Study on Modeling and Real-time Rendering of Snow Scene

Author's signature:_____

Supervisor's signature:_____

External Reviewers: _____

_____

_____

_____

_____

Examining Committee Chairperson:

_____

Examining Committee Members:

_____

_____

_____

_____

_____

Date of oral defence:_____

浙江大学研究生学位论文独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得浙江大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：　　签字日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解浙江大学有权保留并向国家有关部门或机构送交本论文的复印件和磁盘，允许论文被查阅和借阅。本人授权浙江大学可以将学位论文的全部或部分内容编入有关数据库进行检索和传播，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后适用本授权书）

学位论文作者签名：　　导师签名：

签字日期：　　年　　月　　日　　签字日期：　　年　　月　　日

摘 要

自然场景的真实感实时模拟，一直是计算机图形学领域研究的难点和热点。而气象景象，如：风、雨、雾、霾、雪、沙尘暴等，与人们的生活息息相关。其中，雪场景的真实感建模与实时绘制技术，其研究进展在虚拟现实、军事演练、运动仿真、雪灾的预防和救援、影视特技及游戏设计等众多领域有着广泛的应用价值。本文对雪场景的建模与实时绘制展开研究，主要内容有以下两个方面：

1) 针对现有方法难以同时实时生成大规模动态雪场景的积雪及飘雪效果的问题，本文提出并实现了一种交互式大规模雪场景建模与实时绘制的新方法。为了精细地模拟场景的积雪效果，提出一种基于视点的自适应降雪遮挡图模型，该模型能在实时更新地物的遮挡关系的同时，大大减少大规模雪场景中积雪的计算量并提高了计算精度；对于场景的飘雪，采用一种基于视点的雪粒子分层建模技术来减少雪粒子数量，将视点变换及降雪粒子系统移至 GPU 中进行加速计算；采用动态多旋转纹理来模拟飘落雪花的形状以增加其真实感；采用几何与纹理混合绘制的方式以减少大场景的复杂度，最终成功地实现了野外和城市两个大规模雪场景的实时漫游，在场景中可看到压雪累累的树枝雪挂景象及轿车背上厚厚积雪等冬天美景。

2) 雪崩是冬天山区经常出现的景象，对滑雪等冬季运动的爱好者对说是一种灾难性的现象。但目前计算机图形学领域对此景象的模拟却少有涉及。为此，本文提出并实现了一种基于计算流体动力学的实时雪崩模拟的新方法。该方法在平滑粒子动力学 (SPH) 计算框架的基础上，首先根据雪崩形成的物理机制，采用非牛顿流体模型对雪崩运动过程进行建模；再考虑到雪崩运动过程中粘滞力的流变特性，以速度张量 Cross 模型来有效地表示粘滞力的时变；采用了未知因素修正的 SPH（XSPH）模型来维持雪崩运动的稳定性。最后通过优化设计的 GPU 加速方案，成功地实现了百万量级粒子的雪崩景象的实时模拟。

在论文的最后，作者对本文的研究工作进行了总结，并指出了本课题进一步研究的方向。

关键词： 自然场景模拟，大规模雪场景，积雪和飘雪，基于视点的自适应降雪遮挡图，实时绘制，雪崩景象，非牛顿流体，时变粘滞力，GPU 加速

Abstract

Real-time rendering of nature scene is a challenging topic in research of computer graphics. Meteorological scenes such as wind, rain, fog, haze, snow, sandstorm, etc are highly related to the people's daily life. Modeling and rendering of snow scene with high realism has wide applications in the areas such as virtual reality, military drill, snowy disaster's prevention and rescue, film special effects and game design, etc. In this thesis, the main research work is related with modeling and real-time rendering of snow scene. The following two items are the key contents of this thesis:

1) Current methods cannot simultaneously simulate both the snow accumulation and falling effects for large-scale dynamic snow scenes. To solve this problem, we propose a new method of modeling and real-time rendering of snow scene. To simulate the accumulation of snowflakes on different objects in the scene precisely, we present an adaptive occlusion map for the falling snowflakes, with respect to the object distribution within the current view frustum. The occlusion map will be updated in real-time allowing dynamic objects to be included in the scene. To simulate the huge amount of dynamically falling snow flakes, we adopt a view-dependent particles grouping technique to reduce the amount of the snowflakes, meantime move the view transform and evolution of snow particle system into GPU for acceleration. To enhance the realism of falling snowflake, we adopt a dynamic texture sequence of multi-rotary snowflake. Besides, a hybrid approach of geometry and texture is employed to render the distant view of large-scale snow scenes with less computation. Based on the above approaches, we successfully implemented the real-time walkthrough of large-scale snow scenes including both urban and suburb circumstances, and generate the beautiful winter view such as heavy snow accumulation on the tree branches and the covers of cars, etc.

2) Snow avalanche is a natural phenomenon that typically occurs in mountainous terrain, it is a catastrophic phenomenon for winter sports enthusiasts like skaters. But in Computer Graphics area, there are rarely works about snow avalanches. We propose a new method based on CFD(Computational Fluid Dynamics) to real-time simulate this phenomenon. This method is ran under the general GPU framework of SPH (Smoothed Particle Hydrodynamic). Because of phys-

ical mechanism of snow avalanches, we use Non-Newtonian Fluids Model to describe the motion of the avalanche, considering the rheological behavior of snow avalanche, Cross Model(a type of rheological function) is used to simulate the time-variant characteristic of viscous force. To keep stability of snow avalanche,we use the XSPH technique (X factor correction of SPH). Besides, Using the technology of GPGPU in High-end graphics display card, we successfully implemented the real-time snow avalanches simulation.

At the end of this thesis we conclude our work , and presents some limits which can be studied in the future.

Keywords: natural phenomenon simulation, large-scale snow scene, accumulative and falling snow effects, view-dependent adaptive falling snow occlusion map, real-time rendering, snow avalanches stimulation, Non-Newtonian Fluids, time-variant viscous force, GPU acceleration

目 录

图目录

第1章 绪论

1.1 引言

本章主要阐述了本文的选题背景，介绍了自然场景模拟特别是大规模雪场景以及雪崩现象的模拟的应用及研究意义，回顾了在计算机图形学领域中对雪场景研究的发展历史及现状，具体介绍了积雪、降雪研究工作的进展，并介绍了地质学等领域对雪崩景象仿真的相关知识及计算模型。最后介绍了本文的工作及结构安排。

1.2 课题背景

自然场景的真实地实时模拟，一直是计算机图形学研究的热点和难点。现阶段自然场景的模拟大致分为以下几类：花草树木、地形植被的模拟[4-6]；雾、雨、雪、沙尘暴等天气现象的模拟[7-9]；海洋、湖泊等大尺度水场景的模拟[10-12]；奇特自然景象如：龙卷风、极光、海市蜃楼、佛光等的模拟[13-15]，而气象场景特别是大规模雪场景的真实感建模和绘制技术，在影视动画、数字娱乐、教育科普、军事仿真等领域中有着广泛的应用。因此，如何真实感地实时绘制出大尺度动态雪景象，则是解决此问题的关键技术所在。

雪场景往往给人一种美好的感觉：漫天飞舞的雪花使人有浪漫之印象；银装素裹的树枝雪挂景象（如图 1.1 所示）更是北国特有的奇观；夜晚，皑皑白雪环抱中的小木屋所映照出的灯光总给夜归人以家的温馨感。然而，狂风暴雪也会预示着环境的恶劣和艰难。大规模雪场景的实时绘制技术在军事演练、极端气象条件下武器装备的性能仿真、雪灾的防治和救援等方面也有着重要的应用价值。

在影视产业中，特效制作起着越来越大的作用。这往往是好莱坞大片的必杀技，极大地增强了美国文化影视产业对世界其它国家的竞争优势。并有借此宣传推广自己的价值观念而压迫其它文化种类的趋势。在影片《木乃伊归来 3》（如图 1.2）中惊险刺激、气势磅礴的 3D 雪崩特效吸引了全世界影迷的眼球。另一方面在游戏领域中，随着图形学技术的发展，场景的建模和绘制越来越逼真，极大的加强了玩家的用户体验，让玩家有身临其境之感。在 Electronic Arts 公司出品的最新的极品飞车 (need for speed) 游戏中，加入了一条

雪崩赛道（如图 1.3 所示），非常惊险刺激，充分显示了速度与激情，玩家大呼过瘾。

图 1.1 银装素裹的冬日美景

图 1.2 《木乃伊归来3》（即：《Mummy 3》）中的雪崩剧照[2]

此外，雪场景的模拟在防灾救灾中也中起着极其重要的作用。特别对于我国来说幅员辽阔，气候多变。2008 年的特大雪灾[1]让我们见识到了雪的强大威力，而雪崩发生的突然性和巨大的破坏力，每年都会给全世界带来巨大的财产损失和人员伤亡，因此对于雪崩景象的真实感实时模拟，也可以为大众提供一种直观的体验，为大众的避灾自救教育开创一种全新的途径。

[1]http://zh.wikipedia.org/wiki/2008 年中国雪灾

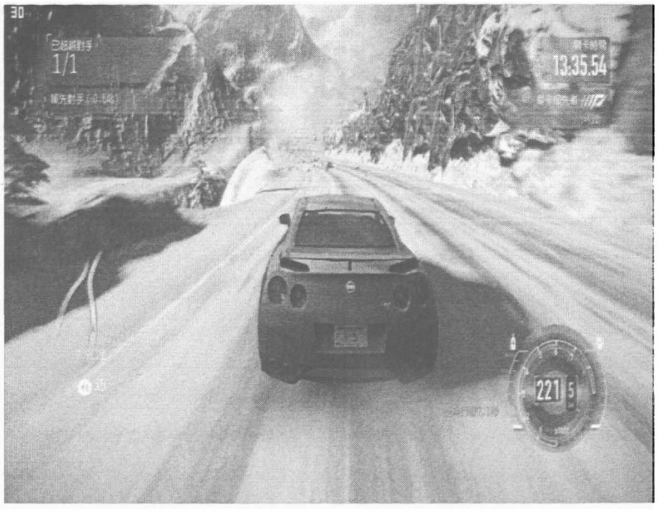


图 1.3 《极品飞车 17》中的雪崩赛道

1.3 国内外研究现状

在计算机图形学领域中，对雪场景的建模与绘制的研究工作主要分为积雪和降雪两个方面。对于降雪和积雪模拟和绘制，目前国内外已经有不少的研究工作。对于雪崩现象的模拟，在计算机图形学领域中，至今鲜有见到，目前国内外对雪崩模拟的研究，主要是地质学界关于雪崩现象产生的触发条件、气象地质因素及数值模拟计算上。下面我们将分别介绍积雪、降雪及雪崩模拟方面国内外已有的相关研究工作。

1.3.1 积雪的研究

关于积雪方面的研究工作，Nishita 等[16] 以 Metaball 模型对积雪场景进行建模，并计算光线在雪表面的散射效果来绘制积雪效果。Fearing 等[17] 所提出的降雪模型，是通过向天空发射粒子来计算确定每一表面点的积雪厚度，并通过稳定性模型将部分雪粒子从积雪不稳定处移位至稳定处，绘制出雪后晴日美妙的积雪景象。Premoze 等[18] 在模拟高山地形在不同季节积雪融雪景象时，采用了地形高度数据并结合全色航空图构建出地形的几何与基本纹理，再根据较为复杂的地理气候模型计算出不同季节积雪区域的变化，将预计算得到的积雪区域叠加至地形的基本纹理上，从而展示山地不同季节的景观变化。但此模型没有考虑积雪的厚度的变化。陈彦云等[19] 采用了位移映射和体纹理映射技术构造出较为逼真的积雪场景，但没能实现树上雪挂等不同高度景物上积雪效果。Felman 等[20] 通过求解 3D Navier-Stokes 方程来模拟地面的积雪效果。Thiis 等[21] 用数值计算的方法对建筑物附近

的积雪效果进行了模拟，但其绘制效果尚有待提高。

以上工作注重于积雪效果的真实感或数值模拟，由于其计算模型的复杂性，无法达到实时绘制的效果。为此，Haglund 等[22] 提出一种实时计算积雪厚度的方法。通过网格记录雪花落到地面的高度，再按照各点的高度来生成三角形的积雪面片，这种方法可较方便的调整积雪的精细度，其缺点是在前期建模过程中需交互地在地形表面划分网格，工作量巨大，且网格划分的随意性较大，难以精细考虑分布在每一网格内景物上的积雪效果。Moeslund 等[23] 在 Haglund 的基础上提出了网格构造的改进方式，首先在粗网格内来统计雪粒子的数量，再对网格进行细化和合并，效果有所改进，但其动态调整网格的策略则会影响系统运行的效率。Ohlsson[3] 在 CPU 中通过绘制引擎接口获取以天空为视点的场景的深度图，以此来预先计算生成整个场景的降雪遮挡图，再获取顶点法向量图、3D 噪声图及光照映射图，然后将这三张图传输到 GPU 进行纹理存储，在 GPU 中利用遮挡图生成暴露因子来实时计算绘制积雪颜色。该方法的缺点是场景的降雪遮挡图是固定的，不能进行实时的更新，因此如果有新增加的地物，就不能表现出其表面的积雪效果；另一方面，由于降雪遮挡图采样精度的限制，只适用于小场景且地物顶点数较少的情况下，若应用于大规模场景中，则可能出现区域走样。

1.3.2 降雪的研究

对于降雪方面的研究工作，早期 Reeves 等[24] 采用简单的粒子系统来模拟降雪过程，但其场景的真实感不够理想。Langer 等[25] 采用逆向 Fouirer 变换对冬季场景背景图像进行处理，先在频域产生一系列表面点，再通过逆向 Fouirer 变换，通过一个不透明度函数来形成雪粒子效果，最后通过循环形式产生降雪的效果。但该方法从本质上说是 NPR 式的二维图像处理技术。王长波等[8] 考虑了风力驱动的作用，采用了离散化的 Boltzmann 方程，用网格方式加入了风粒子和雪粒子的相互作用以达到实时飘雪的效果。但该方法未考虑视域内雪粒子的分层显示，难以实现大规模三维雪场景的实时漫游。刘波等[26] 在城市场景漫游中以粒子系统产生飘雪景象，但在地面、植物及建筑物表面的积雪效果尚缺乏量化的层次感。

若要在大规模动态雪场景中实现数量巨大的雪粒子的实时飘雪，传统的 CPU 计算绘制架构难以胜任。近年来，随着计算机图形硬件 GPU 编程技术的飞速发展，越来越多的研究者开始利用 GPU 强大的并行处理能力来加速复杂场景的计算及绘制。Lutz[27] 使用异构的 CPU+GPU 系统来加速粒子系统的计算。其主要策略为：在 CPU 中初始化粒子的状态，管理粒子的生成和消亡；而在 GPU 中把实时的粒子速度和位置信息存储在纹理

中、最终将纹理转化为顶点数据来绘制，实现了百万粒子量级的动态场景实时绘制效果。Smith[28] 在 Lutz 的基础上进一步开发了 GPU 的大规模并行处理能力。只是在初始化阶段运用 CPU 生成一些随机数据点集，把这些数据点集存储在 GPU 中的显存以后，剩下的计算工作全部交给 GPU 来完成，实现了多绘制进程的同步合作，大大提高了计算效率。

但至今，对于同时能产生积雪和降雪效果的大规模雪场景的实时生成方面的研究工作，尚很少见到。

1.3.3 雪崩的研究

雪崩以其突发性、潜在性、难预测性为人们所熟知。因为它的速度快和冲击力大等特点，已经严重影响到山区居民的生命和财产安全，并影响到交通网络、基础设施、农林畜牧及冰雪旅游业的可持续发展，目前，冰雪项目作为一种健康的冬季休闲娱乐运动方式，越来越受到大众的喜爱并纷纷加入。而雪崩对于冰雪运动特别是滑雪运动则是一种潜在的危害，而雪崩一直是地质学、气象学、灾害学领域的研究热点。目前在雪崩形成机制、动态模拟、抛程、预防与防治、风险评价与区划等领域取得了较大的进展。

雪崩的分类方法有很多种，根据雪崩始发区雪层特征，雪崩可以分为松雪雪崩和雪版雪崩两类[29]。松雪雪崩突发于一个相对有粘性的干湿雪层，从一点开始，而雪板雪崩是较厚和较硬板块雪层下相对稀疏的松雪层在陡坡上的断裂和崩塌[30]，起动方式由一条线开始，而雪板雪崩触发因子包括近表面局部载荷的快速增加 (如降雪) 和雪层特性空间变化 (如表面增温)[31]。按雪层含水状态，雪崩又被分为干雪雪崩和湿雪雪崩两类，干雪崩常发生于山顶，而湿雪崩常发生于中低山[32]。

雪崩的动态数值计算模拟始于 20 世纪 30 年代，其目的在于预测雪崩发生概率和估计雪崩始发区雪深、雪崩冲击力、流速、抛程等[33]。雪崩与积雪结构及其稳定性息息相关。雪崩动态模拟的前提是要对积雪结构与稳定性进行监测和模拟。积雪稳定性又由雪层张力和应力决定。雪层张力和应力随时间演变，当雪层应力大于强度张力时，雪层稳定性失衡，进而引发雪崩释放。雪层张力和应力变化率已在持久性稳定松雪层进行过模拟。Jamieson 等[34] 提出了积雪稳定性指标的计算模型。雪崩动态模拟模型可以在给定雪崩释放情况下计算出雪崩抛程，提供雪崩冲击力估算值。在实践中，动态模型大部分被简化，使用含两个摩擦参数的流变模型。摩擦参数的合理值可由已知雪崩历史事件抛程反算获得。对于不同条件参数的设置，则由过去雪崩规模大小、路径海拔、轨迹和雪崩周期决定[35]。

而然，上述的研究工作要么侧重于雪崩的预测，要么侧重于雪崩的数值模拟，但是在

计算机图形学领域，目前对雪崩景象的真实感建模与绘制方面的工作尚未见过。而游戏业及影视工业界虽对雪崩特技有过模拟，但其实现的具体技术细节则不得而知。并且，为追求视觉效果，其雪崩效果往往过于夸张，不一定符合其发生的物理规律。

1.4 本文的工作

针对上述前人工作的不足，本文展开了对雪场景建模与实时绘制新方法的研究。本文的工作主要分为两个方面：一是提出并实现了一种交互式大规模雪场景建模与实时绘制的新方法；二是基于地质学原理初步实现了对雪崩景象的实时模拟。本文的主要工作可以概括为：

- 提出一种基于视点的自适应降雪遮挡图模型来精细地模拟场景的积雪效果，该模型可以克服以往全局遮挡图模型在大尺度区域中积雪精度差的缺点；并采用一种基于视点的雪粒子分层建模技术来有效减少场景中降雪粒子的数量，采用几何与纹理混合绘制的方式以减少大场景的复杂度，从而实现了大规模积雪飘雪场景的实时漫游。

- 提出并实现了一种基于计算流体动力学的实时雪崩模拟新算法。该算法基于平滑粒子动力学 (SPH) 计算框架，采用了未知因素修正的 SPH（XSPH）模型来维持雪崩运动的稳定性。根据雪崩运动过程中粘滞力的流变特性，以非牛顿流体模型对雪崩运动过程进行建模，采用速度张量 Cross 模型来有效地表示粘滞力的时变，从而初步实现了雪崩动态景象的实时绘制。结合 SPH 模型、加入流变方程，采用 GPGPU 的技术，优化加速了经典 SPH 的计算框架，成功实现了雪崩场景的实时模拟。

本文的主要内容组织安排如下：

本文第 2 章，详细介绍了大规模雪场景的实时绘制工作，着重介绍了基于视点的自适应降雪遮挡图积雪模型和基于视点的分层粒子飘雪技术，最后介绍了大规模雪场景的实时绘制方法并呈现相应的绘制结果。

本文第 3 章，详细介绍了雪崩场景的实时模拟的过程，首先介绍雪崩的分类和建模方法，接着简述 SPH 流体模拟的基本原理及运动模型，然后讨论了根据雪崩特点所采用的非牛顿流体建模的过程，最后阐述了基于 GPU 加速的雪崩场景实时模拟的实现。

在本文的最后，作者对全文工作进行了总结，并指出了目前尚存在的问题及进一步深入的研究方向。

1.5 本章小结

本章首先介绍了本文的选题背景，阐述了大规模雪场景实时模拟和雪崩场景实时模拟的重要意义。然后回顾了计算机图形学领域对于雪场景(包括积雪、飘雪)建模与绘制的相关工作，以及地质学等相关领域对雪崩现象以往的数值模拟建模工作，最后介绍了本文的主要工作。

第2章 大规模雪场景的实时绘制

2.1 引言

大规模雪场景的真实感实时绘制在虚拟现实、雪灾的预防和救援、军事仿真及游戏设计等领域有着广泛的应用价值，但现有方法难以同时生成大规模雪场景建模与实时绘制的新方法。为此本章提出并实现了一种交互式大规模雪场景建模与实时绘制的新方法。首先阐述适合于大尺度场景的基于视点的自适应降雪遮挡图的交互积雪效果模型；然后讨论了飘雪效果的高效生成；再论述了大规模雪场景实时生成的方法，并展示了绘制的结果。

2.2 基于视点的自适应降雪遮挡图积雪模型

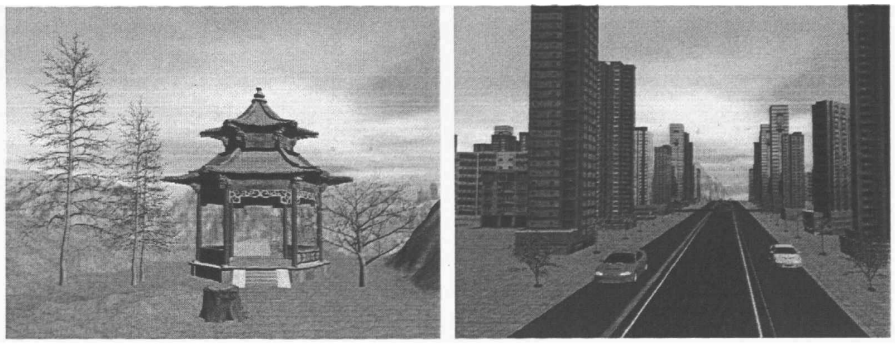
本节中我们首先描述雪场景的构建，再讨论针对大规模雪场景的自适应降雪遮挡图的建立；然后阐述积雪堆积效果的交互实现。下面分别加以讨论。

2.2.1 雪场景的构建

本文首先利用商用建模软件 3DSMax 建造起两个野外及城市大规模场景，如图 2.1，并对不同的地物特征，如：路面、建筑物、不同种类的树木、亭子、山体等根据其积雪效果，建立起不同精细度的三维几何模型，对距离不同的物体进行 LOD（层次分辨率模型）的处理，并对远处景物采用几何与积雪纹理相结合的方法，以简化场景的复杂度。这就为后面的实时绘制打好了基础。

2.2.2 自适应降雪遮挡图的建立

文献[3]对整个场景采用俯视深度图来判断地物相对于天空的遮挡关系。该方法根据顶点来进行积雪，在 GPU 顶点着色器的操作中，通过竖直方向拉伸顶点来表示积雪的最新位置和厚度。在片段着色器中，根据预先生成的自适应降雪遮挡图，来确定片段的暴露因子，计算拉升后顶点的积雪颜色，最终将原来片元的颜色和积雪片元的颜色混合，绘制出



(a) 野外场景 (b) 城市场景

图 2.1 3DSMax 建造的野外场景和城市场景

最终的积雪场景。虽然该方法把原先需要在 CPU 中确定的积雪位置转化为到 GPU 中进行顶点拉伸，再在 GPU 中进行顶点颜色计算。但在实际应用中，还是存在许多问题：

1. 当处理大规模的场景时，场景中地物高低不一，在计算降雪遮挡图时，很难设定统一的俯视点位置。
2. 遮挡图的分辨率有限，难以表现树枝、草叶上较为精细的积雪效果。
3. 由于降雪遮挡图是预计算生成的，对于场景中新加入的物体，会产生错误的积雪效果。

为此，本文提出了一种基于视点自适应降雪遮挡图的新方法来改善积雪效果。本文根据当前视域四棱锥的位置，来动态地调整降雪遮挡图的采样区域，并在每一帧中动态更新该遮挡图，从而能够实时计算采样区域内的遮挡关系，这样就能实现大规模场景的实时地物积雪厚度和积雪阴影的精细模拟。整个过程分为如下三步：

Step1 根据当前视域四棱锥参数来动态确定遮挡图的采样区域，如图 2.2，利用绘制引擎的接口实时获取采样区域内的局部俯视深度图，绘制到纹理中，把它作为当前一帧遮挡图参数传入 GPU。

Step2 利用 GPU 可以在顶点程序直接读取纹理的特性，直接在顶点程序中进行偏移的计算，相比文献[3] 的方法，可少绘制一遍，从而提高了绘制的速率。

Step3 在像素程序中利用 Blinn-Phong 光照明模型，进行光亮度的计算，绘制出积雪的效果 (具体流程图请见图 2.3)。

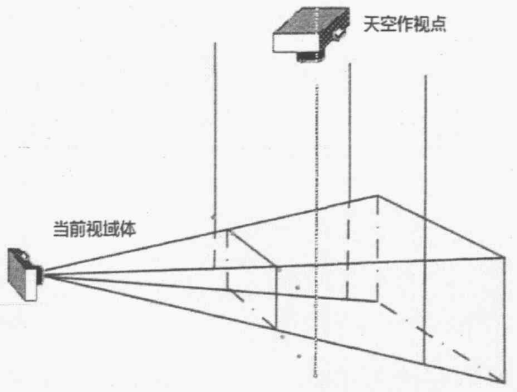


图 2.2 基于视点的自适应降雪遮挡图采样

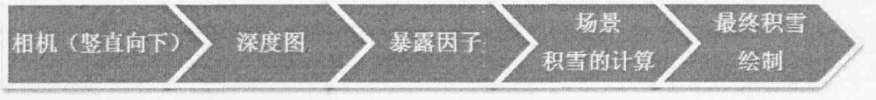


图 2.3 基于视点的自适应降雪遮挡图绘制流程

Step1 的主要目的是通过相机的位置和参数获得自适应的场景降雪遮挡图 (遮挡图的遮挡处理关系见下图 2.4)。遮挡图的获取采用了文献[36] 阴影图（shadow mapping）的思想，取天空为光源。由于只对当前视域内的景物生成降雪遮挡图，所以遮挡图的采样精度比较高。新加入的地物，可以实时的更新遮挡关系。对于离观察者很远的地表网格本文不进行降雪的遮挡判断，而是直接使用积雪纹理混合的方法，简化计算。所有的光亮度计算都在像素级上完成。由于目前的 GPU 都能从的顶点着色器程序中读取纹理图，所以可以在顶点着色器中直接根据遮挡图和顶点的法向进行偏移量的计算，根据网格的细分程度，可以对深度图进行不同次数的采样，从而可精确地判断积雪的遮挡关系。

2.2.3 积雪堆积的实现

在获取了实时更新的场景自适应降雪遮挡图后，即可生成积雪堆积的效果。在积雪的绘制中，积雪处应呈现出雪颜色和原有物体表面颜色的混合；对于被遮挡的区域，由于未积雪，会保持地物原有的颜色。

在 GPU 中进行积雪的判断和积雪量的计算。本文采用文献[3] 的积雪预测方程，见公式 (2.1)，来计算最终的效果。在方程中，积雪的效果由顶点的暴露程度 $f_e(\boldsymbol{p})$ 和顶点本身的倾斜程度 $f_{inc}(\boldsymbol{p})$ 来决定。根据预先生成的积雪遮挡图来确定竖直方向地物的遮挡关系，计算顶点暴露程度；顶点处表面的倾斜程度由顶点法向量决定。根据倾斜程度来计算积雪

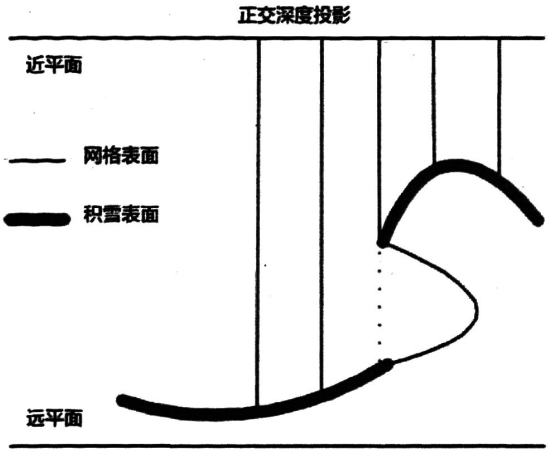


图 2.4 正交遮挡判断图

后地表在竖直方向上的拉伸，从而表现出积雪厚度变化，而暴露程度值作为最终透明信息存储。在最终绘制时，通过深度比较，确定地面景物的积雪表面，根据其透明度信息绘制积雪；而被遮挡的片元，由于未通过深度测试，最终无积雪，从而成为积雪阴影。

$$f_p(\boldsymbol{p}) = f_e(\boldsymbol{p}) \cdot f_{inc}(\boldsymbol{p}) \tag{2.1}$$

其中 $\boldsymbol{p}$ 是要计算的点，f_p 是预测函数，f_e 是暴露因子，f_{inc} 是倾斜因子。其中暴露因子是指场景中某一点积雪的概率，其值在 0-1 之间，0 为全遮挡不积雪点；1 为全积雪点；其定量计算公式为：

$$f_e(\boldsymbol{p}) = \frac{\sum_{\boldsymbol{M}_{samples}} D_{texdepth} - D_{actdepth} \geq 0}{\boldsymbol{M}_{samples}} \tag{2.2}$$

式中 $D_{texdepth}$ 为遮挡图中该点的深度，$D_{actdepth}$ 为该点实际的深度，$\boldsymbol{M}_{samples}$ 为总的采样数目。倾斜因子是该点的法向与竖直向上法向的夹角。其值由公式 (2.3) 决定,，式中 $\boldsymbol{N}$ 为法向矢量。为了增加雪场景的真实感，本文对法向量 $\boldsymbol{N}$ 和倾斜因子 $f_{inc}(\boldsymbol{p})$ 的值采用了倍频柏林噪声[37] 进行扰动，其随机性更强，进一步增强了场景细节的真实感。

$$f_{inc}(\boldsymbol{p}) = \boldsymbol{N} \cdot (0,1,0) \tag{2.3}$$

另外，为解决边缘的走样问题，本文在像素程序中加入纹理的四邻域多重采样步骤，经多重采样处理后，场景的细节真实感有所增强，边缘较为平滑。

2.3 基于视点的分层粒子飘雪模型

2.3.1 视点相关的粒子系统

在传统的粒子系统中，往往将粒子系统的发射源放置在固定点或者相对于视点固定，这种方法比较直观和实用，但是在大规模三维场景的漫游中，由于视点不断变化，在漫游过程中，该方法不能保证飘雪的一致性和连续性。为此，本文提出了与视点相关的粒子系统。

在具体的实施过程中，对粒子系统进行视点变换，使粒子投影到视域棱锥近平面上的位置相对不变，人眼感受到的降雪画面能保持连续性和一致性。我们利用非参数控制的方法[27]，把大部分的计算从 CPU 移到了 GPU 中，从而大大地提高了图形硬件并行计算功能的利用率。

2.3.2 基于视点的分层粒子飘雪模型

在雪场景漫游时，在观察者眼中，近处的雪花比较大，远处的雪花比较小。传统方法 (Direct X 或 OpenGL 中常用，见公式 (2.4)) 简单地引入距离项作为缩放雪粒子尺寸的因子，式中 $F_{finalSize}$ 为最终显示的雪花的大小；S_{size} 为雪花的原始大小；A、B 和 C 为控制系数,D 为离视点的距离。按该模型绘制的雪花虽然也有远近之分，但是总体上来说，其层次感不够细致。为此，本文提出了一种基于视点的分层粒子模型，见图 2.5，该模型结合视域体尺度，将粒子划分为多个层次，每个层数都有自己的粒子的基准尺寸，在结合公式 (2.4) 计算得到层内的不同距离雪花粒子的尺寸。用这种方法得到的最终结果比原有方法更具有层次性和真实感；同时，为了避免雪粒子距视点过近而产生走样，可将处于视域体内和离视点最近一层窗口以外的雪粒子隐藏掉。多次实验发现，将粒子分为 3-5 组可以获得比较好的视觉效果。

图 3.10 为整个粒子系统的框架图，白色部分为传统的粒子系统方法，深色部分为本文加入的主要改进，其中 CPU 的主要工作创建和初始化粒子；GPU 更新粒子的位置，更新粒子的速度，判断是否与地面进行了碰撞及消亡。

$$F_{finalSize} = \frac{h_{viewportHeight} \times S_{size}}{\sqrt{1/A + B \times D + C \times D^2}} \tag{2.4}$$

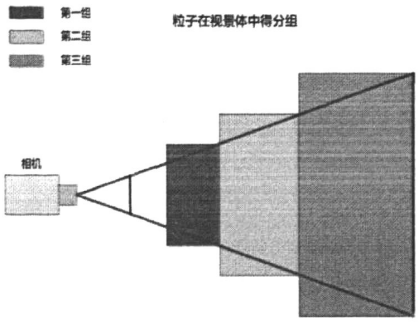


图 2.5 粒子分层策略 (以 3 层为例)

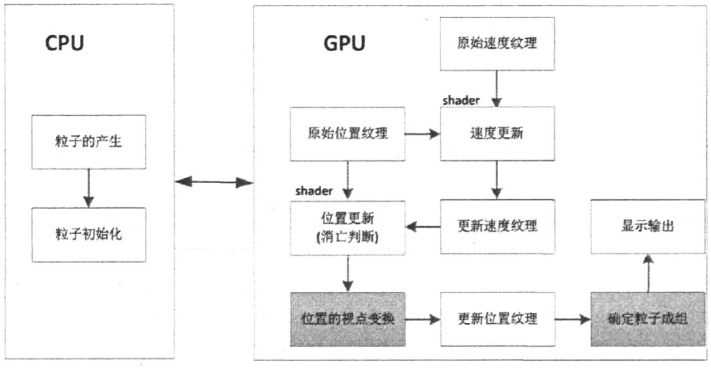


图 2.6 本文粒子系统实现框架

2.3.3 雪粒子系统的绘制

在本系统中，雪花的形状主要采用纹理映射的方法产生。在模拟空中飘落的雪花时，主要考虑的是粒子的位置变化及其自旋效应。其位置的确定可用如下通常的动力学方程来描述:

$$\boldsymbol{V}_n = \boldsymbol{V}_{n-1} + K \cdot \boldsymbol{W} + \boldsymbol{\alpha} \cdot n + \boldsymbol{r}(n) \tag{2.5}$$

$$\boldsymbol{S}_n = \boldsymbol{S}_{n-1} + \boldsymbol{V}_n \cdot \triangle t \tag{2.6}$$

其中 $\boldsymbol{V}_n$ 和 $\boldsymbol{V}_{n-1}$ 分别分别为第 n 时刻及前一时刻粒子的速度；K 为常数；$\boldsymbol{W}$ 为风速；$\boldsymbol{\alpha}$ 为粒子的加速度；$\boldsymbol{r}(n)$ 为第 n 时刻粒了速度的随机扰动量；$\boldsymbol{S}_n$ 和 $\boldsymbol{S}_{n-1}$ 分别为第 n 时刻及前一时刻粒子的位置。考虑到时间的离散特性，本文用差分方程在表示雪粒子的动力学特性。公式 (2.5) 中 $K \cdot \boldsymbol{W}$ 表示风力，其中 K 为空气与物体表面的粘滞系数，$\boldsymbol{\alpha} \cdot n$ 为加

速度项，第四项表示粒子碰撞产生的随机扰动项。粒子旋转的机制比较复杂，本文采用 18 张纹理随时间替换的方法[38] 来表示雪花的旋转，如图 2.7 所示。

图 2.7　雪花的旋转纹理图

在降雪计算过程中，将粒子的生存时间与初始高程值 h 绑定，在粒子的下降过程中，生存时间随同粒子的高程值 h 一起减小，在绘制前，将粒子的 h 值与地表的高程值 $height$ 做比较。其中有一个小技巧，当 $h < height$ 时，粒子的生命周期结束，但是并不将它删除，而是重新从出生点初始化该粒子，粒子系统的循环利用，可以提高粒子系统的效率。

2.4　大规模雪场景的实时绘制

本节首先介绍场景光亮度计算; 再描述基于纹理混合的场景积雪绘制; 然后再阐述利用进行 GPU 加速计算及实时绘制的实现。

2.4.1　光亮度计算

在光亮度计算中，本文所采用的是经典的 Blinn-Phong 光照模型, 物体表面的光强度 (物体的颜色) 取:

$$C_n = I_a K_a + I_p K_d (\boldsymbol{L} \times \boldsymbol{N}) + K_s I_l (\boldsymbol{N} \times \boldsymbol{H})^n \tag{2.7}$$

式中右边三项分别为环境光项、漫反射项和高光项。则最后雪场景中物体颜色的计算公式为:

$$C_n = f_p \times C_s + (1 - f_p) \times C_n \tag{2.8}$$

式中 C_s 和 C_n 分别为雪的颜色和物体本来的颜色，f_p 为混合系数, 其值可由公式 (2.1) 定量计算得到。

最后，像素级的法向计算需要用柏林噪声进行扰动以增强其随机性。本文使用了 $64 \times 64 \times 64$ 的 3D 噪声纹理。在噪声纹理中，本文使用 Perlin[37] 的方法，使其频率加倍，振幅减半，用此方法进行采样，其结果更具真实感。

2.4.2 基于几何与纹理混合的场景积雪绘制

对大规模动态雪场景绘制，由于其基于自定义的 GPU 顶点和片段计算量大，操作处理周期长，对系统会造成沉重的负担。为了减轻 GPU 处理顶点的计算量，本文结合了基于纹理混合的绘制方法，即在自适应降雪遮挡图精细采样区域之外不使用 GPU 的顶点和片段的几何处理，只使用 alpha 融合来实现积雪纹理和地表纹理的叠加。这样就大大加速了积雪场景的计算。对远处地物表面进行积雪处理时，为了避免不应被积雪的地物表面混合积雪纹理，本文定义了一些地物的简单积雪规则。以房屋模型为例，三维建筑包含许多的面片信息，如屋顶、门、窗等。积雪一般存在于法向量与竖直方向夹角较小的面片上，例如屋顶面元，如图 2.8 所示。因此在绘制远处景物表面积雪时，先计算各面片的法向量，检测出法向量与竖直方向的夹角小于 90 度的面元，然后根据夹角的大小和一定的扰动生成不同程度的积雪变化纹理，将它们与原来面片的纹理做一个混合，最终生成积雪效果。

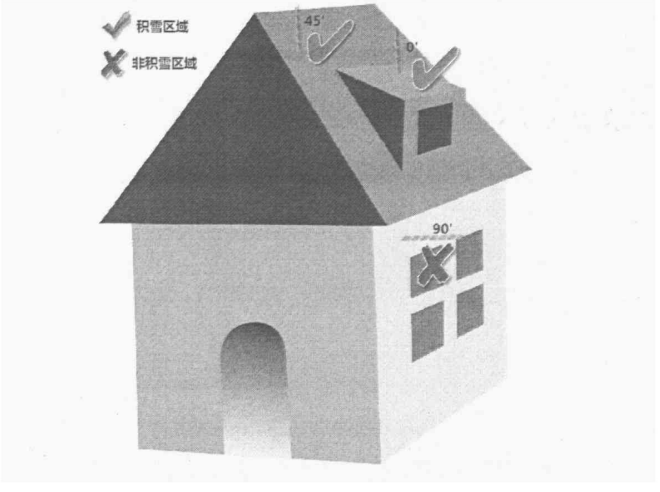


图 2.8 远处建筑纹理混合规则示意图

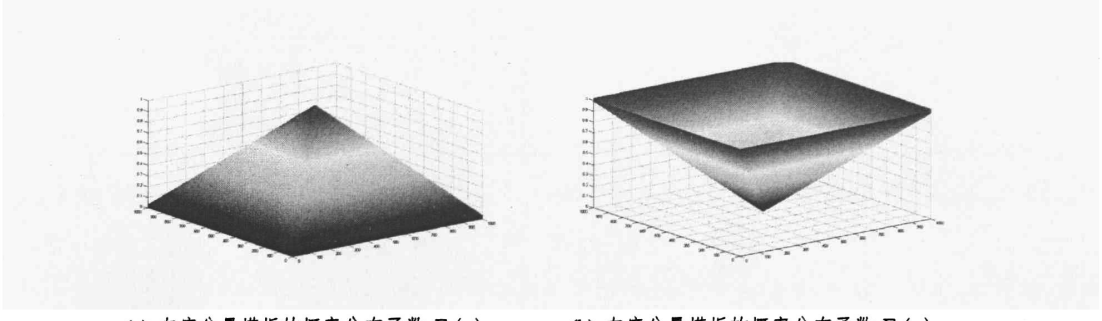
由于积雪具有连续性，为了使得积雪颜色有一个渐变的效果，可采用纹理颜色变化的密度函数形式为：

$$F_1(x) = \min(x, borderX - y, y, borderX - y) \tag{2.9}$$

但上面这个密度函数的趋势是中间区域增幅很陡峭，四周区域坡度扁平，见图 2.9(a)，而纹理边缘灰度值低表现出来的效果就是混合后地物表面边缘积雪缓慢，不合常理。为了避免这种现象，除了构造公式 (2.9) 这种中心向四周扩散的纹理外，我们还反转此种纹理生成另一种四周向中心扩散的密度函数 (见公式 (2.10))，其密度函数的趋势与上一个密度

函数刚好相反，见图 2.9(b)。通过上述两种的混合，平滑了积雪的纹理，保证了需要积雪的地物都可以得到几乎平等的积雪机会。

$$F_1(x) = \max(x, borderX - x, y, borderY - y) - int(\max(\frac{borderX}{2}, \frac{borderY}{2}) + 0.5) \quad (2.10)$$



(a) 灰度分量模板的概率分布函数 $F_1(x)$ (b) 灰度分量模板的概率分布函数 $F_2(x)$

图 2.9 灰度分量模板的概率分布函数

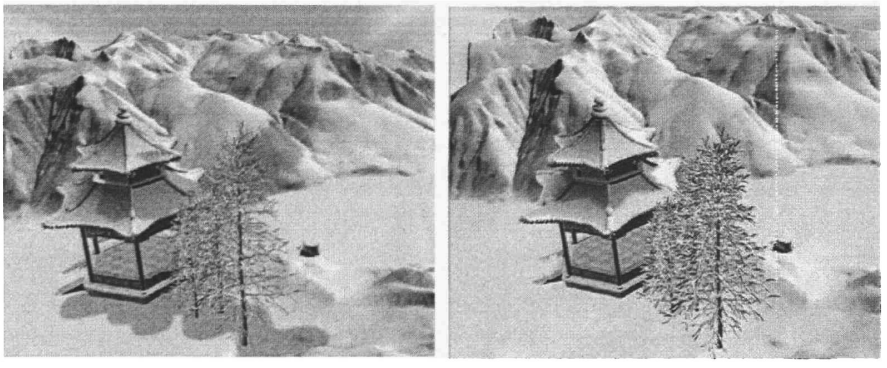
2.4.3 GPU 加速计算及实时绘制实现

由于目前 GPU 的并行处理能力非常强大，本文将场景中物体的遮挡关系判断及其表面光亮度的计算全部移到 GPU 中进行处理。CPU 主要的任务是进行场景数据的初始化及逻辑判断。这样就优化了系统的计算方案，从而使实时处理百万级面片复杂场景成为可能。在此基础上，本文实现了大规模动态雪场景（包括降雪及积雪）的实时绘制。

2.5 绘制结果

根据上述建模和绘制方法，本文在 Intel(R) Core(TM) i7-2600 CPU @ 3.40GHZ，8.0GB 内存，NVIDIA Geforce GTX 580 显卡的高端 PC 机上实现了大规模雪场景的实时生成。本文的雪粒子系统共有 5 万多个雪粒子。构建了野外场景和城市场景两个场景。野外场景的规模为 $4000 \times 4000m^2$，包含多棵树木及造型复杂精美的中国传统式亭子，场景的复杂度超过 100 万个面片，整个动态雪场景的平均绘制速达 42fps 以上；对于城市场景，其场景的复杂度超过 700 万个面片，整个动态雪场景的平均绘制速达 35fps 以上，以上两例皆能满足实时性的需要。欲了解更多详情，请观看本文所附的视频文件，下面图 2.10 为某一漫游视点下全局降雪遮挡图和本文的自适应降雪遮挡图对大场景积雪计算绘制效果的比较。

从中可以看出，对于大规模动态雪场景，全局遮挡图对积雪估计的区域及厚度的计算精度较低，遮挡误差较大，而本文方法则较好地克服了这一缺点。图 2.11 - 图 2.15 为部分实时绘制的结果。

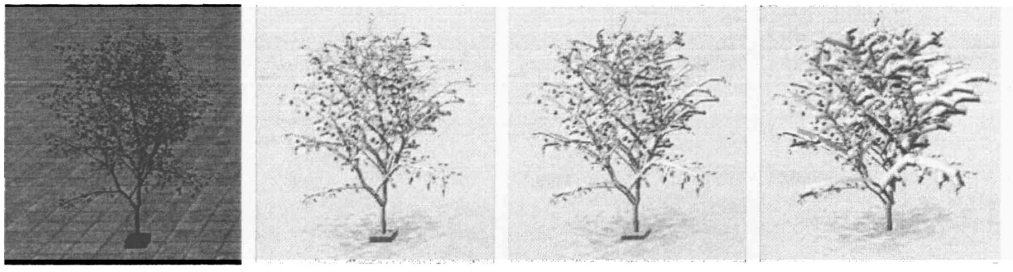


(a) 全场景降雪遮挡图[3]　　(b) 本文自适应降雪遮挡图效果

图 2.10　Ohlsson 的方法[3] 与本文的方法的效果对比

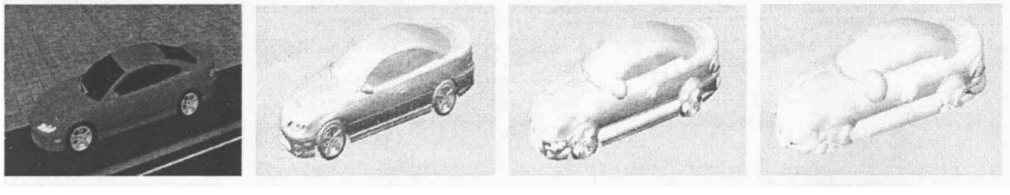
2.6　本章小结

本章提出了一种交互式大规模雪场景建模与实时绘制的新方法。该方法能高效地进行雪场景的建模，可定量地实时生成不同积雪量及飘雪效果的大规模雪场景。本方法首先提出了一种基于视点的自适应降雪遮挡图模型，该模型能解决以往全局场景降雪遮挡图方法所引起的大尺度场景个别区域积雪失真的缺点，可有效地提高大规模雪场景积雪的真实感，同时可对动态物体进行实时的积雪；所采用的基于视点的雪粒子分层技术可增强飘雪场景的层次感和视觉体验，并有效地减少了飘雪粒子的数目，提高了绘制效率；通过 GPU 图形硬件加速方案的优化设计，实现了大规模雪场景的真实感实时生成。本文方法在虚拟仿真及游戏设计等领域有着广泛的应用前景。



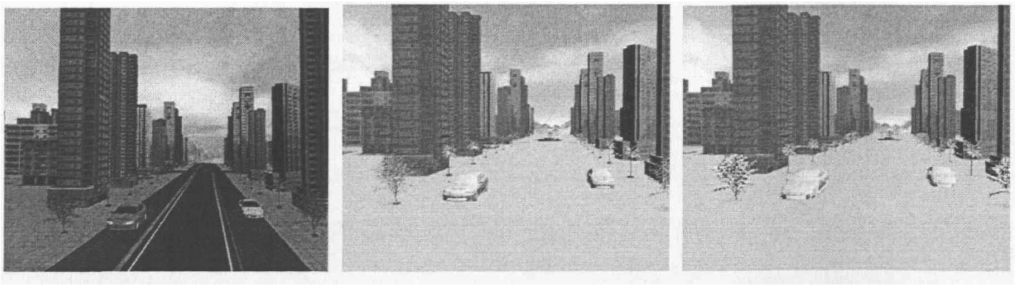
(a) 无积雪　(b) 小积雪　(c) 中积雪　(d) 厚积雪

图 2.11　不同积雪量的树挂效果



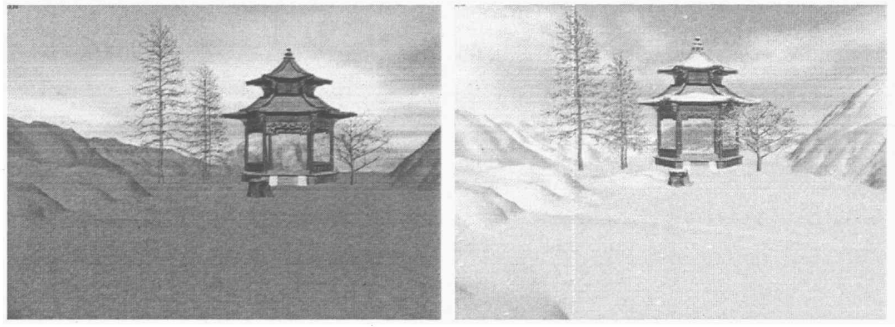
(a) 无积雪　(b) 小积雪　(c) 中积雪　(d) 厚积雪

图 2.12　轿车不同的积雪效果



(a) 无积雪　(b) 中积雪　(c) 厚积雪

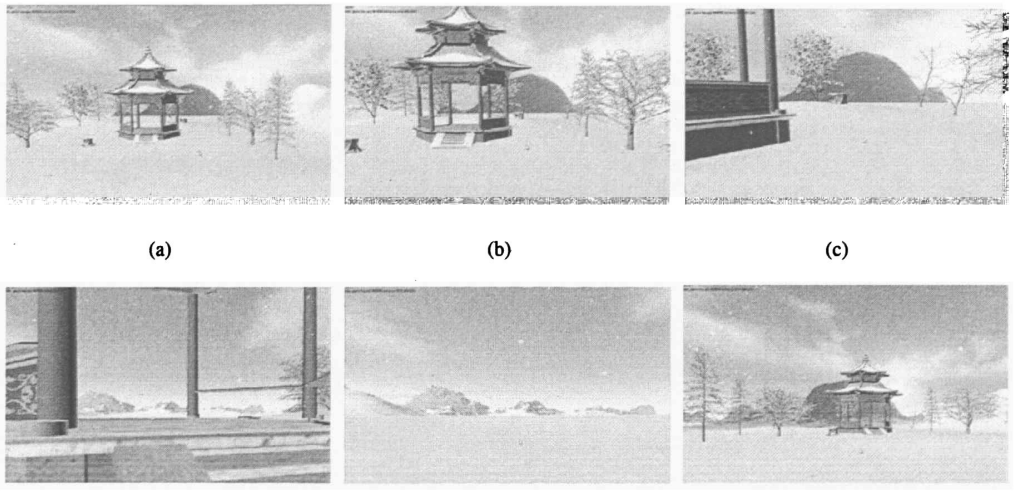
图 2.13　城市场景不同积雪量的景象对比



(a) 无积雪　　(b) 小积雪

(c) 中积雪　　(d) 厚积雪

图 2.14　野外场景不同积雪量的景象对比



(a)　　(b)　　(c)

(d)　　(e)　　(f)

图 2.15　野外雪场景的实时漫游图像序列

第3章 雪崩现象的实时模拟

3.1 引言

雪崩，俗称白色雪龙，是在长年积雪的山中常有的自然灾害，每年都有很多人死于雪崩，造成大量的经济损失。雪崩产生原因通常是覆雪处于一种“危险”的平衡状态下，如果稍微有外力作用，就会失去平衡，造成雪块滑动，进而引起更多的覆雪运动，使大量的积雪瞬间倾盆而下，如图 3.1，住在附近的人及村庄往往不能幸免，并对从事冰雪运动特别是高山滑雪者产生很大的危险。

图 3.1 壮丽的雪崩景象

在气象学、力学、灾难预防、数值计算领域对于雪崩的研究已经有很多年的历史。但是在计算机图形学领域，考虑到雪崩成因的复杂性、形态的多样性，对雪崩的运动过程模拟，现在还几乎是空白。

虽然雪崩现象形态多样，但其雪粒子的流动性是一个相同的基本特征。对于流体模拟的研究，近年来一直是计算机图形学领域的一个热点，自上个世界 80 年代 Reeves 首次将粒子系统引入了计算机图形学领域[24] 以来，基于粒子的拉格朗日方法中的 SPH (Smoothed Particle Hydrodynamics，即：平滑粒子流体动力学) 方法已成为了流体模拟的一

种主流方法。由于它可以模拟更多的流体细节（如泡沫、水花等）和处理更复杂的流体表面并能与多个基于物理的绘制引擎相兼容，在许多应用领域中有优于基于网格的欧拉方法之处。

本章将基于 Muller 等人[1] 经典的 SPH 的方法的基础上，实现了两种 SPH 模型，用简单的 SPH 模型来模拟牛顿流体；用复杂的 SPH 模型来模拟非牛顿流体。最终结合本文实现的复杂 SPH 模型、加入流变方程，成功实现了雪崩场景的实时模拟。

3.2 雪崩的分类和建模方法

雪崩非常壮丽的自然景观，但是由于其成因的复杂性、出现的突然性、形态的多样性，要用统一的模型来建模是非常困难的。

从宏观上来看，我们可以从三个方面来研究雪崩：

1. **雪崩成因**：哪些事件可以对雪崩的发生有贡献。
2. **雪崩释放**：如何破坏雪的平衡，让雪崩现象发生。
3. **雪崩运动**：雪崩发生后的运动过程。

雪崩的成因受到很多因素的影响。从微观上来讲；一堆堆的雪是由雪花组成的，而雪花是一种带有杂质的晶体，晶体的运动受到湿度和温度的影响；从宏观上来讲：随着长时间雪的堆积，各个雪层的物理特性都不一样，并且物理特性会随着时间的变化而变化。雪崩的释放从宏观上可以看成由于外力的作用下雪堆的平衡状态被打破，在重力的带动下，雪堆开始向山坡下倾滑，上层雪带动下层雪，最终形成雪崩。但是风、雨、温度、人工因素也会导致雪崩的释放。雪崩的运动就是指雪堆滑下山坡的过程，雪的重力、密度、山坡的摩擦力等因素的各异，导致最终雪崩的形态各异。这也是无法用一个统一的模型来建模雪崩的主要原因。本节主要研究和分析雪崩的运动特性，忽略其余两项。

3.2.1 雪崩的分类

由于雪崩现象的形态多样性，首先根据雪堆的宏观结构来进行分类：

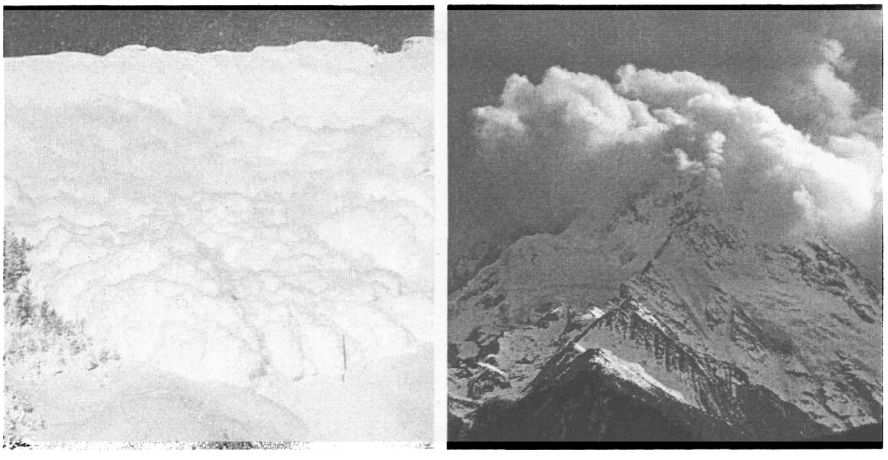
松雪坍塌 通常发生在陡峭的山坡上，松散的积雪下面可能会有一定的空间，这样的积雪在受力情况下会发生坍塌。但是雪崩的密度相对比较小，所以危害也不大。

雪板雪崩 多发生于30至45度的开放坡面上。冻结的雪形成一定厚度的冰层，冰层下方可能会是一个空洞的空间。在受到外力(特别是风力)作用的情况下，这样的冰层很可能会坍塌。大块大块的雪板破碎滑动，非常危险。

当然我们也可以根据雪的干湿程度对雪崩进行分类[1]：

湿雪雪崩 一般发生在30度以上的雪坡。有一定湿度和重量的积雪因为没有完全冻结，在外部因素影响下向下坡方向下冲，表现为流体的特性，如图3.2(a)。相对干湿雪崩来说，速度比较慢，通常只有$5-25m/s$,但是密度比较高，可达$150-500kg/m^3$[39],这是完全贴着地面流动的雪崩。

干雪雪崩 其触发是因为干的雪堆受到太大的压力，大多由人为的原因造成(如滑雪)，其传播速度可达$50-100m/s$、密度可达$5-50kg/m^3$高度可达$10-100m$[39],大量的雪粒子悬浮在空中，并没有贴着山体滑动，形成了所谓的"雪云"景象，如图3.2(b)。



(a) 湿雪雪崩 (b) 干雪雪崩

图3.2 湿雪崩(左)和干雪崩(右)真实截图

3.2.2 雪崩的建模

对于雪崩的建模方法，从不同的研究角度可以分为[39]：

[1]http://baike.baidu.com/view/39897.htm

统计学方法 基于统计的方法在安全选址上有重要的应用，这些方法需要过去雪崩的精确数据来计算边界，其主要关心的是雪崩停止的位置。

动量计算方法 使用 depth-averaged 的动量方程来计算雪崩的流动特性，它们大都使用浅水 (Saint-Ventant) 方程来模拟这种流动性。

计算流体动力学（CFD）方法 由于雪崩现象具有天生的流动性。采用计算流体动力学（Computational Fluid Dynamics，简称：CFD）的方法来建模肯定是一个重要的手段，但 CFD 方法通常适用于粘滞系数为常数的牛顿流体，从本质上来讲雪崩属于非牛顿流体，因此需要考虑雪崩的流变特性，但是这方面的研究目前并不是很多。

本章将采用计算流体动力学（CFD）的方法对雪崩景象进行模拟。原因是 CFD 方法是基于物理的，并已发展研究了多年，计算模型相对成熟。而对于属于非牛顿流体的雪崩，只要将其流变特性（即粘滞系数在过程变化中为非常数），考虑进来加以修正，则较易获得理想的模拟效果。

3.3 流体模拟基础及 SPH 模型

基于粒子的 SPH(Smoothed Particle Hydrodynamics，平滑粒子流体动力学) 的方法是目前流体模拟的热门方法之一。相对于基于网格的欧拉方法，它对流体的细节处理得更好，并且易于进行并行加速处理，从而达到复杂场景实时处理的目的。

本节先介绍流体模拟的理论基础，并着重介绍 SPH 的计算模型和方法。

3.3.1 流体模拟基础

基于物理的流体模拟大多是以 Navier-Stokes 作为基本的流体运动物理方程实现力的求解与模拟。Navier-Stokes 方程，是描述粘性不可压缩流体的动量守恒和质量守恒方程，简称 N-S 方程。

N-S 方程可以看成是牛顿第二定律在流体模拟上的一个应用。它的基本形式如下：

$$\nabla \cdot u = 0 \tag{3.1}$$

$$\frac{\partial u}{\partial t} = -(u \cdot \nabla)u + \mu \nabla^2 u - \frac{\nabla p}{\rho} + f \tag{3.2}$$

公式 (3.1) 是质量守恒方程，公式 3.2 是动量守恒方程。式中 u 为速度，p 为压强，μ 为粘滞系数，ρ 为密度，f 为外力。

公式 (3.1) 定义了流体的保体积特性，速度场的散度为 0，也就是通常我们所说的流体的不可压缩性。通常我们会把流体的密度视作常量，所以体积本质上等效于质量守恒。

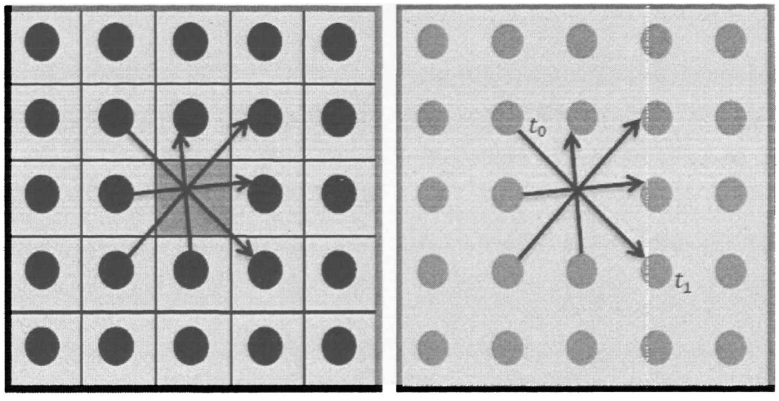
公式 (3.2) 本质上是从牛顿第二定律 $F = ma$ 推导出来的。等式左侧是流体的加速度，等式右侧为流体受力的影响，从左至右依次为对流，粘滞力，压力和外力。平流和扩散是对流的主要方式，其中平流表示流体内部的整体运动，而扩散是指分子的随机布朗运动。外力项 f 主要是指重力，当然还有一些其他力的作用 (如空气浮力，流体与其他物体的交互产生的力)。

欧拉方法 欧拉方法把空间划分成网格，分析每个网格节点上的流体的密度、速度、压强等参数随时间的变化，如图 3.3(a)。通过将连续 N-S 方程离散成网格以后，然后通过微小时间步长跟踪网格结点参数的变化，得到整个流场的时变信息。虽然基于网格的方法能够很好的描述流体的许多属性，如密度、压强等，但网格的分辨率会对流体细节造成很大的影响，如果单单依靠提高网格的分辨率，又会导致求解过程太慢。为了提升流体的模拟细节，Enright[40] 等人在网格法的基础上引入粒子水平集方法，流体表面细节通过粒子来追踪。目前，基于粒子的拉格朗日方法变成了研究的主流。

拉格朗日方法 拉格朗日方法[41] 是从流体微团（粒子）出发，如图 3.3(b)，来分析微团的密度、压强、速度、粘滞力、漩涡力等参数的时变信息，从而研究整个流体的运动。N-S 方程同样是拉格朗日方法及基础，公式 (3.2) 描述不可压缩流体 N-S 方程的基本拉格朗日形式：

$$\frac{\partial u}{\partial t} = \mu \nabla^2 u - \frac{\nabla p}{\rho} + f \tag{3.3}$$

公式 (3.3) 的参数意义与公式 (3.2) 相同。需要注意的是它和欧拉形式的 N-S 方程有两点区别。第一，质量守恒是其天然属性，因为粒子的总数是守恒的，从单个粒子来讲其质量也是守恒的，所以不受质量守恒公式 (3.1) 的约束。第二，不需要对流项，对流项是对流体运动的宏观描述，对单个粒子作分析时，不需要对流项。拉格朗日方法优点在于容易表达和实现，其完全不需要定义有限空间网格，就可以让流体粒子流动到场景的任意角落。因此拉格朗日流比较适合大规模流体场景的模拟，它另一个优势在于容易表述流体的丰富细节 (如泡沫、浪花等)。但是对于精细、平滑的流体表面，不是拉格朗日法的强项，切随着粒子数的增加，其计算量也会不断增大。



(a) 基于网格的欧拉方法　　(b) 基于粒子的拉格朗日方法

图 3.3　欧拉法和拉格朗日法对比图

3.3.2　SPH 方程[1]

SPH 的基本思想与拉格朗日流体模拟方法相同，通过离散的采样点来计算连续的流场，这些离散的采样点被称为光滑粒子。每个光滑粒子在模拟中具有一些属性，四个基本属性分别为：粒子的质量、半径、位置和速度。除此以外，粒子还在模拟过程中计算其他属性，如密度、压力、加速度等，当然还可以根据需要计算粘滞力、漩涡力等，这些属性信息本来应该是粒子所在空间位置的属性，在模拟的过程中将其引申为粒子的自身属性。

SPH 从其本质上来讲，是一种差值的方法，它将粒子分散于 3D 空间，每个粒子被认为是占据着空间的一个片段，然后任何位置的物理量如密度，压力等，都可以通过周围离散点集的对应物理量插值得到，也因此粒子必须较为密集，以使插值结果更能体现该位置的物理属性。SPH 使用以下的积分插值公式：

$$A_i = \int_{\Omega} A(\boldsymbol{r}')W(\boldsymbol{r}-\boldsymbol{r}', h)d\boldsymbol{r}' \tag{3.4}$$

其离散化后的形式可以表示为：

$$A_i = \sum_j A_j \frac{m_j}{\rho_j} W(\boldsymbol{r}_{ij}, h) \tag{3.5}$$

公式 (3.4) 中 A_i 代表 $\boldsymbol{r}$ 位置的某一属性，$W(\boldsymbol{r}-\boldsymbol{r}', h)$ 是核半径为 h 的光滑滑核函数，见图 3.4，公式 (3.5) 中 A_i 代表粒子 r 处的属性，j 是在其核半径范围内的邻域粒子数目。

在基于粒子的拉格朗日方法中，模拟开始的时候，一般都会给每个粒子赋予初始位置

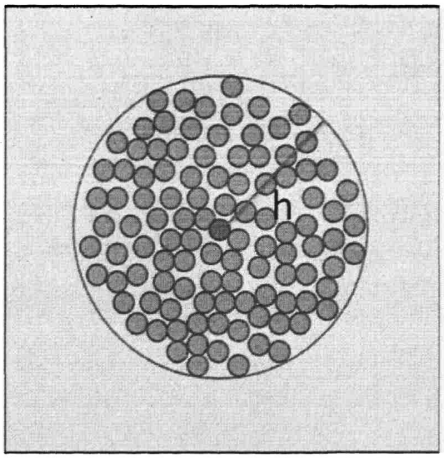


图 3.4 某一 (红色) 粒子的核半径范围

和初始速度，然后通过在每个时间步长计算所有粒子位置的密度、压力、并最终更新粒子位置和速度。

以公式 (3.5) 为基础，SPH 模型的物理属性计算如下。

3.3.3 密度

在拉格朗日方法中，需要计算作用于粒子的力如重力，粘滞力等，而其中质量是粒子所固有的不变属性，因此首先必须在每个时间步长计算粒子密度。粒子 i 的密度计算公式如下：

$$\rho_i = \sum_j m_j W(\boldsymbol{r}_{ij}, h) \tag{3.6}$$

式中 ρ_i 是粒子 i 处的密度，m 是粒子的质量，h 是核半径，$\boldsymbol{r}_i, \boldsymbol{r}_j$ 分别是粒子 i 和粒子 j 的空间位置。

3.3.4 内部力

内部力指的是流体粒子之间产生的力，它包括压力和粘滞力等，在公式 (3.3) 中分别对应压力项和粘滞力项。

压力的计算公式：

$$f_i^{pressure} = -\sum_{i \neq j} \frac{p_i + p_j}{2} \frac{m_j}{\rho_j} \nabla W(\boldsymbol{r}_{ij}, h) \tag{3.7}$$

粘滞力计算公式：

$$f_i^{viscosity} = \sum_{i \neq j} m_j (\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2}) W(\boldsymbol{r}_{ij}, h) \tag{3.8}$$

式子中 p 是压强，其余参数含义与公式 (3.6) 相同。

3.3.5 外部力

外部力，即公式 (3.3) 中的外力项 f，是流体粒子受到的除流体粒子给予的力之外的所有力，包括重力，浮力等，该项等于其他所有力的加和：

$$f^{external} = \sum_n f \tag{3.9}$$

其重力计算公式:

$$f_i^{gravity} = \rho_i g \tag{3.10}$$

浮力公式:

$$f_i^{buoyancy} = b(\rho_i - \rho_0) g \tag{3.11}$$

式子中 g 为重力加速度，b 为浮力系数。

从插值公式 (3.5) 可以看出，平滑核函数以及核半径参与了 SPH 流体所有物理量的计算过程，它们在很大程度上影响了密度，压力等重要物理属性的插值结果，进而会很大影响了流体模拟的结果。

3.3.6 平滑核函数

在 SPH 方法中，插值所使用的核函数对流体的稳定性、模拟速度和物理特征的合理性，都有着重大的影响。因此对核函数的选取非常重要。因为核函数是满足径向对称的，

可以将核函数写成以下形式 $W(r,h)$，其中 r 是距离。Monaghan[42] 提出所有的核函数必须满足如下两个条件:

$$\int_r W(r,h)dr = 1 \tag{3.12}$$

$$\lim_{h\to\infty} W(r,h) = \theta(r) \tag{3.13}$$

式中 $\theta(r)$ 是狄克拉函数:

$$\theta(r) = \begin{cases} \infty & \|r\| = 0 \\ 0 & otherwise \end{cases} \tag{3.14}$$

本文工作的简单 SPH 流体模拟部分是基于 Müller 2003 年提出的核函数[1]，首先是 Poly6 核函数:

$$W_{poly6}(r,h) = \begin{cases} \frac{315}{64\pi h^9}(h^2 - r^2)^3 & 0 \le r \le h \\ 0 & otherwise \end{cases} \tag{3.15}$$

它的梯度:

$$\nabla W_{poly6}(r,h) = -r\frac{945}{32\pi h^9}(h^2 - r^2)^2 \tag{3.16}$$

拉普拉斯算子:

$$\nabla \cdot \nabla W_{poly6}(r,h) = \frac{945}{8\pi h^9}(h^2 - r^2)\left(r^2 - \frac{3}{4}(h^2 - r^2)\right) \tag{3.17}$$

Poly6 核函数的优势在于只包含了 r 的平方项，避免了开销较大的三次方运算。它可以被用在除了粘滞力和压力外的所有涉及到核函数的计算，但是该核函数在中心附近的梯度也是趋向于零的，也就是说当粒子距离过近的时候，粒子之间的排斥力会消失，而这显然是不符合物理的，所以不能作为压力的核函数。但一问题可以通过引入 Spiky 核函数解决。

$$W_{spiky}(r,h) = \begin{cases} \frac{15}{\pi h^6}(h - r)^3 & 0 \le r \le h \\ 0 & otherwise \end{cases} \tag{3.18}$$

$$\nabla W_{spiky}(r,h) = -\frac{45}{\pi h^6}(h-r)^2 \tag{3.19}$$

当计算粘滞力时,poly6 核函数的问题在于它的拉普拉斯算子在很容易变成负值，见图3.5(左)，从而会使流体模拟得到错误的结果，即粒子收到粘滞力的约束，但是却运动的飞快。而正确的结果应该是粘滞力使粒子的速度趋向于跟周围粒子相同。因此我们需要构造一个专用于粘滞力的核函数，保证其拉普拉斯算子在任何情况下是正值。函数如下：

$$W_{viscosity}(r,h) = \begin{cases} \frac{15}{2\pi h^3}(-\frac{r^3}{2h^3}+\frac{r^2}{h^2}+\frac{h}{2r}-r) & 0 \le r \le h \\ 0 & otherwise \end{cases} \tag{3.20}$$

拉普拉斯算子：

$$\nabla \cdot \nabla W_{viscosity}(r,h) = \frac{45}{\pi h^5}(1-\frac{r}{h}) \tag{3.21}$$

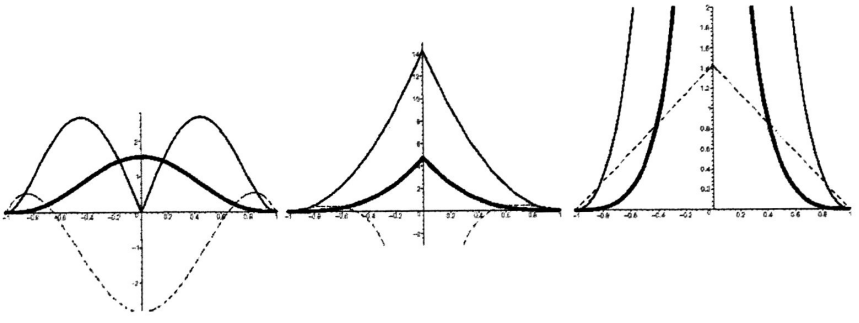


图 3.5 poly6（左）Spiky（中）viscosity（右）三种核函数的函数值（粗线）、梯度的绝对值（细线），和拉普拉斯算子（虚线）在 h=1 时的值[1]

本文使用上述的三种核函数作为简单 SPH 流体模拟时的核函数，实现了 SPH 框架以作为雪崩模拟的的基础。

3.4 牛顿流体和非牛顿流体

按照流体力学的观点，流体可分为理想流体和真实流体两大类。理想流体在流动时无阻力，故称为非粘性流体。实际流体流动时有阻力即内摩擦力 (或称剪切力)，故又称为粘

性流体。根据作用于流体上的剪切应力与产生的剪切速率之间的关系，粘性流体又可分为牛顿流体[2]和非牛顿流体[3]。

3.4.1 牛顿流体

大多数人可能会直觉地认为粘滞力和所谓流体的“厚度”有关，那么纯净水的厚度非常低而蜜糖的厚度非常大。对于日常生活，以上理解已经足够，因为人们所见到的大多数流体都是牛顿流体。

Sir Isaac Newton 提出了以下通用的牛顿模型[43]：

$$\tau = \mu(|\dot{\gamma}|)\dot{\gamma} \tag{3.22}$$

$$\mu_{eff} = \frac{\tau}{\dot{\gamma}} \tag{3.23}$$

其中 τ 是剪切应力，γ 是剪切速率，μ 是粘滞系数，而 $|\dot{\gamma}|$ 第二应变率不变量。那么剪切应力和剪切速率就是线性关系，如图 3.6，通常把 μ_{eff} 称为有效粘滞力。

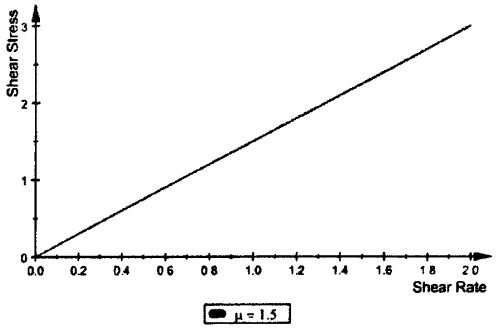


图 3.6 在 $\mu = 1.5$ 的情况下，剪切应力和剪切速率之间是线性关系

这里需要注意的是温度相对于压力来说，对粘滞系数的影响更大。对于水来说。温度从 0°C 到 100°C，粘滞系数的的变化范围是 1.79×10^{-3}Pa·s 到 0.28×10^{-3}Pa·s

由于牛顿流体切应力和剪切速率之间呈现线性关系，由由离散的 SPH 方程，公式 (3.5)，可知：对于不可压缩的的 N-S 方程，其粘滞力项为 $\frac{\mu}{\rho_i}\nabla^2 v$，从而推导出以下计算粘滞力的公式：

[2]http://en.wikipedia.org/wiki/Newtonian_fluid

[3]http://en.wikipedia.org/wiki/Non-Newtonian_fluid

$$f_i^{viscosity} = \frac{\mu}{\rho_i}\nabla^2 v = \frac{\mu}{\rho_i}\sum_{i\neq j}\frac{m_j}{\rho_j}(v_j - v_i)\nabla^2 W(\boldsymbol{r}_{ij}, h) \tag{3.24}$$

对于其求解的精确性，非常依赖于对光滑核函数的拉普拉斯求解。这也是 SPH 的重大缺点之一，因为粒子的稍稍扰动会对二次求导项造成重大的影响。虽然 Müller[1] 等人引入了特殊的核函数来保持粒子的稳定性，但是求解的误差还是比较大，其求解公式为：

$$f_i^{viscosity} = \frac{\mu}{\rho_i}\nabla^2 v = \sum m_j \frac{4(\mu_i + \mu_b)r_{ij}\cdot\nabla W(\boldsymbol{r}_{ij}, h)}{(\rho_i + \rho_j)^2(|\boldsymbol{r}_{ij}|^2 + \eta^2)} \tag{3.25}$$

3.4.2 非牛顿流体

非牛顿的流体的剪切应力和剪切速率并不是简单的线性关系。有效粘滞系数受到外界的因素包括时间、温度等的影响。由于其内部机理的复杂性，很难测量，一般可以根据造成非牛顿性的主要因素来进行分类：

时间粘性 (Time-dependent viscosity) 流体 固定剪切应力施加到流体上，流体的粘性随着时间的改变而改变。其中抗流变流体 (Rheoptic Fluids) 的粘性随着时间变化而加强，这种流体比较少见，奶油就是这种类型；而触变性流体 (Thixotropic fluid) 的粘性随着时间的变化而减小，泥浆就属于触变性流体。

剪切应力粘性 (Shear-stress dependent viscosity) 流体 流体粘性随着剪切应力的变化而变化。胀流性流体 (dilatant fluid) 会随着剪切应力的施加，其有效粘度也会随着增加，淀粉和水的混合就是这种流体；而假塑性流体的性质则正好相反，岩浆、油漆、血液都属于这个类别。

对于非牛顿流体的建模，已经存在很多种成熟的方法：

幂律分布模型 (Power-Law) 这是相对简单的一种可以用来近似真实流体的非牛顿流体模型，通常也被称作 Ostwald-de Waele 定律[43]，其公式：

$$\mu(|\dot{\gamma}|) = K\,|\dot{\gamma}|^{n-1} \tag{3.26}$$

其中 K 是流体的一致性系数，n 是流体的表现系数，见图 (3.7), 这个模型可以用来模拟假塑性流体 ($n < 1$) 和胀流型流体 ($n > 1$), 而当 $n = 1$ 时，就退化成了牛顿流体。

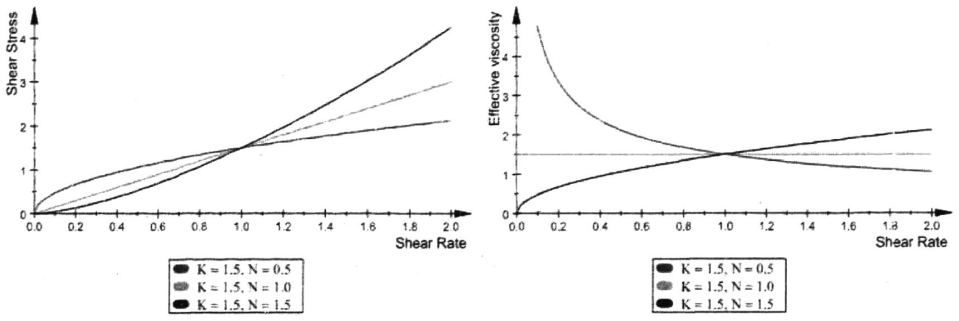


图 3.7　剪切应力曲线 (左) 和有效粘度曲线 (右)，在 $K = 1.5, n$ 取 $0.5, 1.0, 1.5$ 的情况

Bingham 模型 此模型在高剪切应力的情况下表现的像流体，在剪切应力很小的时候表现出固体的特性, 见图 3.8, 其公式为:

$$\tau = \tau_p + K\dot{\gamma} \quad for\,|\tau| \geq \tau_p \tag{3.27}$$

$$\tau_p = 0 \quad |\tau| \leq \tau_p \tag{3.28}$$

其中 K 是一个动态常数，τ_p 是屈服应力。

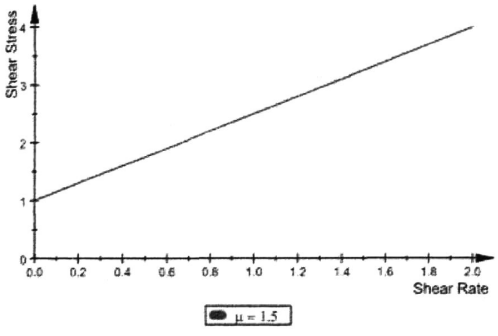


图 3.8　在 $K = 1.5$ 的情况下，Bingham 模型模型剪切应力和剪切速率的曲线

Cross 模型[44] 此模型和幂律分布模型很类似，但是它限制了粘性的下界 μ_0 和上界 μ_∞，它们可以保证流体在很小的剪切速率下，表现的像牛顿流体；但是在剪切速率比较大的时候，又会表现出幂律分布模型的特性，其公式如下:

$$\mu(|\dot{\gamma}|) = \mu_\infty + \frac{\mu_0 - \mu_\infty}{1 + K\,|\dot{\gamma}|^n} \quad 0 \leq \mu_0 \leq \mu_\infty \tag{3.29}$$

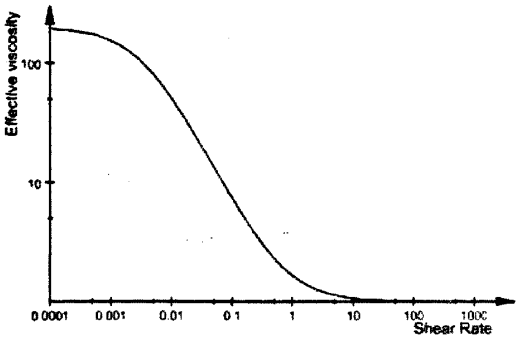


图 3.9 Cross 模型的有效粘性曲线，随着剪切速率的增大，呈现对数级衰减 ($\mu_0 = 200, \mu_\infty = 1, n = 1, K = 300$)

总上可以看出，剪切应力和剪切速率不是线性关系，非牛顿流体的粘性计算是非常复杂的，为了正确的计算流体的正应力和偏应力，需要更加高级的方法来求解。

本章将采用 Cross 模型，因为雪崩景象在发生初期速度较慢，其特性类似于牛顿流体；而到后来速度越来越大，其非牛顿流体的特性变得极为明显，而这与 Cross 模型所能表现出来的性能相当符合，故我们以 Cross 模型可有效地表示雪崩动态景象的流变特点。

Hosseini[45] 等人提出了一种正确计算 SPH 模型偏应力的方法，从公式 (3.23) 得到了计算通用牛顿流体的剪切应力的方法。$|\dot{\gamma}|$ 是第二应变率不变量，其计算公式为：

$$|\dot{\gamma}| = \sqrt{trace(\dot{\gamma})^2} \tag{3.30}$$

其中 $trace$ 算子是矩阵的迹，$trace(A) = \sum_{i=1}^{n} a_{ii}$，可以用速度的张量场来表示剪切应变速率 $\dot{\gamma}$:

$$\nabla \boldsymbol{v} = \begin{bmatrix} \frac{\partial v_x}{\partial x} & \frac{\partial v_x}{\partial y} & \frac{\partial v_x}{\partial z} \\ \frac{\partial v_y}{\partial x} & \frac{\partial v_y}{\partial y} & \frac{\partial v_y}{\partial z} \\ \frac{\partial v_z}{\partial x} & \frac{\partial v_z}{\partial y} & \frac{\partial v_z}{\partial z} \end{bmatrix} \tag{3.31}$$

速度的张量场包含对称的部分和非对称的部分，非对称的部分称为旋转速度张量，对于流体的粘度没有什么影响，对称的部分计算公式为：

$$\dot{\gamma} = \frac{1}{2}(\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^T) \tag{3.32}$$

以上计算得到的张量变化率称为柯西应变张量 (Cauchy's strain tensor)。

$$\dot{\gamma} = \frac{1}{2}(\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^T) = \begin{bmatrix} \frac{\partial v_x}{\partial x} & \frac{1}{2}(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x}) & \frac{1}{2}(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x}) \\ \frac{1}{2}(\frac{\partial v_y}{\partial x} + \frac{\partial v_x}{\partial y}) & \frac{\partial v_y}{\partial y} & \frac{1}{2}(\frac{\partial v_y}{\partial z} + \frac{\partial v_z}{\partial y}) \\ \frac{1}{2}(\frac{\partial v_z}{\partial x} + \frac{\partial v_x}{\partial z}) & \frac{1}{2}(\frac{\partial v_z}{\partial y} + \frac{\partial v_y}{\partial z}) & \frac{\partial v_z}{\partial z} \end{bmatrix} \tag{3.33}$$

对于速度张量场的偏导数可以直接使用比 SPH 的方法更精确的方法[46] 来计算：

$$\nabla \boldsymbol{v}_i = \sum_j \frac{m_i}{\rho_i}(\boldsymbol{v}_j - \boldsymbol{v}_i) \otimes W(\boldsymbol{r}_{ij}, h) \tag{3.34}$$

其中 $\otimes$ 是外积，$\nabla \boldsymbol{v}$ 速度的应力张量，根据以上推论，就可以得出非牛顿流体近似的张力计算方程：

$$f_i^{stress} = \frac{1}{\rho_i}\nabla \cdot \tau_i = \frac{1}{\rho_i}\sum_{j \neq i} \frac{m_i}{\rho_j}(\tau_i + \tau_j) \cdot \nabla W(r_{ij}, h) \tag{3.35}$$

3.5 雪崩动态景象的实时模拟

根据上一节对牛顿流体和非牛顿流体特点的具体分析，本节我们尝试用 SPH 的通用框架来实现牛顿流体和非牛顿流体的实时模拟，，并通过比较它们的差别和优劣，从而选出优化参数方案，实现对雪崩动态景象的实时模拟。下面分别加以描述。

通用的 SPH 的计算框架分为 3 步，见图 3.10：

Step1 更新数据结构 (最重要的是为了加速空间邻域查找所建立的网格、列表、哈希数组)。

Step2 计算 SPH 粒子的内部力，由于力的计算要依赖于密度，所以密度必须要先算。

Step3 计算 SPH 的外部力、一定时间步长的属性更新、以及最终的颜色计算。

3.5.1 牛顿流体 SPH 实现

对于通用的牛顿流体，本文使用上一章介绍的 SPH 核函数，这里有三个主要的方程：

密度计算：

$$\rho_i = \sum_{j \neq i} m_j W_{poly6} \tag{3.36}$$

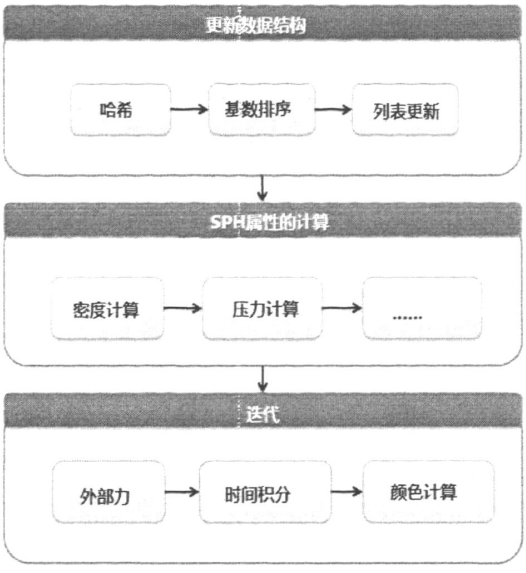


图 3.10 通用的 SPH 计算框架

压力计算：

$$f_i^{pressure} = -\frac{1}{\rho_i}\sum_{j\neq i} m_j \frac{P_i + P_j}{2\rho_j}\nabla W_{spiky} \tag{3.37}$$

关于压强 P, 本文使用理想气体公式来求解：

$$P = k(\rho - \rho_0) \tag{3.38}$$

最后对于粘滞力的计算公式为：

$$f_i^{viscosity} = \frac{\mu}{\rho_i}\sum_{j\neq i} m_j \frac{\boldsymbol{v}_j - \boldsymbol{v}_i}{\rho_j}\nabla_2 W_{viscosity} \tag{3.39}$$

本文的 SPH 系统中，由于粒子质量是恒定的，所以可以采用预计算进行加速：

$$\rho_i = m * W_{poly6}^{coeffs}\sum_{j\neq i} W_{poly6}^{variable} \tag{3.40}$$

$$f_i^{pressure} = -m * \nabla W_{spiky}^{coeffs}\frac{1}{2} * \frac{1}{\rho_i}\sum_{j\neq i}\frac{P_i + P_j}{\rho_j}\nabla W_{spiky}^{variable} \tag{3.41}$$

$$f_i^{viscosity} = \mu * m * \frac{1}{\mu_i} * W_{spiky}^{coeffs} \sum_{j \neq i} \frac{\boldsymbol{v}_j - \boldsymbol{v}_i}{\rho_j} W_{viscosity}^{variable} \tag{3.42}$$

其中本文用 $W^{variable}$ 来表示光滑核函数中的变量部分，W^{coeffs} 表示光滑核函数中的常量部分，可以预计算。当然，因为计算中有数据之间的依赖性，压力和粘滞力的计算都依赖于密度，所以第一步先要计算密度，然后压力和粘滞力的计算可以再同时进行，见图 3.11。

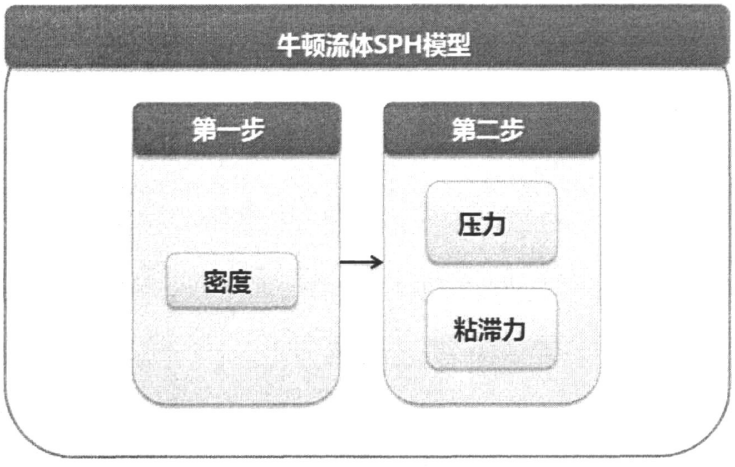


图 3.11 牛顿流体 SPH 方法的二步计算

3.5.2 非牛顿流体的 SPH 实现

非牛顿流体的 SPH 实现是以上一步的牛顿流体的实现为基础的。但是由于粘性系数 μ 不是常量，所以需要用额外方法来剪切应力和粘滞力。本节在 Müller[1] 等人实现的牛顿流体 SPH 模型的基础上，结合使用了 Hosseini[45] 等人的模型，此模型使用了更加精确的核函数，对应力的计算误差进行了修正，最重要的是此模型可以支持流变特性，用它来模拟非牛顿流体便再好不过了。

本文最终模拟的雪崩效果就是采用了非牛顿 SPH 的实现。下面展开描述具体的实现细节。

对于密度的计算，我们使用通用的 SPH 计算公式 (3.6)：

$$\rho_i = \sum_{j \neq i} m_j W_{poly6} \tag{3.43}$$

对于 SPH 的压力计算，相比于之前通用的方法，公式 (3.7), 本文使用更为精确的方法：

$$f_i^{pressure} = -\sum_{j \neq i} m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \nabla W_{spiky} \tag{3.44}$$

关于压强 P, 本文使用理想气体公式来求解：

$$P = k(\rho - \rho_0) \tag{3.45}$$

从 Müller[1] 等人的经验中得知，使用小一点的粘滞系数，不仅可以用来保持流体的稳定，还可以处理大一点的时间步长：

$$f_i^{viscosity} = \frac{\mu}{\rho_i} \sum_{j \neq i} m_j \frac{v_j - v_i}{\rho_j} \nabla^2 W_{viscosity} \tag{3.46}$$

文本还加入了 XSPH[47] 项来保持粒子运动的稳定性：

$$f_i^{xsph} = \sum_{j \neq i} 2 m_j \frac{v_j - v_i}{\rho_i + \rho_j} W_{poly6} \tag{3.47}$$

使用公式 (3.34) 来计算速度应变张量：

$$\nabla \boldsymbol{v}_i = \sum_j \frac{m_i}{\rho_i} (\boldsymbol{v}_j - \boldsymbol{v}_i) \otimes W_{cubic} \tag{3.48}$$

可以计算得到 SPH 的剪切应力：

$$f_i^{stress} = \frac{1}{\rho_i} \sum_{j \neq i} \frac{m_j}{\rho_i} (\tau_i + \tau_j) \cdot \nabla W_{cubic} \tag{3.49}$$

其中 τ 定义为剪切应力，本文系统支持好几种流变模型，具体见下文。上面两个公式我们使用了 cubic spline 核函数，见公式 (3.50)，其曲线为图 3.12。

$$W_{cubic}(\boldsymbol{r}_{ij}, h) = \alpha_D \begin{cases} 1 - \frac{3}{2} q^2 + \frac{3}{4} q^3 & 0 \leq q \leq 1 \\ \frac{1}{4} (2 - q)^3 & 1 \leq q \leq 2 \\ 0 & q \geq 2 \end{cases} \tag{3.50}$$

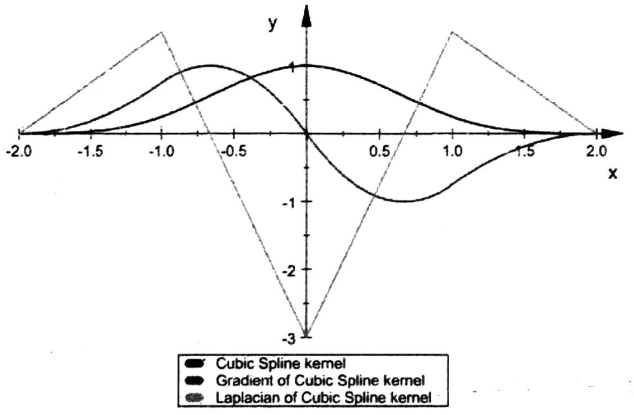


图 3.12 Cubic Spline 核函数曲线

cubic spline 核函数的好处是计算比较精确，也比较适合并行计算。

当然，和前面的牛顿流体一样，我们也可以把一些常量移除循环框架以外，进行预计算来加速：

$$\rho_i = m * W_{poly6}^{coeffs} \sum_{j \neq i} W_{poly6}^{variable} \tag{3.51}$$

$$f_i^{pressure} = -m * \nabla W_{spiky}^{coeffs} \sum_{j \neq i} \left(\frac{P_i}{\mu_i^2} + \frac{P_j}{\mu_j^2} \right) \nabla W_{spiky}^{variable} \tag{3.52}$$

$$f_i^{xsph} = 2m W_{poly6}^{coeffs} \sum_{j \neq i} \frac{\nu_j - \nu_i}{\mu_i + \mu_j} W_{poly6}^{variable} \tag{3.53}$$

$$f_i^{viscosity} = \mu * m * \frac{1}{\mu_i} * \nabla W_{spiky}^{coeffs} \sum_{j \neq i} \frac{\boldsymbol{v}_j - \boldsymbol{v}_i}{\rho_j} \nabla^2 W_{viscosity}^{varible} \tag{3.54}$$

$$\nabla \boldsymbol{v}_i = m \sum_j \frac{1}{\rho_j} (\boldsymbol{v}_j - \boldsymbol{v}_i) \otimes \nabla W_{cubic} \tag{3.55}$$

$$\tau = rheological function() \tag{3.56}$$

$$f_i^{stress} = \frac{m}{\rho_i} \sum_{j \neq i} \frac{\tau_i + \tau_j}{\rho_j} \cdot \nabla W_{cubic} \tag{3.57}$$

非牛顿力学 SPH 流体模拟相对复杂，它的求解需要分为三步，见图 (3.13), 因为剪切力的计算需要速度张量，而速度张量的计算又依赖于密度。相对于牛顿流体，因为涉及到 3×3 的张量矩阵计算，这也是复杂 SPH 模型比简单 SPH 模型效率低的主要原因。

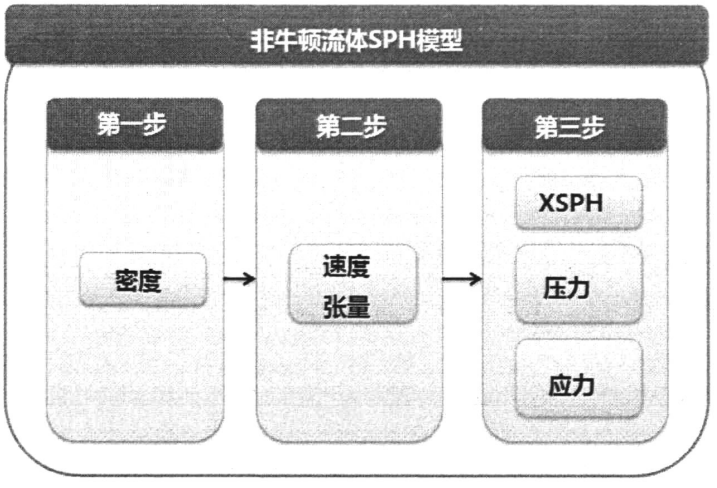


图 3.13 非牛顿流体 SPH 方法的三步计算

3.5.3 流变模型具体实现

在本文的系统框架中，我们实现了几种流变的模型，在上一节中，我们已经介绍过了几种流变模型，下面本文用代码来分析一下如何实现它们。所有的这些模型都需要变形张量 $\dot{\gamma}$ 和第二应变率不变量 $|\dot{\gamma}|$:

```
matrix3 deformation_tensor_i =
0.5*(velocity_tensor_i+transpose(velocity_tensor_i));
float t = trace(deformation_tensor_i);
float deformation_amount = sqrtf(t*t);
```

Cross 模型 (Cross Rheology) 实现，基本上就是公式 (3.29) 的方法，为了保证流体的稳定性，同样需要把粘性限定在一定范围以内。

```
float viscosity = K*pow(deformation_amount,n-1.0f);
viscosity = clamp(viscosity, 1.0f,300.0f);
stress_tensor = viscosity*deformation_tensor_i;
```

3.5.4 地形的生成和检测

考虑到流体粒子和山体地形的碰撞检测，我们需要计算任意给定点的地形的高度。其中地形数据以高度图的形式给定，如图 3.14 左。在高度图中，没一点的像素值代表了哪一点的高程值。有了地形的高度值，就可以对每一个粒子判断和地形的碰撞。但是仅仅有高度值，还不可以做碰撞检测，本文使用一张法向图，如图 3.14 右，来判断每一点的朝向，结合地形的面片信息，就可以算出面片的法向，如图 3.15，有了面片的高程信息和法向信息，就可以对粒子进行碰撞检测了。

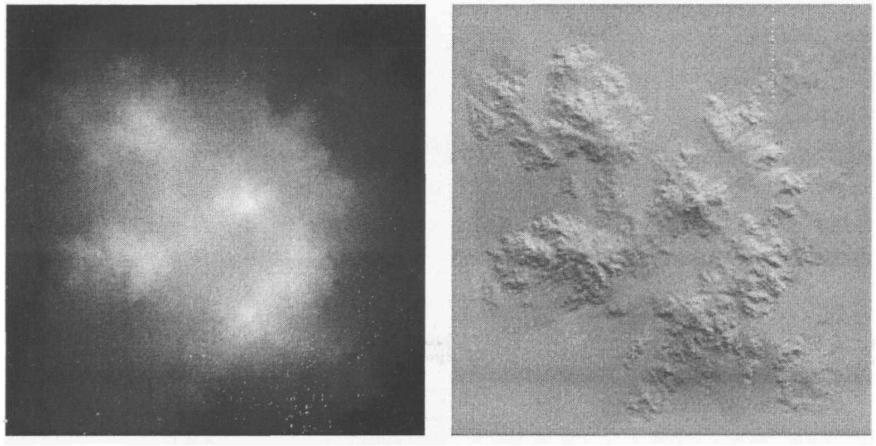


图 3.14 地形的高度图（左）和地形的法向图（右）

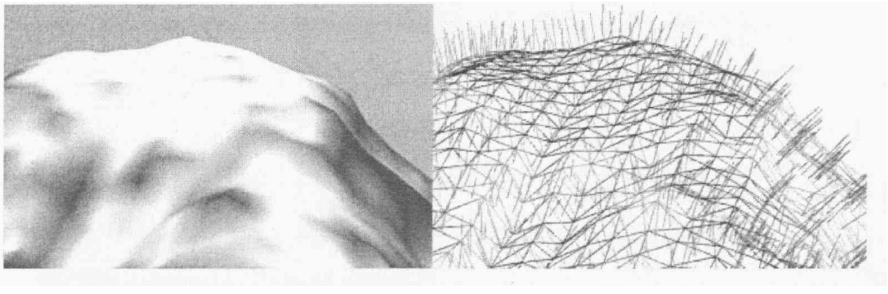


图 3.15 带有面片法向的地形图

3.5.5 各项力的整合

在 SPH 算法的最后阶段，就是要对各项力 (包括内部力，外部力)，当然在这一步，我们还有其他的计算，例如粒子颜色的计算。

对于时间步长积分，本文使用经典的 Leap-Frog 算法，一种具有二阶精度的迭代算法，在实际应用中，

$$x_{i+1} = x_i + v_{i+1/2}dt \tag{3.58}$$

$$v_{i+1/2} = v_{i-1/2} + a_i dt \tag{3.59}$$

其中 i 是时间步长，之所以被称为 Leap-Frog，是因为速度直接跳过了位置，见图 3.16。在 Leap-Frog 算法中，当对控制方程进行积分时，粒子的速度和位移以半个时间步长向前偏移。在第一个时间步长结束后，密度、速度将被用于将密度、速度往前推进半个时间步长，而粒子的位移则前进一个时间步长。

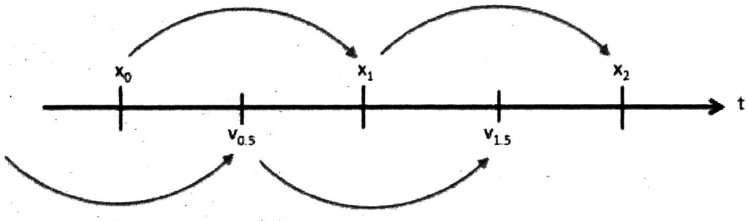


图 3.16　蛙跳算法 (Leap-Frog) 迭代

当然蛙跳算法还有另一种形式，所以的物理量都在整数的时间步长上：

$$x_{i+1} = x_i + v_i dt + \frac{a_i}{2}dt^2 \tag{3.60}$$

$$v_{i+1} = v_i + \frac{(a_i + a_{i+1})}{2}dt \tag{3.61}$$

蛙跳算法是对简单的欧拉方法求解和用复杂的高级数值算法求解的一个折中，是非常流行的一个方法。

为了保持非牛顿流体粒子运动过程的稳定性，避免通常牛顿流体在动态过程吕所产生的飞溅现象，在此我们特意引入 XSPH[47] 技术。XSPH 是 X-Factor SPH 的简称，其优点是计算量不是很大，粒子运动的稳定性显著增加。本节我们使用 XSPH 模型来平滑粒子的速度，其形式可表现为：

$$v_i = v_i + \epsilon \sum_{j \neq i} 2m_j \frac{(v_j - v_i)}{(\rho_i + \rho_j)} W(\boldsymbol{r}_{ij}, h) \tag{3.62}$$

XSPH 的核心思想就是用周围粒子的速度，对中心粒子来做平滑处理。其中 ϵ 是一个范围在 $0-1$ 之间的参数，通常取 0.5。

对于 SPH 算法来说，最耗时的核半径内的邻域粒子的查找上。本文所使用的是由 NVIDIA 官方提出并实现的一种基于哈希、基数排序、均匀网格的查找算法[4]。

通过把空间划分成均匀的网格，网格的划分粒度是粒子核半径，那么所处于空间网格某一点的粒子，只需要在邻域查找就可以了。下图 3.17 为 2D 均匀网格示意图。

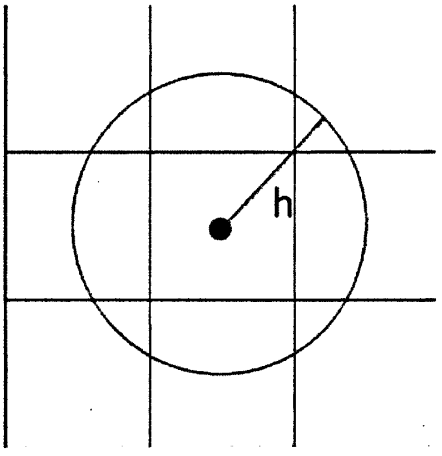


图 3.17 2D 均匀网格查找示意图

具体的算法步骤如下：

1. 把虚拟的空间划分成均匀网格。
2. 找到每个粒子所隶属于的空间网格。
3. 对每个粒子的空间网格位置进行了一次哈希求值。
4. 根据哈希值进行排序。
5. 把粒子的所以属性都调整到对应的位置。
6. 创建网格的空间索引，来跟踪记录每一个小网格在排序数组中的起始位置和结束位置。

隶属于同一空间网格的的粒子在排序的线性数组中，紧密的排列在一起。对于每一个空间网格，查找对应的空间索引就可以找到起始位置和结束位置所谓的邻域粒子查找就只需要遍历一遍数组就可以了。

[4]http://www.nvidia.com/object/cuda_get.html

对于其中的排序部分，本文使用了适合于GPU并行计算的基数排序[48]。

其实本方法也适合用来模拟其它的一些贴地流(包括泥石流和火山爆发的岩浆流等)。下面一节我们将对模拟的绘制结果进行展示和分析。

3.6 绘制结果

根据上述建模和计算的结果，本文在Intel(R) Core(TM) i7-2600 CPU @ 3.40GHZ，8.0GB内存，NVIDIA Geforce GTX 580显卡的高端PC机上实现了百万级别粒子的SPH实时模拟并实现对具有流变特性的雪崩现象的实时模拟。其中GPU并行计算使用了CUDA架构，绘制部分由OGRE绘制引擎完成。

CUDA(Compute Unified Device Architecture)，显卡厂商NVIDIA推出的运算平台。CUDA是一种由NVIDIA推出的通用并行计算架构，该架构使GPU能够解决复杂的计算问题。它包含了CUDA指令集架构（ISA）以及GPU内部的并行计算引擎。

OGRE（Object-Oriented Graphics Rendering Engine，即：面向对象图形渲染引擎）是一个用C++开发的面向场景、非常灵活的3D引擎，它旨在让开发人员更容易、更直接地利用硬件加速的3D图形系统开发应用。这个类库隐藏了底层系统库（如：Direct3D和OpenGL）的所有细节，提供了一个基于世界对象和其他直观类的接口。

图3.18为基于Müller[1]的方法，$256K$粒子简单牛顿流体的模拟，其中粒子速度的梯度对大，粒子的颜色越鲜艳。

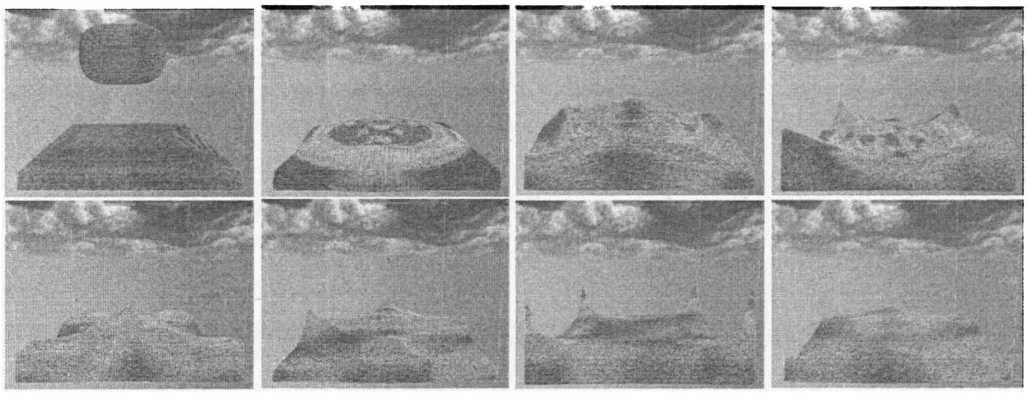


图3.18 使用简单的SPH模型，256K的粒子模拟时间序列（次序从左到右，自上而下，以下同）

图3.19为$256K$简单牛顿流体的粒子滑下山坡的时间序列图。

从这些图中可以看出，对牛顿流体的模拟，会产生液体状的粒子快速扩散的现象，不

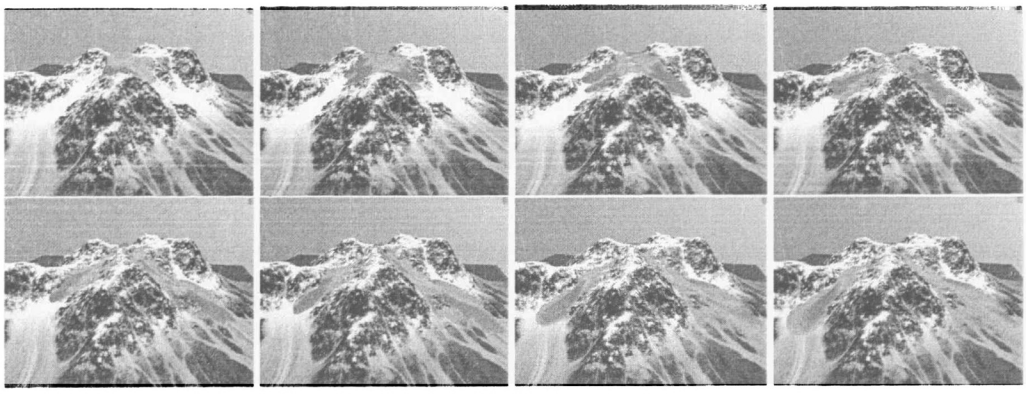


图 3.19 使用简单的 SPH 模型，256K 的粒子从山顶开始下滑时间序列

能产生“滚雪球”式的抱团状推动式前进的效果，这不符合雪崩运动的特点，故在本文中不宜采用。而一定要采用非牛顿流体模型来模拟雪崩景象。

为了表现出雪崩流动的场景，本文用高度图自动生成相对陡峭的山坡，通过使用合适的摩擦系数、流变模型、流变参数，本文所采用的方法可以生成雪崩向下滑动的景象。了从视觉上接近雪蹦，本文把所有 SPH 粒子渲染成白色。通过多次实验，我们发现时间步长的设置非常重要，越高的粘稠度就需要越小的时间步长。实验发现在取最大粘滞系数为 300 时，我们可以使用的时间步长是 0.0005，这种情况下还可以保证粒子流动的稳定性。使用 GTX 580 的高端显卡，对于 512K 粒子的模拟，可以达到 25fps，带到实时的要求。

使用 128K 的粒子，使用 0.001 的时间步长，采用的流变模型把最大的粘滞力系数定为 100，可以保证雪崩粒子的稳定性。

例使用 Cross 模型的来模拟雪崩的非牛顿流体特性，图 3.20 展示了最终雪崩过程模拟的时间序列图，从中可以看出，雪崩流体中粒子的飞溅情况不再出现，其稳定性得到了较好地改进。

3.7 本章小结

本章首先对现有的雪崩建模方法进行介绍，阐述了流体模拟及平滑粒子流体动力学 (SPH) 方法的基本原理。并采用了计算流体动力学方法对雪崩景象进行建模，对比了牛顿流体及非牛顿流体的区别并进行了大量的模拟实验。实验表明牛顿流体模型不适合于具有粘滞力流变特性的雪崩景象的模拟。由于速度张量 Cross 模型能较好地表示雪崩过程中的非牛顿流体特点，我们引入 Cross 模型以模拟此运动过程中的粘滞力的流变特性。采用未知因素修正的 SPH （XSPH）模型来避免了传统流体模拟中粒子飞溅的效果，保持了雪崩

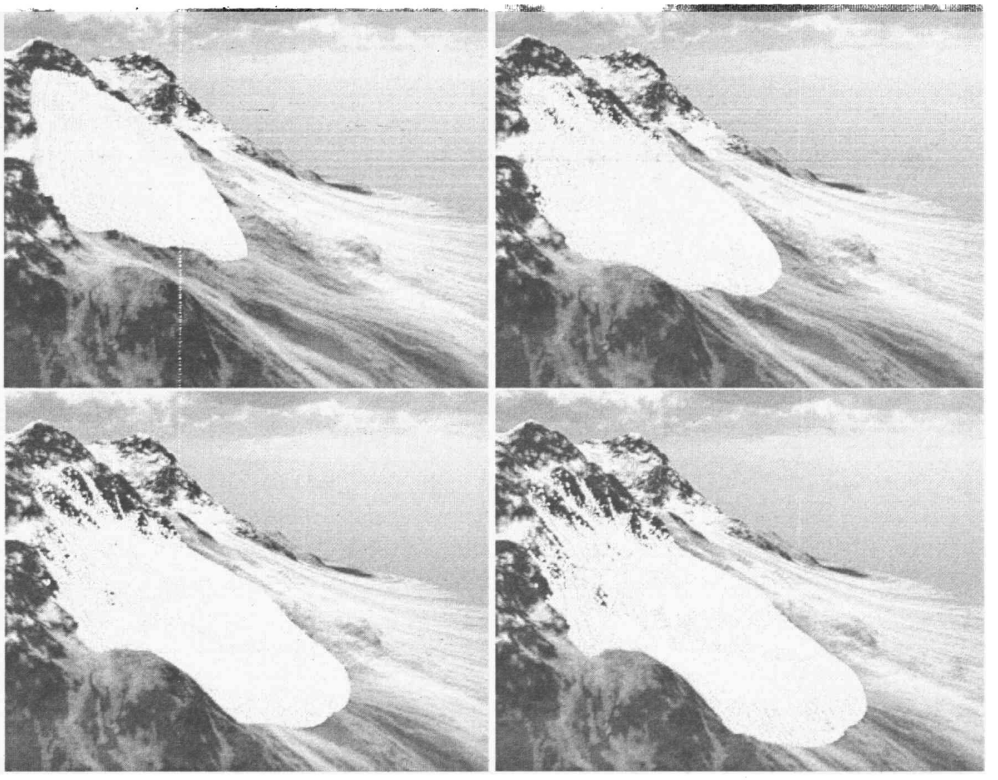


图 3.20 雪崩景象实时模拟的视频序列图，其中 Cross 模型中的参数为：
$\mu_\infty = 1, \mu_0 = 100, n = 2, K = 3000$,动摩擦系数为 0.2，$256K$ 的粒子

运动的稳定性；并通过优化设计的 GPU 加速方案，初步实现了百万量级粒子参与的较为复杂的雪崩景象的实时模拟。但目前本算法尚未能模拟由于雪崩过程中由于雪粒子间激烈碰撞而产生的高速“雪云”的雾状效果，光照及传输衰减的效果也较为初步，因此场景的真实感还有待进一步进行改进。

第4章 总结与展望

4.1 本文的工作总结

本章首先对全文的工作进行总结；再对目前所存在的问题进行分析，并对未来的工作进行展望，指出了进一步深入研究的方向。

雪场景因其美轮美奂的妙姿而深受人们的喜爱。近年来随着气候变化的加剧，各地大雪景象甚至雪灾频频出现；另外冬天的冰雪运动也吸引着越来越多的人参加，因此雪场景的建模与绘制工作有着现实的意义。另外，该技术的研究进展，在虚拟现实、军事演练、运动仿真、雪灾的预防和救援、影视特技制作、计算机动画及游戏制作等众多领域有着广泛的应用价值。

本文围绕着雪场景的真实感建模与实时绘制这一主题展开研究，主要包含了大规模雪场景建模与实时绘制及雪崩景象实时模拟这两个方面。本文的主要贡献总结如下：

- 提出了一种交互式大规模雪场景建模与实时绘制的新方法。该方法能高效地进行雪场景的建模，可定量地实时生成不同积雪量及飘雪效果的大规模雪场景。
- 提出了一种基于视点的自适应降雪遮挡图模型，该模型解决以往全局场景降雪遮挡图方法所引起的大尺度场景区域中积雪效果失真的缺点，可有效地提高大规模雪场景积雪的真实感，并可推广到对动态物体进行实时积雪的效果。
- 在计算机图形学领域里首次实现了对雪崩动态景象的模拟。提出并实现了一种基于计算流体动力学的实时雪崩模拟的新算法。该算法基于雪崩形成的物理机制，在平滑粒子动力学 (SPH) 计算框架内，采用非牛顿流体模型对雪崩运动过程进行有效地建模；以速度张量模型来体现雪崩运动过程中粘滞力的时变特性，从而实现了雪崩动态景象的模拟。
- 将未知因素修正的 SPH（XSPH）模型引入雪崩景象的模拟中，从而避免了传统流体模拟中粒子飞溅的效果，保持了雪崩运动的稳定性；并通过优化设计的 GPU 加速方案，成功地实现了百万量级粒子的较为复杂的雪崩景象的实时模拟。

4.2 未来工作的展望

虽然本文的工作取得了一些成果，但由于研究时间的限制，本工作还存在一些不足，希望在今后的工作中加以改进。具体列于下面：

在大规模雪场景的建模与实时绘制方面：

- 本方法目前对形状较为复杂且造型精度较高的地物目标（如：树枝），其交互操作产生的积雪厚度效果较好，而对于造型较为简单的大规模地物（如道路、屋顶等），其交互积雪厚度效果反而不够好，若增加其网格复杂度则势必降低交互操作的效率。因此有必要研究对此类地物的高效交互积雪效果新算法。
- 将积雪效果算法推广至动态物体上。比如：一汽车刚从车库驶出时身无积雪，随着在大雪场景中行驶时间的增加，其背上积雪厚度越来越高等。
- 对三维雪场景风雪交互机制进行深入研究，提高其交互作用的效率，并在保证其实时性的前提下增强场景的真实感。
- 改进雪场景的光照模型以增强其真实感，并进而模拟时变的积雪及其融化的过程。
- 对积雪的堆积和融化过程进行模拟。

在雪崩的模拟研究方面：

- 对雪崩的触发条件进行研究。
- 对雪崩所形成的“雪云”景观进行多相流建模及真实感绘制。
- 模拟雪崩景象在行进过程中与障碍物的相互作用，如对房屋及村庄的吞没及摧毁等。

参考文献

[1] M. Müller, D. Charypar, M. Gross. Particle-based fluid simulation for interactive applications. In Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation. Eurographics Association, 2003, 154–159.

[2] Tae-Yong Kim, Lucio Flores. Snow avalanche effects for mummy 3. In ACM SIGGRAPH 2008 talks. ACM, 2008, 7.

[3] P. Ohlsson, S. Seipel. Real-time rendering of accumulated snow. In Sigrad Conference. 2004, 25–32.

[4] Changbo Wang, Zhangye Wang, Qi Zhou, Chengfang Song, Yu Guan, Qunsheng Peng. Dynamic modeling and rendering of grass wagging in wind[J]. Computer Animation and Virtual Worlds, 2005, 16(3-4):377–389.

[5] 张龙, 张钰勃, 陈为, 何戡, 丁子昂, 王章野, 彭群生. 湿地场景的实时动态模拟[J]. 计算机辅助设计与图形学学报, 2008, 20(8):1007–1014.

[6] Christian Dick, Jens Krüger, Rüdiger Westermann. Gpu ray-casting for scalable terrain rendering. In Proceedings of EUROGRAPHICS. Citeseer, 2009, 43–50.

[7] Wang Changbo, Zhangye Wang, Xin Zhang, Lei Huang, Zhiliang Yang, Qunsheng Peng. Real-time modeling and rendering of raining scenes[J]. The Visual Computer, 2008, 24(7):605–616.

[8] C. Wang, Z. Wang, T. Xia, Q. Peng. Real-time snowing simulation[J]. The Visual Computer, 2006, 22(5):315–323.

[9] Shiguang Liu, Zhangye Wang, Zheng Gong, Lei Huang, Qunsheng Peng. Physically based animation of sandstorm[J]. Computer Animation and Virtual Worlds, 2007, 18(4-5):259–269.

[10] Claes Johanson, Calle Lejdfors. Real-time water rendering[J]. Lund University, 2004.

[11] Yaohua Hu, Luiz Velho, Xin Tong, Baining Guo, Harry Shum. Realistic, real-time rendering of ocean waves[J]. Computer Animation and Virtual Worlds, 2006, 17(1):59–67.

[12] Qizhi Yu, Fabrice Neyret, Eric Bruneton, Nicolas Holzschuch. Scalable real-time animation of rivers. In Computer Graphics Forum. Wiley Online Library, 2009, volume 28, 239–248.

[13] Shiguang Liu, Zhangye Wang, Zheng Gong, Qunsheng Peng. Real time simulation of a tornado[J]. The Visual Computer, 2007, 23(8):559–567.

[14] Changbo Wang, Zhangye Wang, Qunsheng Peng. Real-time rendering of sky scene considering scattering and refraction[J]. Computer Animation and Virtual Worlds, 2007, 18(4-5):539–548.

[15] Liu Shiguang, Wang Zhangye, Gong Zheng, Wang Changbo, Peng Qunsheng. Physically based simulation of buddha glory[J]. Progress in Natural Science, 2006, 16(6):656–662.

[16] T. Nishita, H. Iwasaki, Y. Dobashi, E. Nakamae. A modeling and rendering method for snow by using metaballs[J]. Computer Graphics Forum, 1997, 16(3):357–364.

[17] P. Fearing. Computer modelling of fallen snow. In Proceedings of the 27th annual conference on Computer graphics and interactive techniques. ACM Press/Addison-Wesley Publishing Co., 2000, 37–46.

[18] S. Premoze, W. Thompson, P. Shirley. Geospecific rendering of alpine terrain. In Eurographics Workshop on Rendering. 1999, 107–118.

[19] 陈彦云, 孙汉秋. 自然雪景的构造和绘制[J]. 计算机学报, 2002, 25(9):916–922.

[20] B.E. Feldman, J.F. O'Brien. Modeling the accumulation of wind-driven snow. In ACM SIGGRAPH 2002 conference abstracts and applications. ACM, 2002, 218–218.

[21] T.K. Thiis. Large scale studies of development of snowdrifts around buildings[J]. Journal of Wind Engineering and Industrial Aerodynamics, 2003, 91(6):829–839.

[22] H. Haglund, M. Andersson, A. Hast. Snow accumulation in real-time[J]. Proceedings of SIGRAD, 2002, 2002:11–15.

[23] T.B. Moeslund, C.B. Madsen, M. Aagaard, D. Lerche. Modeling falling and accumulating snow. In Vision, Video and Graphics. Heriot Watt University, Edinburgh, The Eurographics Association, 2005, volume 2.

[24] W.T. Reeves. Particle systems—a technique for modeling a class of fuzzy objects. In ACM SIGGRAPH Computer Graphics. ACM, 1983, volume 17, 359–375.

[25] M. Langer, Q. Zhang. Rendering falling snow using an inverse fourier transform[J]. ACM SIGGRAPH technical sketches program, 2003.

[26] 刘波, 王章野, 王丽英, 华炜, 彭群生. 大规模城市场景的高效建模及其实时绘制[J]. 计算机辅助设计与图形学学报, 2008, 20(9):1153–1162.

[27] L. Latta, et al. Building a million particle system. In Game Developers Conference. 2004, volume 2004.

[28] M.J. Smith. Sandstorm: A Dynamic Multi-contextual GPU-based Particle System using Vector Fields for Particle Propagation[M]. ProQuest, 2008.

[29] EJ Hopfinger. Snow avalanche motion and related phenomena[J]. Annual Review of Fluid Mechanics, 1983, 15(1):47–76.

[30] J. Schweizer, J.B. Jamieson, M. Schneebeli. Snow avalanche formation[J]. Reviews of Geophysics, 2003, 41(4):1016.

[31] J. Schweizer, JB Jamieson. Snow cover properties for skier triggering of avalanches[J]. Cold Regions Science and Technology, 2001, 33(2):207–221.

[32] E. Troshkina, T. Glazovskaya, N. Kondakova, V. Sokolov. Zoning of snowiness and avalanching in the

mountains of western transcaucasia[J]. Annals of Glaciology, 2001, 32(1):311–313.

[33] M. Barbolini, F. Cappabianca, F. Savi. A new method for the estimation of avalanche distance exceeded probabilities[J]. Surveys in geophysics, 2003, 24(5):587–601.

[34] B. Jamieson, C.D. Johnston. Snowpack factors associated with strength changes of buried surface hoar layers[J]. Cold Regions Science and Technology, 1999, 30(1):19–34.

[35] P. Gauer, K. Kronholm, K. Lied, K. Kristensen, S. Bakkehøi. Can we learn more from the data underlying the statistical α– β model with respect to the dynamical behavior of avalanches[J]. Cold Regions Science and Technology, 2010, 62(1):42–54.

[36] W.T. Reeves, D.H. Salesin, R.L. Cook. Rendering antialiased shadows with depth maps. In ACM SIGGRAPH Computer Graphics. ACM, 1987, volume 21, 283–291.

[37] K. Perlin. An image synthesizer[J]. SIGGRAPH Comput. Graph., 1985, 19(3):287–296.

[38] L.B.W.Z.W. Liying, H.W.P. Qunsheng. Efficient modeling and real-time rendering of large-scale urban scenes[J]. Journal of Computer-Aided Design & Computer Graphics, 2008, 9:011.

[39] Christophe Ancey. Snow avalanches[J]. Geomorphological Fluid Mechanics, 2001, 319–338.

[40] D. Enright, S. Marschner, R. Fedkiw. Animation and rendering of complex water surfaces[J]. ACM Transactions on Graphics (TOG), 2002, 21(3):736–744.

[41] M. Kelager. Lagrangian fluid dynamics using smoothed particle hydrodynamics[J]. Graduate project at DIKU under supervision of assistant professor Kenny Erleben, 2006.

[42] J.J. Monaghan. Smoothed particle hydrodynamics[J]. Annual review of astronomy and astrophysics, 1992, 30:543–574.

[43] N. Balmforth, R. Craster. Geophysical aspects of non-newtonian fluid mechanics[J]. Geomorphological Fluid Mechanics, 2001, 34–51.

[44] JM Sac-Epee, K. Taous. On a wide class of nonlinear models for non-newtonian fluids with mixed boundary conditions in thin domains[J]. Asymptotic Analysis, 2005, 44(1):151.

[45] SM Hosseini, MT Manzari, SK Hannani. A fully explicit three-step sph algorithm for simulation of non-newtonian fluid flow[J]. International Journal of Numerical Methods for Heat & Fluid Flow, 2007, 17(7):715–735.

[46] A. Paiva, F. Petronetto, T. Lewiner, G. Tavares. Particle-based viscoplastic fluid/solid simulation[J]. Computer-Aided Design, 2009, 41(4):306–314.

[47] JJ Monaghan. On the problem of penetration in particle methods[J]. Journal of Computational physics, 1989, 82(1):1–15.

[48] N. Satish, M. Harris, M. Garland. Designing efficient sorting algorithms for manycore gpus. In Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on. IEEE, 2009, 1–10.

攻读硕士学位期间主要的研究成果

1. [1] 单宇翔，王章野，杨春燕，彭群生. 大规模雪场景的实时绘制，计算机辅助设计与图形学报，2013，25 (EI 检索期刊，将发表)

致 谢

时间过的飞快，我的研究生学习阶段即将完成，也将离开我学习了快七年的浙江大学。回顾这几年来学习与生活中的点点滴滴，我由衷的感谢这几年来关心、帮助过我的师长、同学和朋友们，正因为有你们的指点，我才能顺利的完成研究生的学业。

首先，我要感谢我的导师王章野副教授，细致入微的科研指导，他扎实勤勉的生活作风，讲求实效的工作态度，都使我受益匪浅。不管在学习还是生活中，王章野老师都始终如一地给予我关心和帮助，论文的完成中也凝聚着他很多的心血。我很荣幸能在攻读硕士学位期间师从王章野老师，籍此论文完成之际，谨向我的导师致以衷心的感谢！

其次，我要感谢彭群生教授，认真的帮我修改 CAD 学报的论文，仔细斟酌，改了又改，我们见识到了老一辈学术工作者认真的态度。在两年来的课堂和讨论班上，他深厚的学术功底，严谨的工作作风，敏锐的学术洞察力，都深深的感染了我，使我受益良多。

再次，感谢自然场景组的成员，宫立山、张昆、任治国、张海涛、廖继渊、王丰金硕士，你们的帮助使我能够在两年来一次又一次顺利的完成项目和课题。同样要感谢张鑫、刑冠宇博士，林奶养、袁霞、张婷、陈蕾英硕士等同学，正是你们在学习和生活中对我的支持，才使我不断的进步。

还要感谢浙江大学 CAD&CG 国家重点实验室为我们提供良好宽松的学术研究环境、轻松自由的讨论氛围和优越的软硬件条件。

最后感谢我的父母在生活中对我无微不至的关怀和始终如一的支持，使我可以安心于学术与科研。

单宇翔

二零一三年一月于浙大紫金港