PaperFree检测报告简明打印版

相似度：41.14%
编　号：RXWHNMXCDRXJBZZH
标　题：论文查重版
作　者：-
长　度：10955字符
时　间：2017-05-10 14:49:17
比对库：中国学位论文全文数据库；中国学术期刊数据库；中国重要会议论文全文数据库；英文论文全文数据库；互联网资源；自建比对库

相似资源列表(学术期刊，学位论文，会议论文，英文论文等本地数据库资源)
1. 相似度：5.44%　篇名：《VBR流式视频的最短路径率平滑传输算法》
来源：《计算机学报》　年份：2004　作者：谢建国
2. 相似度：1.56%　篇名：《卡牌游戏之后是什么？》
来源：《家用电脑与游戏》　年份：2013　作者：FXCarl
3. 相似度：1.35%　篇名：《毕业论文致谢词》
来源：《学习博览》　年份：2011　作者：卫小平
4. 相似度：0.45%　篇名：《毕业论文致谢词》
来源：《学习博览》　年份：2011　作者：卫小平
5. 相似度：0.37%　篇名：《卡牌游戏之后是什么？》
来源：《家用电脑与游戏》　年份：2013　作者：FXCarl
6. 相似度：0.29%　篇名：《A~ * 寻找最短路径算法及实现》
来源：《电脑编程技巧与维护》　年份：2013　作者：王文举
7. 相似度：0.23%　篇名：《迷宫游戏的设计与开发》
来源：《齐齐哈尔大学学报：自然科学版》　年份：2014　作者：田翠华
8. 相似度：0.09%　篇名：《基于Unity3 D技术的数字人体腧穴教学信息系统的设计与实现》
来源：《中国医学教育技术》　年份：2014　作者：张季
9. 相似度：0.08%　篇名：《“定肘”对提升乒乓球击球效益的运动力学分析》
来源：《体育研究与教育》　年份：2013　作者：谢冬兴

相似资源列表(百度文库，豆丁文库，博客，新闻网站等互联网资源)
1. 相似度：4.51%　标题：《1.2.1 电子游戏作为软实时模拟_游戏引擎架构_红黑联盟读书频道》
来源：http://book.2cto.com/201402/40619.html
2. 相似度：3.12%　标题：《让你的A*寻路更加真实(续)_踏浪星空编程_新浪博客》
来源：http://blog.sina.com.cn/s/blog_6247da3c0101gfis.html
3. 相似度：2.89%　标题：《A*寻路-- 更加真实的路径(二) - mybloglucis009的专栏- 博客频道- ...》
来源：http://blog.csdn.net/mybloglucis009/article/details/8497460
4. 相似度：2.81%　标题：《A*算法的Actionscript3.0实例- 橡树小屋- 博客园》
来源：http://www.cnblogs.com/babyzone2004/archive/2010/09/17/1829615.html
5. 相似度：2.48%　标题：《Floyd算法- EnigmaJJ - 博客园》
来源：http://www.cnblogs.com/twjcnblog/archive/2011/09/07/2170306.html
6. 相似度：2.29%　标题：《A*算法原理- 小_鱼的专栏- 博客频道- CSDN.NET》
来源：http://blog.csdn.net/u011068464/article/details/29913535
7. 相似度：2.10%　标题：《图形学实验报告一(画直线)_百度文库》
来源：http://wenku.baidu.com/view/af0a1c5177232f60ddcca161.html?from=rec
8. 相似度：1.87%　标题：《Floyd算法详讲- 飘过的小牛- 博客频道- CSDN.NET》
来源：http://blog.csdn.net/niushuai666/article/details/6772706
9. 相似度：1.87%　标题：《最短路径—Dijkstra算法和Floyd算法 - as_ - 博客园》
来源：http://www.cnblogs.com/biyeymyhjob/archive/2012/07/31/2615833.html
10. 相似度：1.67%　标题：《计算机图形学——生成直线的DDA算法 - Naruto - 博客频道 - CSDN...》

来源：http://blog.csdn.net/zhangkaihang/article/details/7464400
11. 相似度：1.67% 标题：《计算机图形学直线生成_百度文库》
来源：
http://wenku.baidu.com/link?url=5WcK3qEWNCjBhI51XgGZ_SkuLAnCPK5idltK0s9yCrt54Uh6vEK1OlDY
12. 相似度：1.49% 标题：《A星算法【转】- 下篇_学步_新浪博客》
来源：http://blog.sina.com.cn/s/blog_9cee0fd90101cs3d.html
13. 相似度：1.35% 标题：《2016毕业论文感谢词》
来源：http://www.oh100.com/a/201606/347978.html
14. 相似度：1.20% 标题：《3d游戏寻路简介- Benjamin - C++博客》
来源：http://www.cppblog.com/zhangyq/archive/2014/12/13/209170.html
15. 相似度：1.11% 标题：《A*方法总结 - 逍遥剑客 - 博客频道 - CSDN.NET》
来源：http://blog.csdn.net/xoyojank/article/details/1423329
16. 相似度：1.04% 标题：《再探A*寻路算法 - 道客巴巴》
来源：http://www.doc88.com/p-301948375506.html
17. 相似度：0.98% 标题：《计算机图形学第二讲 直线和圆的生成_免费下载_百度文库》
来源：http://wenku.baidu.com/link?url=gZYnpEGiZsYdwdIb_sLUHEznBZ8mHE78D-Ae3jK6szksNXwupfqggtlzE-adwPlzic972xWyNT-_IeCPZk5GapRp1TYhtqvYoKBgcJIFJUG
18. 相似度：0.98% 标题：《第二章 二维基本图形的生成_百度文库》
来源：
http://wenku.baidu.com/link?url=cEjWD71nB2OpnSVI5R2YbkN3I22YdTaGplZWBlpWcIzJMBIVwxz9MXenQhepa
19. 相似度：0.90% 标题：《毕业论文 - 道客巴巴》
来源：http://www.doc88.com/p-14452153605.html
20. 相似度：0.88% 标题：《A*寻路初探（转载） - 冰封一夏 - 博客园》
来源：http://www.cnblogs.com/bfyx/archive/2012/11/02/2750957.html
21. 相似度：0.83% 标题：《五月| 2012 | Kid's Zone》
来源：http://www.kidsang.com/archives/date/2012/05
22. 相似度：0.74% 标题：《让你的A*寻路更加真实(续) - 其他 - 游戏开发者社区》
来源：http://bbs.9ria.com/thread-95620-1-1.html
23. 相似度：0.69% 标题：《推箱子_图文_百度文库》
来源：http://wenku.baidu.com/view/fafc5013bd64783e09122b9d.html
24. 相似度：0.67% 标题：《unity3d 摄像机抖动情况和解决方案汇总 - yxriyin的专栏 - 博客频道 - ...》
来源：http://blog.csdn.net/yxriyin/article/details/39237673
25. 相似度：0.66% 标题：《游戏开发最基础的组成元素是什么? - 计算机- 知乎》
来源：https://www.zhihu.com/question/24913042
26. 相似度：0.62% 标题：《从原点出发,遍历50个点,再回到原点的最短路径,求matl..._百度作业帮》
来源：https://www.zybang.com/question/92555f0f70e00c1949615c5bf06ddf94.html
27. 相似度：0.58% 标题：《漏斗平滑算法 - yxriyin的专栏 - 博客频道 - CSDN.NET》
来源：http://blog.csdn.net/yxriyin/article/details/39207709
28. 相似度：0.56% 标题：《如何理解floyed算法 5 - 百度知道 - 全球最大中文互动问答平台》
来源：https://zhidao.baidu.com/question/295460856.html
29. 相似度：0.55% 标题：《图形学复习_文档库》
来源：http://www.wendangku.net/doc/fddf555d312b3169a451a40e.html
30. 相似度：0.47% 标题：《高德 轨迹平滑算法 - 蓝讯》
来源：http://www.lxway.com/1297551.html
31. 相似度：0.34% 标题：《《计算机系统应用杂志》2010年07期是CSCD期刊核心吗》
来源：http://www.med126.com/qikan/2016/20160724043004_1647712.shtml
32. 相似度：0.31% 标题：《1.5.1 雷神之锤引擎家族_游戏引擎架构_红黑联盟读书频道》
来源：http://book.2cto.com/201403/40629.html
33. 相似度：0.29% 标题：《算法》
来源：http://www.fx114.net/qa-99-110930.aspx
34. 相似度：0.27% 标题：《... 结果完成,每一步都倾注了老师大量的心血,在论文修改过程中,雷》
来源：http://www.39394.com/fanwen/doc/266199.html
35. 相似度：0.24% 标题：《E3 2016：网传id Software下一部作品会是《QUAKE》 _ 游民星空 ...》
来源：http://www.gamersky.com/news/201606/763805.shtml

36. 相似度：0.16% 标题：《3D游戏大全_3D游戏排行榜_苹果3D游戏_苹果ios3D游戏_iPhone3D类...》
来源：http://www.yiwan.com/zt/pingguo3D/
37. 相似度：0.10% 标题：《从集中式到分布式- 五月的仓颉- 博客园》
来源：http://www.cnblogs.com/xrq730/p/4944818.html
38. 相似度：0.10% 标题：《玩游戏,会用按键精灵的叫我,只要他按住一个键就行_360问答》
来源：http://m.wenda.so.com/q/1372528732065098
39. 相似度：0.10% 标题：《DotA2 wiki百科 | 中华网游戏大全》
来源：http://game.china.com/gbox/dota2/baike.html

全文简明报告

一、引言

1.1 背景

{ 60% : 随着PC性能的提升、智能手机的普及, }使得现如今人们设备的性能越来越高,3D游戏所需要的性能很容易就可以满足。所以3D游戏市场非常火热。但是在游戏开发过程中,我们时常会遇到游戏画面抖动的情况。如果任由这种情况存在,是对游戏可玩性的一大损害。

1.2 意义

可玩性是如同字面一样,说的是游戏让玩家玩下去的性质。对于电子游戏这一个特殊的软件来说,可玩性是其核心内容。可玩性主要受游戏类型与游戏设计、游戏操作与响应、学习曲线和精通难度的影响。

3D游戏相较于2D游戏来说,从图形学角度来讲,多了一个z轴,使得3D游戏是在空间的层次上进行展示,而2D是在平面的层次上进行显示。2D游戏现实只需要将屏幕的一部分直接投射到屏幕上就可以,但是3D不同,由于3D使得游戏有远近、深浅的概念,简单的投射已经不可行。3D游戏采用了人眼的模式,来展现游戏世界。如果游戏的过程中,游戏画面抖动,将会使玩家感觉自身在抖动,影响玩家在游戏中的判断,降低游戏的可玩性。

由于可玩性是电子游戏的核心,消除游戏中画面抖动势在必行。采用本文的方法可以消除开发过程中出现的画面抖动,提高游戏的体验,提高游戏的可玩性,增加用户粘度,减少开发负担。

1.3 现状

在当前,3D游戏除了一部分卡牌类游戏以外,其余的游戏基本上都需要涉及到画面的移动、人物的移动等。在这些情况下,出现画面抖动的情况非常高。应用消除抖动是势在必行的。而当前,在3D游戏开发过程中中,遇到了画面抖动,一般采用的是漏斗平滑算法来进行消除。

漏斗平滑算法在八格子寻路时效率不好,不适用于某些类型的游戏。

1.4 其他抖动消除算法简述

漏斗平滑算法是3D游戏中,消除抖动时最常见的一种平滑算法。

漏斗平滑算法过程是基于边界的。{ 67% : 每次判断下一个点是否在漏斗中, }再会构建新的漏斗继续进行判断,不在则把之前在的点设为新的起点来构建漏斗,重复进行。但是这个算法是基于边界和点的。在某些情况下会效率比较低下,不适用于某些游戏。

1.5 工作与创新点

在本文中,通过对传统A*算法的拓展,实现了一个平滑算法,实现了适用于3D游戏的平滑算法。

本文的创新点在于本文中所使用的算法是当前已经存在多时的经典寻路算法A*算法的改造,算法简单、可。代码实现上也很简单,方便应用到实际项目中。

二、相关算法

2.1 A*算法

A*算法原本是一种求解最短路径直接搜索方法,是一种启发式搜索算法。

公式表示为:

f(n) = g(n)+h(n)

{ 70% : 其中, f(n) 是从初始状态经由状态n到目标状态的代价估计,g(n) 是在状态空间中从初始状态到状态n的实际代价,h(n) 是从状态n到目标状态的最佳路径的估计代价。 }

A*算法的具体过程为:

{100% : 1,把起始格添加到开启列表。 }

2,重复如下的工作:

{100% : a) 寻找开启列表中F值最低的格子。 }我们称它为当前格。

b) 把它切换到关闭列表。

c) 对相邻的格中的每一个?

{100% : * 如果它不可通过或者已经在关闭列表中,略过它。 }反之如下。

{100% : * 如果它不在开启列表中,把它添加进去。 }{100% : 把当前格作为这一格的父节点。记录这一格的F,G,和H值。 }

{97% : * 如果它已经在开启列表中,用G值为参考检查新的路径是否更好。 }{100% : 更低的G值意味着更好的路径。如果是这样,就把这一格的父节点改成当前格,并且重新计算这一格的G和F值。 }{97% : 如果你保持你的开启列表按F值排序,改变之后你可能需要重新对开启列表排序。 }

d) 停止,当你

* 把目标格添加进了关闭列表,这时候路径被找到,或者

{93% : * 没有找到目标格,开启列表已经空了。这时候,路径不存在。 }

{100% : 3.保存路径。从目标格开始,沿着每一格的父节点移动直到回到起始格。 }这就是你的路径。

2.2 漏斗平滑算法

漏斗平滑算法的原理在之前已经说,{94% : 是判断下一个点是在漏斗范围内,就直接移动过去。 }{100% : 如果出了范围,那么就设置成一个点,重新一轮漏斗。 }

{93% : 如图 所示, O(t) 和 U(t) 构成的区域可以看作是一个单调多边形。 }{96% : 我们用 Π(s,w) 来表示多边形内点 s至点 w 的最短路径,那么 Π(s,e) 即为所求最短路径。 }{96% : 显然, Π(s,e) 只可能与多边形的凹点(例如 v)相交,不可能与凸点(例如 z)相交。 }{97% : 因此,计算最短路径只需要考虑凹点。 }{90% : 假设已知 Π (u,s) = [u,s] 及 Π (s,w) =[s,v,w] ,那么漏斗就是由 Π(u,s) 以及 Π(s, }{ 75% : w) 构成的 V形结构,用 表示,而 s (V 形底端)为漏斗的底。 }{93% : 是由最初漏斗经过数次更新后生成的,最初的漏斗由起始点 s 与 O(t) 、 U(t) 的第一个凹点的连线构成的,如图中虚实线所示。 }

{84% : 在现有漏斗 的基础上,计算出漏斗底至下一凹点 x 的最短路径 Π(s,x) ,那么漏斗就更新为 ,依次类推计算后续凹点直至终点 e ,即可得到 F *e ,而 F *e 的下半部分即为所求的 Π(s,e) }

{84% : 在 的基础上,计算 Π(s,x) 的方法如下。 }{88% : 首先,如果 s 到 x 的直线不与 相交,那么 Π(s,x) 就是直线 sx 。 }{ 79% : 否则,在 中寻找一个点 vi ,使得 vix 与 相切。 }用公式可以表示为:

{ 73% : 其中 tg 表示线段的斜率, v i-1 v i 以及 v i v i+1 为 Fuw 中v i 的前后线段。 }{ 79% : 找到点 v i 后连接v ix ,那么 Π(s,x)=[s,...vi,x]。 }[footnoteRef:1] [1:{85% : 摘自 《基于漏斗的实时VBR 视频最短路径平滑算法》,袁俊杰 徐小良, }《计算机系统应用》2010 年 第 19卷 第 7 期]

以上图来说明漏斗的过程:

1. 起点与两边边界最开始的点连线,这两条线构成了一个漏斗。

2. 分别测试两边边界上的下一个点与起点的联系是否在前面的点生成的漏斗中。如果都在,则执行3;若一条在一条不在,则执行4;若都不在则执行5。

3. 以新的两条线原起点构成新的漏斗,重复2。

4. 以在漏斗中的那条新线为新边界与另一边的原边界和原起点构成新的漏斗,重复2;

5. 如果都不在,则以两条线中的最短的一条中的端点为新的起点,这条线段为路径中的一段加入到结果集合中。重复1。

图A中起点与最近的两个点组成漏斗。B中下侧边界的下一点与起点连线,该线在漏斗中。图C中上边界的下一点与起点连线,连线在原漏斗中。则以这两条线为新的漏斗边。D图中,下边界下一点与起点连线,线在新的漏斗中,E图中,上边界下一点与起点连线,不在漏斗中,舍弃。下边界新线与原上边界线组成新的漏斗。F图中下边界下一点与起点连线,不在漏斗中,舍弃,上边界下一点与起点连线,不在漏斗中,舍弃。选择原漏斗边界中最短一条的端点为新起点。如图G。重新开始漏斗处理。

2.3 弗洛伊德路径平滑算法

弗洛伊德算法是解决任意两点间的最短路径的一种算法,{100% : 可以正确处理有向图或负权的最短路径问题,同时也被用于计算有向图的传递闭包。 }{100% : Floyd-Warshall算法的时间复杂度为O(N3),空间复杂度为O(N2)。 }

{93% : 弗洛伊德算法的基本思想如下:从任意节点A到任意节点B的最短路径不外乎2种可能,1是直接从A到B,2是从A经过若干个节点X到B。 }{99% : 所以,我们假设Dis(AB)为节点A到节点B的最短路径的距离,对于每一个节点X,我们检查Dis(AX) + Dis(XB) ^ Dis(AB)是否成立,如果成立,证明从A到X再到B的路径比A直接到B的路径短,我们便设置Dis(AB) = Dis(AX) + Dis(XB),这样一来,当我们遍历完所有节点X,Dis(AB)中记录的便是A到B的最短路径的距离。 }

其算法描述:

{100% : a.从任意一条单边路径开始。所有两点之间的距离是边的权,如果两点之间没有边相连,则权为无穷大。 }

{100% : b.对于每一对顶点 u 和 v,看看是否存在一个顶点 w 使得从 u 到 w 再到 v 比已知的路径更短。 }如果是更新它。

弗洛伊德路径平滑算法是一种在内部应用了弗洛伊德算法的平滑算法。

{100% : 弗洛伊德路径平滑算法应在通过A*寻路算法得出路径后进行,它的步骤分为两步:一、合并路径数组中共线的节点;二、尽可能地去掉多余拐点。 }这个过程如下图所示:

去掉共线点

去掉多余拐点

{100% : 可以看到,使用弗洛伊德路径平滑处理 后的路径正如我们期望的那样,而且大大削减了路径数组中的节点数目。 }

{99% : 不难发现,若存在三点A(1,1), B(2,2), C(3,3),若B与A的横、纵坐标差值分别等于C与B的横、纵坐标差值,则A,B,C三点共线。 }

{100% : 仔细观察第三幅图你会发现,若路径中存在节点A,B,C,D,E,F,G,如果A与G之间的连线所经过的节点中没有一个节点是不可移动节点,则我们称A与 G之间是不存在障碍物的。 }{100% : 如两节点间不存在障碍物,则可以去掉此两点间其他所有节点。 }{100% : 如上例中A-G这些节点,若A与G之间不存在障碍物,则我们可以去掉 A与G之间的B,C,D,E,F节点,最终路径数组中只剩下A与G两个节点。 }

2.4 DDA算法

数值微分法即DDA法(Digital Differential Analyzer),是一种基于直线的微分方程来生成直线的方法。

算法描述:

{98% : 设(x1,y1)和(x2,y2)分别为所求直线的起点和终点坐标,由直线的微分方程得: }

dx/dy = (y2-y1)/(x2-x1) = m = 直线斜率 = (1)

{85% : 可通过计算x方向上的增量引起y的改变来生成直线。 }

xi+1 = xi+ (2)

yi+1 = yi+ * m (3)

{95% : 也可通过计算由y方向的增量引起x的改变来生成直线: }

yi+1 = yi + (4)

xi+1 = xi + /m (5)

{89% : 选择x2-x1与y2-y1中较大者作为步进方向(假设x2-x1较大),取该方向上的增量为一个象素单位(=1),然后利用式(1)计算另一个方向的增量(=*m =m)。 }{93% : 通过递推公式(2)至(5),把每次计算出的(xi+1, yi+1)经取整后送到显示器输出,则得到扫描转换后的直线。 }

{99% : 之所以取x2-x1和y2-y1中较大者作为步进方向,是考虑沿着线段分布的象素应均匀。 }

三、原因与原理

在了解抖动发生的原因之前这之前,我们需要首先了解3D游戏的发展之路以及一些基本概念。

{ 58% : “游戏”一词泛指棋类游戏例如象棋和《大富翁》;纸牌游戏,例如梭哈和二十一点;赌场游戏例如轮盘

和老虎机;军事战争游戏、计算机游戏、孩子们一起玩耍的游戏。 }{ 74% : 在计算机的语境下,“游戏”一词会使我们在脑海中浮现出一个虚拟世界,玩家可以控制人物、动物或玩具。 }

绝大部分游戏是软实时互动基于代理计算机模拟的例子。

{ 71% : 在电子游戏中,会用数学方法来为真实世界的子集建模,从而使这些模型在计算机中运行。 }显然,{ 70% : 这些模型只能是显示或者想象世界的简化或者近似版本, }{90% : 因此,数学模型是现实或者虚拟世界的模拟。 }

{95% : 基于代理模拟是指,模拟中多个独立的实体(称作代理)一起互动。 }

{ 67% : 所有的互动游戏都是时间性模拟的,即游戏世界是动态的——随着游戏事件和故事的展开,游戏的状态随着时间改变。 }{95% : 游戏也必须响应人类玩家的输入,这些输入是游戏本身不可预知的, }这也说明游戏是互动时间性模拟的,大多数游戏会实时回应用户输入,这即为互动实时模拟。

{80% : 软实时是指即使错过期限却不会造成灾难性的后果。 }所有游戏都是软实时的。

{ 73% : 模拟虚拟世界需要用到很多数学模型。 }{ 70% : 数学模型分为解析式和数值式。 }{94% : 例如,一个刚体因为地心引力而以恒定加速度落下, }其分析式数学模型可写为:

y(t)= gt2 +v0t+y0

{ 57% : 分析式模型可为其自变量设任何值来求值。 }例如上式,{97% : 给予初始条件v0和y0、常量g,就能设任何时间t来求y(t)的值。 }{83% : 在电子游戏中,用户的输入是不能预知的,因此不能预期对整个游戏完全适用分析式建模。 }

刚体受地心引力落下的数值式模型可写为:

y(t+ t)=F(y(t),y`(t),y``(t),...)

{90% : 也就是说,该刚体在(t+ t)未来事件的高度,可以用目前的高度、高度的第一导数,高度的第二导数及目前的时间t为参数的函数来表示。 }{ 76% : 为实现数值式模拟,通常需要不断重复的计算,以决定每个离散时间的系统状态。 }{86% : 游戏也是如此运作的,一个主游戏循环不断执行,在循环的每次迭代中,多个游戏系统,例如人工智能、游戏逻辑、物理模拟等,就会有机会计算或者更新其下一个离散时间的状态。 }{91% : 这些结果最后可渲染成图形显示、发出声效或者书橱至其他设备。 }

{ 60% : 游戏引擎这个术语在20世纪90年代中期形成,这与第一人称射击游戏如id software公司的《DOOM》有关。 }{ 67% : 《DOOM》将其软件构架划分为核心软件组件(如三维图形渲染系统、碰撞检测系统和音频系统等)、美术资产、游戏世界、构成玩家游戏体验的游戏规则。 }{82% : 这样的划分非常有价值,另一个开发商取得了这样的游戏的授权之后,只需要制作新的美术、关卡布局、武器、角色、游戏规则等,对引擎软件做出很少的修改,就可以把游戏打造成新产品。 }

现在主流、常见的商业引擎有:{ 59% : Value公司的Source引擎, }{ 65% : Epic的Unreal引擎, }以及在移动端很常见的跨平台商业引擎Unity3D,Cocos2dx等。另外还有很多游戏公司有自己专用的私有引擎。

一般认为,首个三维第一人称射击游戏是《德军司令部》。{ 63% : 这款游戏有美国的id software于1992年制作,他引领游戏产业进入到了令人们兴奋的领域。 }{ 66% : Id software又相继开发了《DOOM》、《Quake》等游戏。 }随着20多年的发展,3D游戏已经变得非常常见,基本上已经是随处可见。现如今的游戏大多数是3D游戏。

3.1 抖动与抖动发生的原因

在3D游戏开发的过程中,当在游戏内对游戏中的对象进行移动、转向时,游戏的画面会发生剧烈的抖动,会使玩家晕眩,影响玩家体验。

要明白抖动的发生,首先需要明白电子游戏是怎么运作的。首先,需要认识到电子游戏也仅仅是一个可编译和运行的程序。就像其他的程序一样,main函数也是它第一个被执行的函数。游戏的结构如下图所示:

上图包含了五种状态,电子游戏最重要的三种状态在循环中互相连接,这个循环通常称作游戏循环。游戏循环式进行玩家输入,更新游戏内部数据,更新显示的地方。

初始化是尤其开始的地方,通常包含了启动动画以及一个包含了音量、画质等选项的选项菜单,{ 56% : 在这个过程中会进行一些必要参数的设定, }以及申请内存,设置堆栈,检测游戏环境以及加载显示驱动(比如包含集显和独立显卡的机器来加载独立显卡)。在这些都已经设置完毕后,玩家选择了开始游戏,则跳转到下一个状态。

{ 65% : 玩家输入是对玩家所使用的输入设备例如键盘、鼠标、游戏手柄等进行监听, }对玩家的输入的信息进行采集,并且在游戏内部中以游戏能够处理的形式进行存储。供后一个状态使用。

更新游戏内部是游戏的心脏,游戏根据玩家的输入,来改变游戏中的设定值,例如对玩家操作的人物进行移动,玩家游戏中的敌人的运动与反应,同时准备好显示所需要的所有数据与图像。

更新显示是将计算后的画面投射到屏幕上显示出来。一般来说,你可以选择对每个物体挨个在屏幕上绘制或者说按照例行办法,先把所有的对象计算好,然后进行一次性的绘制。在屏幕上进行绘制是一个花费时间比较多的过程。所以一次性进行绘制将会有比较好的性能。

结束游戏是玩家选择退出游戏或者玩家通关之后的一种正常状态。一般会有一个结束动画或者提示语来告诉玩家游戏已经结束了。同时也会进行数据的保存,释放内存,消除堆栈等工作。

摄像机指的不是现实社会中用来照相的那个摄像机,但是两者发挥的作用基本上是一样的。电子游戏中的摄像机用另外的一个词来说是“视角”,也就是看游戏世界的角度。展现在玩家屏幕上的画面便是通过摄像机的“照”出来的画面。

由于玩家的画面是由摄像机照出来的,那么画面抖动会有两个原因:游戏世界在抖动或者是摄像机在抖动。 游戏世界我们没有进行直接操作,那么按道理来说不会发生抖动,这个可能性就可以被排除掉。那么抖动发生的原因在于摄像机在抖动。

开发游戏时,当对游戏内部的对象进行移动、旋转操作时时,由于游戏本身的设定,摄像机也会跟随着玩家移动的对象进行移动、旋转,这在酷跑类游戏中很常见。由于游戏是帧驱动的,游戏循环每执行一次,游戏就播放一帧。每一帧中都会执行摄像机位置的变换,使之看向所需要看向地方,{80% : 摄像机不停变换位置,而且是无规则的。 }{ 79% : 不规则的位置变化会引起抖动。 }

3.2 抖动消除原理

既然画面的抖动是由于摄像机的抖动造成的,那么,我们要消除画面抖动,就只需要消除摄像机的抖动就可以了。画面抖动都发生在画面移动时即摄像机朝向或者位置发生变化时。由于现今的游戏通常都是帧驱动的游戏,我们需要在游戏循环每次执行的时候,让摄像机位置、朝向的改变变得平滑也就是使得摄像机看向的位置移动变得平滑,这样,在游戏的过程中,我们所看到的画面是在进行平滑的变换,就如同我们自身走动一般。

抖动消除有一个前提,时间上要是平滑的。也就是说要么是每一帧执行所用的时间是一样那要么就处理成一样的效果。现代游戏是帧驱动的,我们不能保证每一帧所执行的时间是一致的,但是我们可以获得每一帧执行所消耗的时间,在处理时,我们以消耗的时间为影响因子乘如,这样便实现了时间上的平滑。

再者是路径上的平滑,{100% : 平滑无非两种,一种是延迟,一种是根据以前的速度进行加速度计算,然速度变化变慢。 }

本文中消除抖动的平滑算法分为两部分:直行时平滑部分和转动时平滑部分。分别对应在直行时位移所导致的抖动和转向时旋转导致的抖动。

3.3 算法实现

3.3.1不转向时平滑算法

在直行时,基本上不会涉及到摄像机朝向的变化,也基本上不会导致摄像机旋转。那么我们可以使摄像机朝向某一个确定的方向,控制摄像机沿着平滑的路径,以平滑的速度运动。

对于平滑的路径,我们使用A*寻路找到这样的路径。采用以下步骤:

{100% : 1,把起始格添加到开启列表。 }

2,重复如下的工作:

{100% : a) 寻找开启列表中F值最低的格子。 }我们称它为当前格。

b) 把它切换到关闭列表。

c) 对相邻的格中的每一个?

{100% : * 如果它不可通过或者已经在关闭列表中,略过它。 }反之如下。

{100% : * 如果它不在开启列表中,把它添加进去。 }{100% : 把当前格作为这一格的父节点。记录这一格的F,G,和H值。 }

{97% : * 如果它已经在开启列表中,用G值为参考检查新的路径是否更好。 }{100% : 更低的G值意味着更好的路径。如果是这样,就把这一格的父节点改成当前格,并且重新计算这一格的G和F值。 }{97% : 如果你保持你的开启列表按F值排序,改变之后你可能需要重新对开启列表排序。 }

d) 停止,当你

* 把目标格添加进了关闭列表,这时候路径被找到,或者

{93% : * 没有找到目标格,开启列表已经空了。这时候,路径不存在。 }

{100% : 3.保存路径。从目标格开始,沿着每一格的父节点移动直到回到起始格。 }这就是你的路径。

该过程的位代码为:

开启列表 ← 起始格

关闭列表 ← {}

repeat

当前格 ← min(F(开启列表))

foreach 相邻格(当前格) do

if 相邻格(当前格) in 关闭列表 or 相邻格(当前格) no access then

skip

end

if 相邻格(当前格) not in 开启列表 then

开启列表 ← 相邻格(当前格)

end

if 相邻格(当前格) in 开启列表 then

if G(P(相邻格(当前格)))^ G(当前格) then

当前格 ← 父节点(相邻格(当前格))

renew(F(相邻格(当前格)))

renew(G(相邻格(当前格)))

end

end

end

until 目标格 in 关闭列表 or 开启列表 = {}

save path

使用A*寻路算法之后,我们得到了一条摄像机移动的路径。{ 63% : 这个路径是由一系列的点组成的。 }对于这些点,可能有重复的点,我们需要使用弗洛伊德路径平滑算法来对结果进行简化、平滑。

算法过程:

对于初识连续的点A1, A2, A3... An中的某个点Ai (x i, y i, z i),如果前一点Ai-1 (x i-1,y i-1,zi-1)和后一点Ai+1 (xi+1, yi+1, zi+1),如果xi-xi-1=zi+1-zi且yi-yi-1=yi+1-yi且zi-zi-1=zi+1-zi那么这三点共线,则在点集中除去Ai点。将所有共线点处理后得到新的点集A1 ,A2, A3,.. Ax,如果A j,A j+1, ... A j+ m中的点都不是不可移动点且A j 到 A j +m之间没有障碍物,则删除A j+1到A j+m-1的所有点。

伪代码如下:

初识点集 ← {A1,A2,A3....An}

i ← 1

repeat

if Ai,Ai-1,Ai+1共线 then

delete Ai

end

i ← i+1

until i = n

i ← 1

repeat

if Ai,Ai-1,Ai+1都可以移动 and Ai, Ai+1间没有障碍 then

delete Ai

end

until

通过上述方法,得到了平滑的路径。

得到了平滑的路径之后,我们需要对速度进行处理。由于游戏是一帧一帧进行执行,每一帧所耗费的时间可能不一样,因此,摄像机移动的距离不能简单的设置为一段距离,应当以一段距离乘以一帧执行的时间。距离选取不应太长,否则会出现移动不连贯,出现跳帧的现象。

如果在运动的过程中有速度变化,则需要有一个连续的速度变化过程,不能瞬间就变化到某个速度去。例如启动运动的时候,游戏中物体会由静变化到动,这在这个过程中,需要有一个加速的过程。可以采用一个恒定的加速度。

3.3.2 转向时的平滑算法

2D游戏中由于只有x,y两根坐标轴,只会在平面上进行移动,不会存在转向的问题。但是在3D游戏之中,很有可能会有转向。在这种情况下,摄像机的朝向会发生改变,朝向变化也需要进行平滑。

转向时平滑需要考虑的是自然的转向不是就在原地为圆心做圆周运动,而是以原地以外的某个点为圆心做圆周运动,因此,在转向时,需要进行处理。通过圆的二维平面方程:

得到圆的轨迹。摄像机的朝向与摄像机坐标与圆心的方向相同。

此时运动的长度可以根据角度来。相同时间转动的角度应该是确定的。则根据转过的角度θ来得到每一帧处于的点和朝向。设某点为(x, y, z),为了简化处理,假设圆心为(0,0,0),半径为r,在平面Ax + By = 0上画圆,单位时间转动的角度值为α,则新一点的坐标值为

其中+号还是-号取决于方向。

对于一般情况下转向,就只简单的做一下数学上的推导,在实际游戏开发中,一般都会选择Ax+By=0这个平面为基准面。

在三维空间中,设圆的圆心为(x0, y0, z0),圆的半径为r,圆的法向量为(α, β, γ),则以(x0, y0, z0)为圆心以r为半径的球面方程为

圆所在的平面方程为:

联立两式得到空间中圆的方程:

设之前的点为(x, y, z),{ 59% : 单位时间内转动的角度值为μ, }则有方程:

联立上式,即可得到新的点的坐标。

四、实验与分析

4.1 实验结果

我们在所开发的demo为一个酷跑类型的游戏,游戏过程中,玩家控制游戏中的角色一直前进,中途可能会转向会有障碍物。摄像机在游戏过程中会一直跟随着玩家来一起运动。{ 67% : 游戏demo使用的是Unity3D引擎来制作。 }

在使用平滑算法之前,游戏在匀速直行的时候,基本不会发生抖动。但是如果速度发生变化例如从静到动的时候或者有转向时或者突然横向平移时,游戏会发生剧烈的抖动。摄像机表现失控。可能会发生旋转。

将平滑算法加入到摄像机的控制脚本中之后,在匀速直线移动的时候,运动路径为直线 ,与之前一致,游戏表现正常,与加入平滑算法前的行为保持一致。在加速直线运动时,比如由静到动时,路径为直线,与之前一致,摄像机有一个加速过程。在加速过程中玩家可以很清晰的感觉到物体的速度加快,画面中物体接近摄像机的速度越来越快,过程中未发生抖动,摄像机失控的现象。转向时,摄像机现在会以转向处的外的某个点为圆心,以匀速做圆周运动转过,同时摄像机的朝向也逐步变化。屏幕上会展示出沿途的物体,转到了合适的方向之后,摄像机又开始执行,继续前进,等待玩家的指令输入。发生横向平移的时候,摄像机先水平方向的运动以一定加速度减慢,会同时在横向提供一个加速度使之加速,使之运动起来,朝向不发生改变。在这个过程之中,摄像机移动平缓,画面稳定,没有发生抖动、跳帧的情况。

4.2 结果分析

在应用算法前,匀速之前运动的时候,由于不存在速度上和路径上的不平滑,所以游戏过程中表现非常正常,没有出险抖动、跳帧、失控等不正常现象。但在加速、减数时,由于此时速度发生了变化,由于游戏没有对速度进行处理、平滑,所以速度的突然变化会导致游戏画面的抖动、跳跃。当有横向移动的时候,之前会直接瞬间横向移动,缺少了速度变化的过程,路径也不平滑,摄像机照出来的场景变化大、频繁,导致了抖动和跳帧。当转动时,最开始的是瞬间改变朝向,这样会发生非常剧烈的抖动,还可能会是摄像机失控。后面改为以原地为圆心旋转。但是这使得摄像机照出的画面显得很生硬,不自然。与现实差距较大。

在应用平滑算法之后,匀速直线运动由于路径、速度都是平滑的,结果与之前一样是在预料之中的,也是符合情景的。在加速、减速的直线运动时,之前由于速度不是平滑的,所以对速度进行平滑处理后,摄像机运动的速度有了一个平滑的变化,游戏画面变化不突兀、没有发生抖动也是符合预期结果的。转向时改变后游戏变化连续,未出现抖动现象,符合预期结果。横向移动时,可以感觉到前后速度与横向速度的变化,未出现抖动,符合预期结果。

五、总结与展望

通过应用平滑算法,使得游戏中摄像机能够平滑、稳定的进行移动、旋转,使得游戏的画面能够连续、稳定的进行变换,不会出现游戏画面的抖动。

本文所使用的平滑算法,在匀速直线运动时不会导致异常,在转向、变速运动、横向移动时,能够很好的解决画面抖动与画面不连续的问题,说明了本文的算法是较为可靠的。使能够应用到开发过程中的。

六、致谢

在本文的编写过程中,{ 56% : 感谢指导老师陈杰老师的细心指导, }老师在从设计拟题、文章的编写、文献的选取都提供了很大的帮助,在过程中每周都会进行交流、了解进度、对不清楚的部分进行指导、说明。这篇文章的成形,离不开老师的心血。在此,{100% : 谨向导师表示崇高的敬意和衷心的感谢! }

{96% : 通过论文的撰写,使我能够等系统、全面的学习有关游戏开发新型的、先进的前沿理论知识,并得以借鉴众多专家学者的宝贵经验,这对于我今后的工作和我为之服务的企业,无疑是不可多得的宝贵财富。 }{89% : 由于本理论水平比较有限,论文中的有些观点和归纳和阐述难免有疏漏和不足的地方,欢迎老师和专家们指正。 }