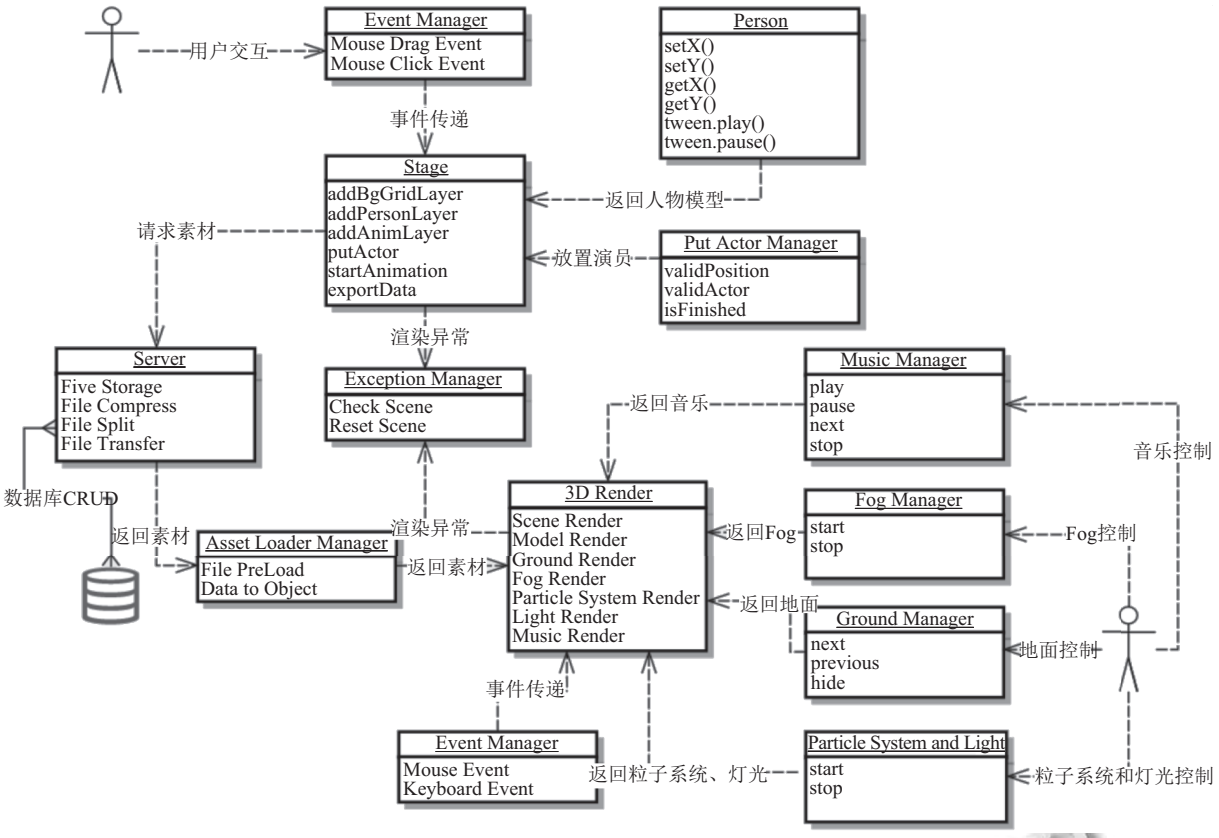


图 2 系统各模块的关系图

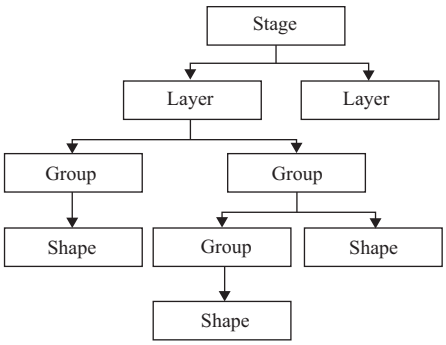


图 3 2D 绘制的结构层次图

由于表演场景的大小不固定, 所以这里增加了一个收集场景配置信息 (场景大小以及演员人数) 的页面. 在获取数据后, 即可根据此参数动态的绘制背景. 背景的绘制全部集中在背景层, 它是由一条条水平线以及竖直线组成的网格. 每条线段是由起点、线长、线宽、线条颜色组成, 人物绘制采用一个正方形格子代表每一个演员, 每一个演员都是一个类, 里面封装了一些常用的信息, 比如演员的坐标、演员的 id、演员的背景等. 绘制的时候将每个演员当作一个 group, 然后往此 group 中添加一个指定颜色的正方形格子以及该演员的 id, 两者组合代表了一个有效的演员. 在放置的时候, 需要判断此处是否已经存在演员, 然后才将演员 group 添加到演员 layer 上.

算法描述:

```
Function drawAreaAndPutActor() {
    for(var i=0; i<numberOfGrid; i++) {
    var line = new Line(informationOfLine);
        bgGridLayer.add(line);
    }
    if(isExist == false) {
        currentActorID++;
        actor = new Person(informationOfPerson);
```

```
            actor.addToLayer(personLayer);
        }
    }
```

2.2 2D 动画框架设计

本系统采用的是线性动画, 首先在动画层上绘制出所有的演员, 在用户点击开始动画之后将所有的演员添加到动画层上, 同时将之前在演员层的所有演员清空.

动画的设计主要有两个关键点: 一是如何设计一帧动画, 二是如何监听一帧动画完成. 这里采用线性动画以及异步事件监听机制来解决这两个问题.

线性动画实现的关键就是设计一个关键的动画类, 去区分以及实现所有演员的动画. 因此必须把所有的动画操作全部封装到一个类中, 该类中含有动画的所有方法以及属性, 这样保证了在初始化每个演员实例的同时保存了该实例自己的所有状态.

此外采用基于事件的监听机制来获得动画完成的通知, 当一帧动画完成后会往事件循环池中发送一个通知, 下一帧动画开始执行的时机也就是上一帧动画完成.

算法描述:

```
Function tweenAnimation() {
    actorArr[actorId].addToLayer(animLayer);
    actorArr[actorId].tween(tweenParams);
    if(checkFinish()){
        sendSignal();
        nextAnimation();
    }
}
```

2.3 3D 模型克隆

由于一个人物三维模型的数据量较大[10], 内部包含的信息有 Mesh、Skeleton、Material、Animation Frame 等, 浏览器加载一个人物模型的速度取决于加载的网速, 在加载的过程中可能会出现很多种情况, 例如, 加载到一半的时候中断了, 亦或是成功加载但是加载速度过慢、消耗时间过长. 本文从数据源、数据传输、数据加载、模型转换多个方向对这种不友善的用户交互体验进行优化, 有效的调度模型数据, 提高渲染效率, 保证人机交互的实时性.

本系统采用 3DS MAX 制作 3D 模型, 在不影响整体舞台、人物质量的情况下尽量减少整个模型文件的大小即采用较少 3D 模型的面数以及分段数. 同时在导出的时候, 尽量减少无关、重复信息的出现, 例如灯光、plane 等信息.

对于数据传输过程的优化可以从数据的请求、传输、响应三个阶段考虑. 在数据请求方面采用 AJAX (Asynchronous Javascript and XML) 技术来完成数据的请求, 它以异步的方式发送请求, 而不会阻塞住整个界面的渲染. 在数据传输方面, 为了彻底解决整个模型的加载受限于公网速度的问题, 采用的方案是在整个系统中内嵌一个小型的服务器, 其主要功能就是转发文件. 这样就将模型文件的传输由外部公网转化为了系统内部的局域网.

由于一个普通的人物模型文件依据人物细节的多少普遍在数十兆以上, 这给整个系统带来了很大的不确定性. 采用的解决方案是预加载技术, 即优先加载一些数据量小的基础场景提供给用户进行交互, 然后在不阻塞主线程渲染 UI 的前提下, 加载主模型数据文件, 这样就在用户不知情的情况下为用户加载了关键资源, 有效的提高了人机交互的实时性.

本系统从多个角度完成对 3D 模型加载的优化, 并取得了较好的效果.

2.4 3D 动画框架设计

这里采用多定时器同步技术来实现 3D 模型动画的控制.

首先初始化中央调度定时器、位置定时器、动画播放定时器、旋转定时器、旋转调整定时器. 各个定时器线程之间采取消息机制来实现数据的同步. 中央调度定时器主要负责统筹调度各个定时任务的执行时机, 使得每个定时器在指定的时间序列、指定的误差范围内实现切换. 位置定时器主要负责整个场景内 3D 模型的位置切换, 保证每个时间序列内每个模型都到达指定的位置. 动画播放定时器是保证每个 3D 人物模型都能够播放各自的指定动画. 旋转定时器则主要负责每个关键场景下人物模型的转向, 保证每个演员的角度都是正确的. 旋转调整定时器负责在演员完成转向后转向的调整.

这其中最主要的就是如何保证整个时间线流转的精准性. JavaScript 原生的事件机制可能会抛弃同一事件源的多次触发事件中的一个或多个. 如图 4 所示.

因此一个高效的中央调度定时器来掌控整个时间线的调度是十分必要的. 中央定时器的事件粒度设置为最大容忍误差, 当优先级高的事件到达后, 哪怕当前事件处理程序未执行完也会被中央调度定时器强制切换到另外一个事件处理函数执行一会, 然后再返回继续执行. 这种机制的存在保证了不会有事件会被丢弃, 优先级高

的能够得到执行, 优先级低的也能在一定的时间范围内得到执行, 极大地提高了整个系统的人机交互的实时性.

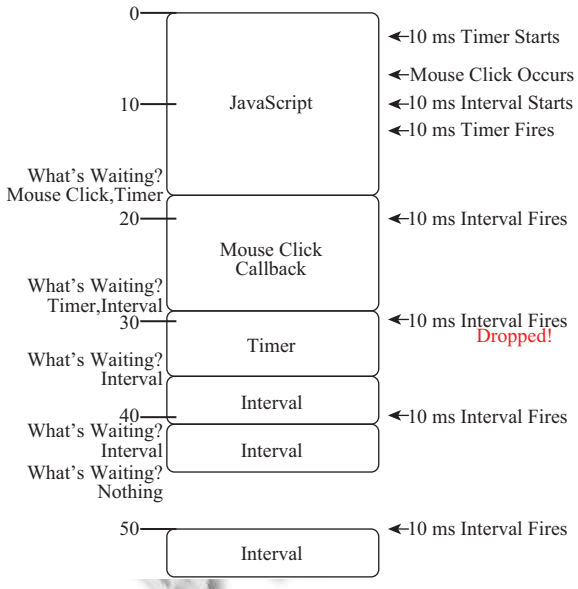


图 4 JavaScript 事件丢弃

2.5 V8 engine 的优化

V8[11]是一个由 Google 开发的开源 JavaScript 引擎, 被用于 Google Chrome 中. V8 在运行之前将 JavaScript 编译成了机器码, 而非解释执行的字节码, 这样能更进一步的提升性能. 此外, V8 还使用了如内联缓存 (inline caching) 等方法来提高性能. 在这些优秀的特性支持下, JavaScript 程序的编译速度速度媲美二进制.

尽管 V8 引擎拥有许多优秀的特性, 但在编写本系统时仍针对 V8 的一些特性进行优化. 例如保证初始化所有对象、同一方向初始化对象成员、避免为数组分配过大的内存等.

整个系统的编写的过程中, 从编译器的层面, 尝试优化代码, 并取得了良好的效果, 整个系统在人机交互的实时性方面有了较大的提高.

3 实验结果

3.1 实验环境

本实验采用的操作系统是 Windows 7; CPU 为 Intel®Core® i7-2670QM 处理器, 主频 2.2 GHZ, 内存 8 G, 显卡 Intel®HD Graphics 1 G; 浏览器为 Chrome 56; 网络带宽为 4M, 实际下载速度为 512 Kbps.

3.2 渲染结果

在上面的章节中, 从数据的产生、数据的存储、数据的传输、模型的渲染、素材的加载、代码执行引擎的优化等角度对本系统做了一些优化. 接下来就来测试下这些优化所带来的性能上的提高. 依次测试开启以及关闭这些优化所带来的加载素材时间以及帧率的差异.

从表 1 中看出, 本系统内置的 server 拥有较好的性能, 能够大幅较少数据在传输过程中的消耗时间, 极大的提高了人机交互的实时性.

从表 2 中可以看出, 本系统采用的模型克隆技术对于大型团体舞这种表演规模具有较好的实用价值, 能够有效的减少对于物理机器内存的依赖, 有利于提高整体运行的稳定性. 开启模型的克隆有效的提高帧率以及减少了内存的消耗, 对于内存以及帧率都有着较好的支持 (这里认为最低的接受标准为 12 fps).

表 1 加载相同模型带来的加载时间差异

序号	素材大小(byte)	关闭内部server(ms)	开启内部server(ms)	时间比
1	1,239,580	3026	70.21	43.099
2	4,855,019	11853	74.47	159.164
3	23,917,608	58392	268.10	217.799
4	47,835,216	116785	343.49	339.992
5	73,589,901	179662	565.71	317.586
		平均值		216.731

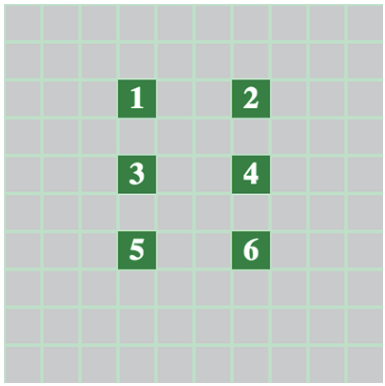
表 2 演员的数量对帧率影响

序号	演员数量	开启模型克隆(fps)	开启模型克隆CPU/GPU内存(MB)	关闭模型克隆(fps)	关闭模型克隆CPU/GPU内存
1	20	60	110/80	58	459 MB/143 MB
2	40	58	150/96	55	786 MB/165 MB
3	80	55	198/118	38	914 MB/225 MB
4	160	41	245/135	21	1.7 GB/243 MB
5	320	19	300/185	overflow	overflow
6	500	14	454/210	overflow	overflow

3.3 交互效果

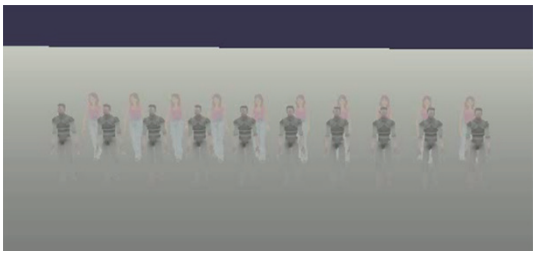
在用户完成 2D 平台的数据输入后, 单击开始动画, 查看 2D 平台下自己编排的队形的展示效果, 如图 5(a), 如果满意, 即可点查看 3D 平台下的效果展示. 接着 3D 平台会调用前面导出的数据, 自动生成模型以及加载场景库. 与此同时还可以使用‘空格’键控制音乐的播放; ‘n’键切换 歌曲; ‘g’键切换地面场景, 如图 5(b); ‘f’键切换烟雾, 如图 5(c); ‘l’键切换灯光效果, 如图 5(d); ‘上下左右’方向键或单击鼠标拖动场景, 即可完成对视角的切换; 滑动鼠标滚轮放大、缩小场景.

特别的, 为了更加逼真的模拟真实的人物, 同时还支持了物理碰撞以及重力效果, 使得系统整体的使用效果较好.

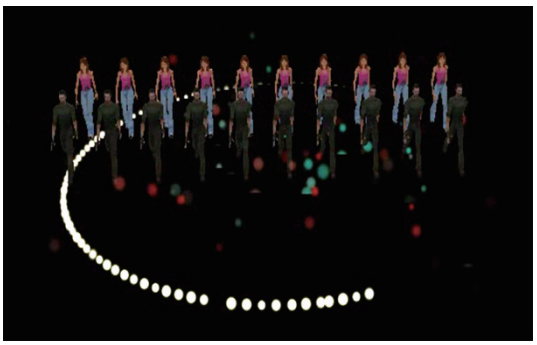


(a) 2D平台设计

(b) 修改天空



(c) 添加烟雾



(d) 添加灯光效果

图 5 虚拟编排系统

4 总结与展望

本文主要讲述了针对于大型团体舞排练的研究与实现, 从 2D 角度以及 3D 角度分别提出、设计了相应的技术方案, 该方案能够较为完美的从多角度展示导演的设计灵感, 极大的还原设计效果. 在整个设计过程中, 从输入到处理到输出, 分别提出了相应的优化方案, 使得整个系统使用起来十分流畅, 极大的提高了导演在设计时的效率. 但是现阶段整个编排系统还不是那么完善, 还有很多值得优化的地方, 比如如何减少内存的占用、如何让 GPU 更多的参与整个绘制的过程等. 相信随着后续算法、技术上的不断优化, 整个系统会变得更加的完美.

参考文献

1 巩凌. 计算机辅助团体操队形图设计的初步研究. 北京体育师范学院学报, 1993, 5(1): 59–62.

2 黄宽柔. 我国团体操的发展与展望. 体育科学学会学报 (体育科学), 1994, 14(1): 29–32.

3 纪庆革, 潘志庚, 李祥晨. 虚拟现实在体育仿真中的应用综述. 计算机辅助设计与图形学学报, 2003, 15(11): 1333–1338. [doi: 10.3321/j.issn:1003-9775.2003.11.001]

4 Murakami Y, Ishida T, Kawasoe T, *et al*. Scenario description for multi-agent simulation. Proc. of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems. Melbourne, Australia. 2003. 369–376.

5 Tecchia F, Loscos C, Chrysanthou Y. Visualizing crowds in real-time. Computer Graphics Forum, 2002, 21(4): 753–765. [doi: 10.1111/cgf.2002.21.issue-4]

6 Marrin C. WebGL specification. Khronos WebGL Working Group. https://cvs.khronos.org/svn/repos/registry/trunk/public/webgl/doc/spec/WebGL-spec.html. [2011].

7 黎瑞成. 基于 WEBGL 和 HTML5 的三维红色娘子军动画系统的设计与实现[硕士学位论文]. 广州: 中山大学, 2015.

8 HTML 5: A vocabulary and associated APIS for HTML and XHTML. W3C Working Draft. https://www.w3.org/TR/2012/WD-html5-20120329/. [2012-03-29/2012-07-30].

9 陈滔滔, 江晓宇, 温佩贤, 等. 基于 Web3D 的人脸模型定制系统. 系统仿真学报, 2014, 26(2): 382–388.

10 崔滨. 海量数据实时三维交互式显示关键技术研究[博士学位论文]. 上海: 上海大学, 2010.

11 Seth Thompson. Introduction, 1 revision. https:/github./com/v8/v8/wiki/Introduction. [2015-11-26].

12 刘旭. Chrome V8 引擎中的 JavaScript 数组实现分析与性能优化. 计算机与现代化, 2014, (10): 66–70. [doi: 10.3969/j.issn.1006-2475.2014.10.016]