基于流动图与时间上升频的实时云层移动研究

陈　烨[1],李顺新[1,2,3]

(1. 武汉科技大学 计算机科学与技术学院,湖北 武汉 430065;
2. 湖北智能信息处理与实时工业系统重点实验室,湖北 武汉 430065;
3. 武汉科技大学 大数据科学与工程研究院,湖北 武汉 430065)

摘　要:现有基于光线步进的云层渲染方法能够较好地得到云层形状,但渲染效率不高且在云层移动时边缘会缺失云层移动形变细节且出现不连续的情况。针对上述问题,提出了一种基于流动图的云层移动渲染算法,算法的核心思想是利用流动图控制云层在移动时竖直方向上的偏移,使云层随风连续形变的同时增加一定前向扭曲效果。其次,为保证渲染的实时性,利用交叉矩阵来降低单帧渲染像素的数量,提高渲染速度。最后,提出使用时间上升频来增强减少渲染像素后的渲染结果。相较于未使用流动图的云层移动算法,所提算法能够在性能增加不超过 10% 的情况下表现出云层移动时的整体形变以及向风向连续翻滚的效果。

关键词:实时渲染;云层移动;光线步进;流动图;时间上升频
中图分类号:TP391　　**文献标识码**:A　　**文章编号**:1673-629X(2023)08-0037-06
doi:10.3969/j.issn.1673-629X.2023.08.006

Research on Real-time Cloud Movement Based on Flow-map and Temporal Upsampling

CHEN Ye[1], LI Shun-xin[1,2,3]

(1. School of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan 430065, China;
2. Hubei Province Key Laboratory of Intelligent Information Processing and Real-time Industrial, Wuhan 430065, China;
3. Institute of Big Data Science and Engineering, Wuhan University of Science and Technology, Wuhan 430065, China)

Abstract: Existing ray-marching based cloud rendering method is good at achieving the shape of clouds, but is not efficient and there are discontinuities at the edges when cloud moves. To address the issue above, a flow-map based algorithm is proposed for clouds movement rendering, and its main idea is to use flow-map to control the deflection of moving clouds in the vertical direction, so that the clouds can moving continuously with the wind while adding a certain forward distortion and deformation effect. Secondly, to ensure real-time rendering, the cross matrix is used to reduce the number of pixels rendered in a single frame for raising the rendering speed. Finally, temporal upsampling is introduced for enhancing the rendering results after the reduction of rendering pixels. The proposed algorithm is able to exhibit an overall shape change in cloud movement and a continuous roll towards the wind direction with no more than 10% increase in performance compared to the cloud movement algorithm without the use of flow-map.

Key words: real-time rendering; cloud movement; ray-marching; flow-map; temporal upsampling

0　引　言

目前,在视频游戏以及可交互模拟应用中,云层渲染技术的应用越来越普遍。CryTek 工作室[1]在 2006 年对云雾进行了数学建模的分析,近几年许多厂商也据此提出了基于顶点网格的形体建模以及基于点云或噪声贴图建模方式等其他体积云以及体积雾的解决方案,然后通过体素渲染[2]或光线步进的渲染方式以及基于物理的路径追踪方式进行渲染。其中,光线步进相较于体素渲染及路径追踪实现简单且开销较小,更适合渲染作为实时应用中远景的云层。

云层移动动画方面,基于物理的实现是通过模拟大气温度以及压强等属性变化来模拟真实的云层移动过程[3],但这种方法不适合对性能要求较高的实时应用如游戏中。地平线游戏制作组[4-5]的解决方案是让

收稿日期:2022-10-06　　**修回日期**:2023-02-09
基金项目:国家自然科学基金联合基金重点支持项目(U1803262)
作者简介:陈　烨(1998-),男,通讯作者,硕士研究生,研究方向为计算机图形学和虚拟现实;李顺新,教授,研究方向为计算机图形学和虚拟现实。

光线步进生成云层随高度偏移世界坐标,并且向上移动云层模拟云层的浮动效果,这种方法渲染出的云层效果与细节不够优秀但足够高效。龚昱宁等[6]在此基础上做出了改进,提出可控域扭曲建模云层来增加移动细节并使云层拥有更多的扭曲效果,这种方法本质是修改用于计算云层密度信息的噪声,在云层移动时依旧不会表现出向前形变的效果。

针对上述问题,该文在地平线组方案的基础上做出一些改进。为了让云层在移动时表现出符合风向的动画效果,提出用流动图来竖向偏移云层采样点的算法,整合了现有的云层渲染算法以及渲染提速算法,优化了地平线制作组方案的动画效果,增加了更真实以及细节更加丰富的前向翻腾效果。

1 相关工作

为了模拟更加真实的天空,云层渲染被广泛研究,并且不断有许多新的想法和突破。对云层模拟一般分成三个方面:云层建模、云层光照、动画模拟[7]。该文主要讨论云层建模以及动画模拟中的云层移动。

1.1 云层建模

相较于非真实感强烈的天空盒或广告牌的云层建模方法,游戏盗贼之海开发团队[8]使用顶点网格和噪声渲染云的基础形状。这种使用几何形体的渲染方法需要手动放置每一朵云,不仅不能很好地表现出云层的流动效果,还增加了显存和存储空间的占用。另一种常见的云层建模方法称为点云,点云的核心思想是将云看成多个单位诸如粒子的组合[9],通过温度、湿度以及风向等参数以及粒子之间的相互作用模拟云的生成过程。Nilsso[10]的方法是首先固定几个半径较大的粒子,然后使用一个随机数发生器来生成其他粒子的位置得到单个云的形状。Prashant[11]通过云图(cloudmap)控制点云参数。点云模拟云层生成过程需要一定的时间导致这种建模方式并不适用于云层的实时更新,且存在每一次参数改变都会带来较多的 GPU 和 CPU 的数据交互,难以满足实时云层移动渲染的渲染需求。因深度学习的兴起,除上述常见云层建模方法外,近几年也有将深度学习与云层建模相结合的工作,如通过神经网络根据云层照片生成云层模型[12],此方法虽不用手动建模,但和上述使用顶点网格的建模方法有着同样的缺点,即无法实时表现云层动态。

针对上述局限性,Schneider 等人[4-5]开发了一个新的云层建模方法,并在游戏《地平线:零之黎明》中使用,作者预计算分形后的云层密度到 3 维噪声中,根据 CryTek 工作室提出的模型,在光线步进的每一个点都对预计算的噪声采样来模拟当前位置云层的密度,并使用天气图控制云出现的位置及云层高度信息,最终得到云层模型。这种云层建模方式的缺点是不是基于物理的,无法模拟温度、风向等参数的改变所产生的变化,但可以通过噪声模 拟流动效果。Häggström[13]改进了控制建模的天气图,他增加控制云层密度的通道,通过调节参数能够实现云层从零到布满整个天空的过渡。Fablan B[14]结合光线步进与体素渲染,使近处的雾更加真实。兰未[15]讨论了该方法在移动端的使用且给出了在移动端的优化方案。

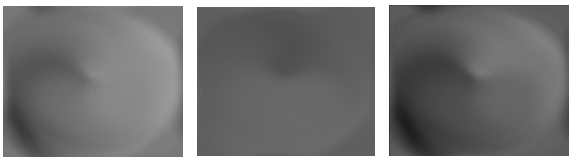
1.2 云层移动

云层移动受到大气湍流的影响在流动时表现出向前的翻滚,对云层的前向翻滚模拟可以渲染出更加可信的云层使整体画面的真实感更加强烈。但对其研究较少,更多的是建模和光照方面的更新,近几年只有地平线制作组和龚昱宁提出了他们的解决方案。

地平线制作组提出的云层移动解决方案是首先让采样点坐标随时间缓慢上升,用此坐标采样基础纹理以此来给云层一个整体向上流动的效果,然后使用卷曲噪声偏移采样点,用此坐标采样细节纹理来模拟边缘的抖动效果。龚昱宁等提出可控域扭曲云层湍流渲染解决方案是在地平线组方案上的优化,主要是增加云层本身的扭曲效果,使云层移动时细节更丰富。此方法首先根据给定频率振幅等属性采样域扭曲反 Worley 噪声得到 3 个用于后续计算的分量,然后根据分量偏移出采样点采样 Worley 及 Perlin-Worley 噪声得到云层扭曲前后的密度值进行混合。该方案没有密度上的预计算操作,需要在一次步进过程中进行大量的采样操作,如果后续将不同位置的扭曲频率振幅等属性整合成图,采样数还会继续增多。

1.3 流动图

流动图是一个记录向量场的纹理,图上的颜色记录该处向量场的方向。流动图一般用于表现平面上的流动效果,如水面的波浪和火焰的摇曳。流动图也可以用于表现云层的流动方向并以此模拟出极端天气如龙卷。Wang[16]在他们的形状可变的云层实现中引入了 3 维流动图来控制云层形状,而该文使用流动图创造云层在竖直方向上的偏移。图 1 为使用的流动图样例。



(a)流动图 (b)流动图 R 通道 (c)流动图 G 通道

图 1 流动图样例

2 文中方法

该文在云层建模实现流程上与地平线游戏制作组

提出的基本一致,但更改了流程中用于实现云层移动的算法,且使用时间上升频对交叉矩阵降低分辨率后的结果降噪,提升渲染效率。

流程分为以下 3 个步骤:

(1)根据交叉矩阵筛选出当前帧需要渲染的像素。

(2)从被选像素开始向云层光线步进。在光线步进过程中,使用流动图控制云层移动并累计云层透明度,然后将该透明度作为步骤 3 中的混合系数。

(3)使用时间上升频,按系数混合当前渲染结果与历史渲染数据作为当前帧输出,保存这个输出用于增强下一帧的渲染效果。

2.1　无缝流动

为了创造云层在竖直方向上的无缝偏移,该文放弃使用地平线组用于偏移细节噪声采样点的卷曲噪声,这种解决方案虽然能模拟出云层移动时部分流动效果但存在不足,对细节的偏移只能影响云层边缘,云层移动会出现主体的云没有动但边缘在快速抖动以及抖动不连续的现象,降低云层真实感。而使用流动图,在设置合理的情况下可以解决这一问题。流动图本质是一个二维的向量场,因此用于控制云层在移动时各部分的移动情况时会使云层整体形状而非仅边缘部分发生改变,而且能够很好地控制云层移动时的翻滚方向。对于不连续现象的产生,该文通过对两个相位的流动图采样叠加来实现无缝循环。对流动图的无缝采样步骤可总结如下:

(1)构造周期相同,相位差半个周期的波形函数 P_0, P_1。为了防止得到的偏移量随时间越来越大,使用 frac 函数将其限制在[0,1]之间。

$$\begin{cases} P_0 = \mathrm{frac}(\mathrm{Time} * \mathrm{speed}) \\ P_1 = \mathrm{frac}(\mathrm{Time} * \mathrm{speed} + 0.5) \end{cases} \tag{1}$$

(2)使用波形函数 P_0, P_1,周期化向量场方向,流动图本身记录的值的范围为[0,1],作为方向使用时需将其映射为[-1,1]。该文使用时间和云层采样点的高度百分比 p_h 作为流动图采样的原始位置。

$$\begin{cases} S_0 = \mathrm{texture}(P_0, p_h) * 2 - 1 \\ S_1 = \mathrm{texture}(P_1, p_h) * 2 - 1 \end{cases} \tag{2}$$

(3)将两个相差半个周期的结果进行加权混合,使一个周期结束到另一个周期开始的跳跃情况被另一层采样覆盖,从而得到一个连续的偏移量,作为云层该点的偏移指向。

$$\mathrm{offset} = \mathrm{lerp}(S_0, S_1, \mathrm{abs}(0.5 - P_0) * 2) \tag{3}$$

2.2　云层移动实现

云层受到大气中风的影响移动,风速会随着地区温度及海拔高度的变化而发生变化。通常来说,在地表一定区域内海拔越高风速越大,云移动也会表现出云顶移动的比底部快进而产生湍流现象。地平线组方案的做法是根据高度百分比来增加采样点偏移的速度,使云层看起来倾斜着向前运动,这种移动的问题是过于整齐,在边缘生成的云会整齐地朝前移动,产生明显的断层。为防止断层产生,用根据采样点高度百分比以及时间共同采样流动图的结果来偏移采样点,这种做法不仅能够使云层移速随高度改变,还能使云层整体形状根据流动图内容扭曲形变。

引入上述采样算法,则云层实时移动的步骤可表示为:

输入:采样点坐标 pos,风速信息(up,forword)。

输出:采样点的密度 density。

步骤一:根据采样点在云层中的高度 A_h 以及当前云层高度 C_h,计算采样点在云层中的高度百分比 p_h。

$$p_h = \mathrm{SAT}(\frac{A_h}{C_h}) \tag{4}$$

步骤二:按公式(3)采流动图并与风速信息一起应用于当前采样点,得到偏移后的采样点 pos´。

$$\mathrm{pos}' \mathrel{+}= p_h \times \mathrm{offset} \times (\mathrm{up}, \mathrm{froward}) \tag{5}$$

步骤三:使用偏移后的采样点位置对形状噪声(GetShape)和细节噪声(GetDetail)采样得到最终云层密度 d。

$$d = \mathrm{GetShape}(\mathrm{pos}') - \mathrm{GetDetail}(\mathrm{pos}') \tag{6}$$

2.3　时间上升频

光线步进算法中有多少像素被光线步进是影响性能消耗的最重要因素。公式(7)是文中方法所需总执行时间的近似。

$$T = n_p * \sum_{0}^{n_s} (t_s + t_m) \tag{7}$$

其中,T 表示总的执行时间,n_p 为需要渲染的像素数量,n_s 为沿观察方向的步进数,t_s 为每一步云层密度采样需要的时间,t_m 为控制云层移动所需的时间。

在 t_s 和 t_m 固定的情况下,减少每帧渲染的像素数 n_p 可以大幅优化性能。但是降低分辨率通常会大大降低图片质量,而使用时间上升频可以在渲染较低分辨率的前提下仍保留质量,时间上升频本质是利用历史渲染结果来增强当前帧的渲染效果,将历史渲染结果与当前结果按一定关系与比例混合。

该文使用 4×4 交叉矩阵 $\boldsymbol{M}$ 降低所需渲染的像素数,每一帧按 $\boldsymbol{M}$ 中的顺序渲染每个 4 × 4 像素块的一个像素,即每帧都以十六分之一的分辨率渲染整个画面。对于每个 4 × 4 的像素块,使用交叉矩阵 $\boldsymbol{M}$ 来控制其渲染顺序而不是按顺序渲染来避免产生较强的割裂感。

$$\boldsymbol{M} = \begin{bmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{bmatrix}$$

上升频步骤可总结如下：

(1)使用逆视图投影矩阵($\boldsymbol{M}_{\text{view}} \cdot \boldsymbol{M}_{\text{proj}}$)$^{-1}$从剪辑空间坐标 P_{cs} 计算世界坐标 P_{ws} 。

$$P_{\text{ws}} = P_{\text{cs}} \cdot (\boldsymbol{M}_{\text{view}} \cdot \boldsymbol{M}_{\text{proj}})^{-1} \tag{8}$$

(2)通过上一帧的图投影矩阵($\boldsymbol{M}_{\text{view}} \cdot \boldsymbol{M}_{\text{proj}}$)计算 P_{ws} 在上一帧的剪辑空间坐标 P'_{cs} 。

$$P'_{\text{cs}} = P_{\text{ws}} \cdot (\boldsymbol{M}_{\text{view}} \cdot \boldsymbol{M}_{\text{proj}}) \tag{9}$$

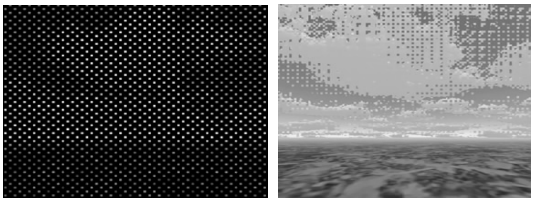
(3)根据 P'_{cs} 采样历史帧结果。

$$C' = \text{texture}(P'_{\text{cs}}) \tag{10}$$

(4)使用当前帧计算得出的透明度 α 作为当前帧与历史结果混合系数进行混合。

$$C_{\text{final}} = C \cdot \alpha + C' \cdot (1 - \alpha) \tag{11}$$

图 2 为单帧渲染画面以及使用时间上升频后的结果。



(a)单帧渲染画面 (b)时间上升频结果

图 2 时间上升频

3 实验结果

3.1 实验环境

实验在 Unity(2021.3.5fc1)上进行,分别使用了 1 920 * 1 080 和 3 840 * 2 160 的渲染分辨率与 GTX3070 显卡。

3.2 云层移动效果

为了说明文中解决方案在云层移动表现效果上具有一定优势,将文中方案与地平线组提出的解决方案进行对比,而与龚昱宁提出的解决方案进对比时将会与原论文实现图片比较。其中基于流动图的效果图为文中方案实现效果,对照组效果图分别为地平线组方案实现效果以及从域扭曲方案原文中截取的效果图。为方便观察云层整体形状与边缘细节,从云层侧方进行了对比,并且每 10 帧记录一次云层变化,然后按时间顺序拼接。观测结果如图 3 和图 4 所示。

将上述实验结果局部放大后得到如图 5 所示的结果图,其中图 5(a)为文中方案实现结果局部放大图,图 5(b)为地平线游戏组方案的实现结果局部放大图,图 5(c)显示了局部截取的位置。由图 5(b)可以看出,地平线组方案实现云层移动时会在边缘产生不连续的情况,表现为边缘云层突然从后方出现并向前移动,在连续的移动中这种情况带来的人造感尤为明显,此外,这种方法实现的云层形状不会在移动时发生改变。相较于上述结果,文中方案不仅能够实现连续的移动且能够在云层边缘处表现出朝风向的形变,从而在连续的移动中表现出前向翻滚细节,见图 5(a)。

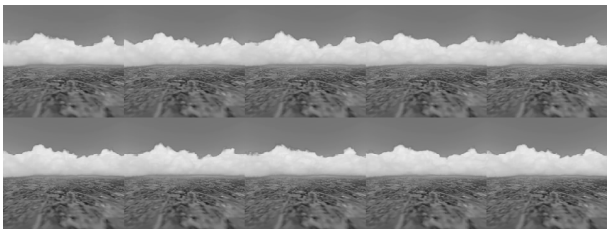


图 3 文中方案实现效果

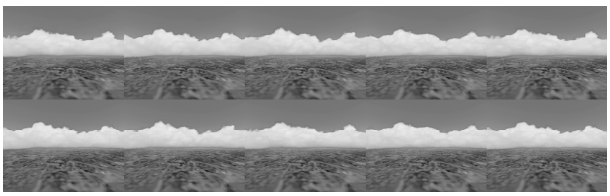
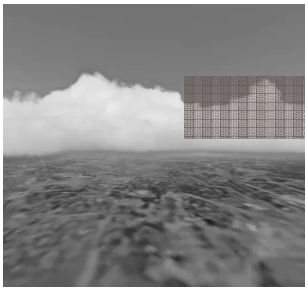


图 4 地平线组方案实现效果

(a) 文中方案实现结果

(b) 地平线组方案实现结果



(c) 对照结果

图 5 局部放大结果

图 6 为域扭曲方法原文[6]截取的效果图,图 7 为文中方案在类似角度观察结果。龚昱宁[6]在论文中表示其方案实现云层移动湍流细节优于低频线组方案,而对比图 6,文中方案拥在渲染时间小于域扭曲方案的同时有其方案类似的细节表现。

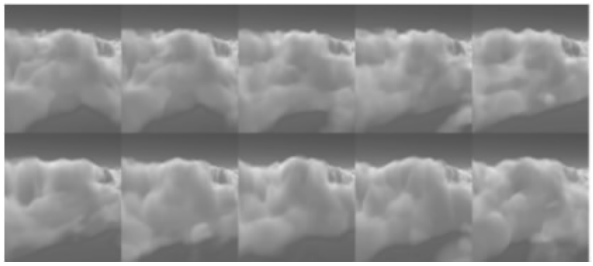


图 6 域扭曲原文截取结果

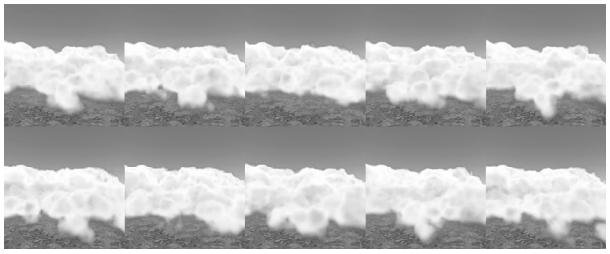
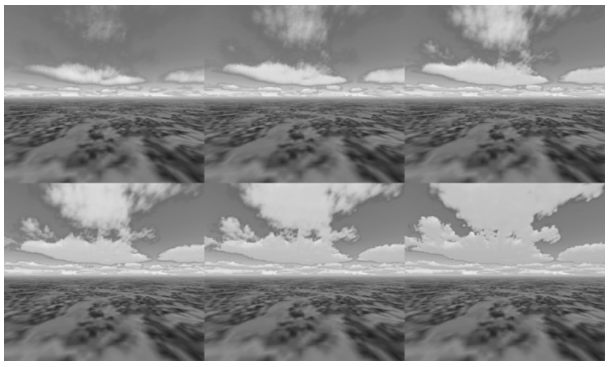


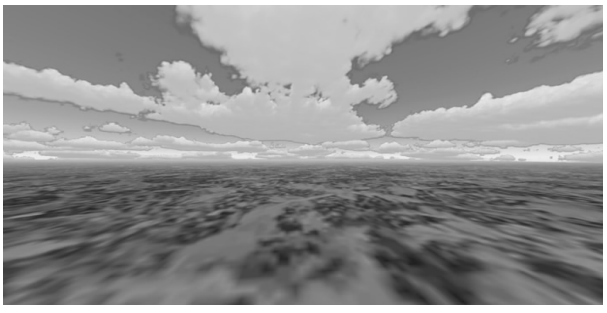
图 7　文中方案实现结果

3.3　性能消耗

光线步进是一个十分耗时的过程,如果不进行性能优化很难在实时应用中使用。由公式(8)可知,渲染耗时与光线步进的次数以及渲染像素数量相关,因此首先对比了不同步进次数渲染一帧的效果,以此得到渲染出可接受云层效果的最低所需步进次数。图 8 是分辨率为 1 920 * 1 080 时步进次数依次为 8,16,32,64,128,256 时的实验局部结果图。对比图 8(a)、图 8(b)可以看出,当步进数为 64 时,文中云层渲染方法虽部分位置细节仍有不足但效果已经能够接受。



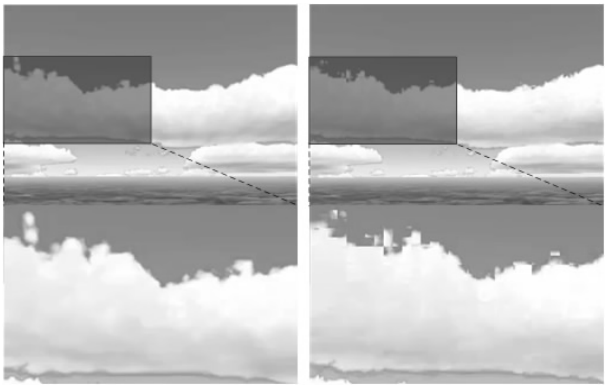
(a)步进次数为 8，16，32，64，128，256 的实验结果图



(b)步进次数为 512 的参考图

图 8　不同步进次数的效果

图 9 为直接渲染结果与使用交叉矩阵减少渲染像素然后使用时间上升频方法提升云层渲染质量后的结果,表 1 为两种方法在步进次数为 8,16,32,64 时的耗时。根据图 9 渲染结果以及表 1 中数据可以看出,使用交叉矩阵和时间上升频方法会在云层边缘处出现失真,但渲染效率大幅提升,在步进次数为 64 时直接渲染需要的时间为 12.7 ms,而文中方法仅需 1.6 ms。由于云是一种低频的信息,在帧率足够的情况下边缘处的失真不会破坏云层整体的真实性。



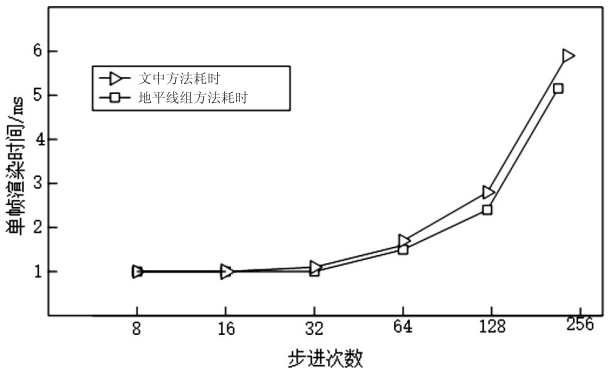
(a)直接渲染　　(b)交叉矩阵+时间上升频

图 9　直接渲染与使用交叉矩阵+时间上升频结果对比

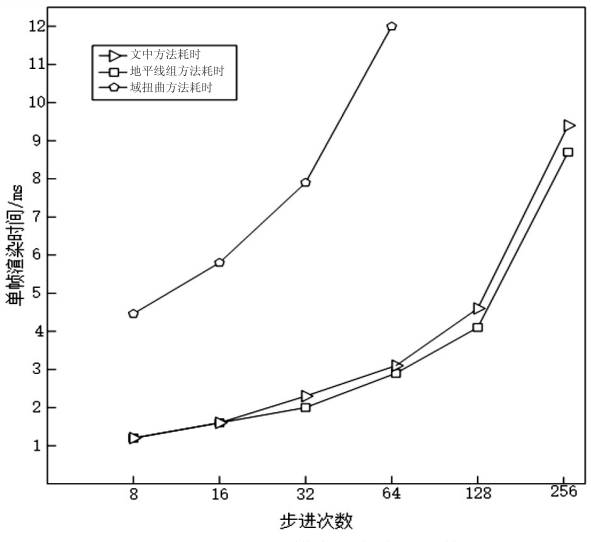
表 1　云层渲染所需时间

方法	8 步	16 步	32 步	64 步
交叉矩阵+时间上升频	1.1	1.2	1.2	1.6
直接渲染	3.0	5.4	8.1	12.7

最后,对比了在不同分辨率以及不同步长的情况下三种渲染方法都应用交叉矩阵以及时间上升频提升渲染效率的耗时,如图 10 所示,其中域扭曲渲染方法所需时间来自原文给出的参考时间。



(a)1 920*1 080 分辨率下各步进次数耗时



(b)3 840*2 160 分辨率下各步进次数耗时

图 10　不同分辨率及步进次数下渲染耗时

图 10(b)展示了三种方法渲染一帧的耗时,域扭曲方法所需时间明显高于文中方法所需时间,虽然使用设备不同,但此耗时仍具有一定参考价值,因为需要控制云层扭曲,域扭曲方法没有对云层密度预计算,因此耗时会远高于使用了预计算的地平线组方法以及文中方法。在步进数为 64 时,文中云层渲染方法虽部分位置细节仍有不足但效果已经能够接受,在渲染分辨率为 1 920 * 1 080 时耗时 1.6 ms 仅比地平线组的耗时增加了 0.1 ms,十分高效。即使在对画面有着更高要求的场景如 4k 分辨率下,渲染速度依旧可以达到 3.1 ms,如图 10(b),比地平线组耗时增加 0.2 ms,仍然十分高效。

4 结束语

提出了使用流动图对云层进行竖直方向上的偏移,使流动云层能够在移动时表现出更加真实的前向翻滚细节,且比较了该方法在不同分辨率以及步进次数下的性能消耗和渲染效果。实验结果表明,所提云层移动方法相较于另外两种方法有着更加真实的云层随风形变细节。使用交叉矩阵以及时间上升频来提升云层渲染效率后在步进数为 64 时能够达到 1.6 ms 每帧,仅比地平线组方法高 0.1 ms,十分高效。但上述结论均建立在拥有一个设置合理的流动图的前提之下,后续将考虑程序化生成一个合理的流动图。

参考文献:

[1] WENZEL C. Real-time atmospheric effects in games[C]// ACM SIGGRAPH 2006 courses. Boston: ACM, 2006: 113-128.

[2] PETTERSSON J. Real-time rendering and dynamics of sparse voxel octree clouds[D]. Lund: Lund University, 2020.

[3] HÄDRICH T, MAKOWSKI M, PAŁUBICKI W, et al. Stormscapes: simulating cloud dynamics in the now[J]. ACM Transactions on Graphics, 2020, 39(6): 1-16.

[4] SCHNEIDER A. Real-time volumetric cloudscapes[M]// GPU pro 360 guide to lighting. [s. l.]: AK Peters/CRC Press, 2018: 473-504.

[5] SCHNEIDER A. Nubis, authoring real-time volumetric cloudscapes with the decima engine[C]//Eurographics. The Netherlands: Eurographics Association, 2018: 27-90.

[6] 龚昱宁,张严辞,陈学超,等. 基于可控域扭曲和有序抖动降噪的高性能云层湍流渲染[J]. 重庆理工大学学报:自然科学版, 2021, 35(1): 118-128.

[7] GOSWAMI P. A survey of modeling, rendering and animation of clouds in computer graphics[J]. The Visual Computer, 2021, 37(7): 1931-1948.

[8] ANG N, CATLING A, CIARDI F C, et al. The technical art of sea of thieves[C]//ACM SIGGRAPH 2018 Talks. Vancouver: ACM, 2018: 1-2.

[9] FERREIRA BARBOSA C W, DOBASHI Y, YAMAMOTO T. Adaptive cloud simulation using position based fluids[J]. Computer Animation and Virtual Worlds, 2015, 26(3-4): 367-375.

[10] NILSSON F. 3D cloud visualization in real-time[D]. Västerbotten: UmeåUniversity, 2022.

[11] GOSWAMI P. Interactive animation of single-layer cumulus clouds using cloudmap[C]//2019 Italian chapter conference-smart tools and apps in computer graphics, STAG 2019. Cagliari: Eurographics - European Association for Computer Graphics, 2019: 71-78.

[12] ZHANG Zili, CEN Yunchi, ZHANG Fan, et al. Cumulus cloud modeling from images based on VAE-GAN[J]. 虚拟现实与智能硬件(中文), 2021, 3(2): 171-181.

[13] HÄGGSTRÖM F. Real-time rendering of volumetric clouds[D]. Västerbotten: UmeåUniversity, 2018.

[14] FABLAN B. Creating the atmospheric world of red dead redemption 2. advances in real-time rendering in 3D graphics and games[C]//Proc. of the ACM SIGGRAPH. Los Angeles: ACM, 2019: 51-57.

[15] 兰 未. 动态体积云的建模、实时渲染及移动端加速方法[D]. 杭州:浙江大学, 2020.

[16] WANG Y, MCGRAW T. Art-directable cloud animation[C]//International symposium on visual computing. [s. l.]: Springer, 2021: 392-399.