文章编号：2095-2163(2021)11-0171-04 中图分类号：TP391.9 文献标志码：A

基于 Perlin 噪声算法的海面仿真

孙攀攀，郑凯东
（西安石油大学 计算机学院，西安 710065）

摘 要：针对随机点生成高度在海面模拟中浪尖处易失真的问题，尝试用 Perlin 噪声生成高度，并利用插值函数进行插值来模拟海浪。根据分辨率进行晶格的构建，计算各顶点的噪声值；通过插值函数，生成平滑的海浪。实验结果表明，该方法不仅可以高效模拟海面，而且能充分地表现海浪连续平滑的特征，解决了海浪浪尖处容易过于尖凸而出现的失真问题。

关键词：海面模拟；Perlin 噪声；网格构建

Sea surface simulation based on Perlin noise algorithm

SUN Panpan, ZHENG Kaidong
(School of Computer Science, Xi ´an Shiyou University, Xi ´an 710065, China)

【Abstract】 Aiming at the problem that the height generated by the random midpoint displacement method based on the geometric construction method is not only cumbersome in the drawing process, but also easy to be distorted at the top of the wave in the sea surface simulation, we try to use Perlin noise to generate the height, and use the interpolation function to perform interpolation to simulate the waves. First, the lattice is constructed according to the resolution, and then the noise value of each vertex is calculated, and then a smooth ocean wave is generated through an interpolation function. The experimental results show that this method can not only simulate the sea surface efficiently, but also fully express the continuous and smooth characteristics of the waves, and solve the problem of distortion that is prone to be too sharp and convex at the wave tip.

【Key words】 sea surface simulation; perlin noise; the grid construction

0 引 言

近年来，自然景物的仿真越来越受人们的青睐，其中最受关注的热点之一就是流体的模拟。现在研究最多的流体有：烟、云、雾和水等。众所周知，中国的海洋面积覆盖率高达 70%。因此，海面仿真对于虚拟自然环境是非常重要的，已被广泛应用于虚拟仿真、游戏娱乐、动漫影视等领域中。目前，国内外用于生成海浪较为常用的方法有：基于物理模型的方法、基于几何构造的方法、基于海浪谱的方法[1]。基于物理模拟的方法主要以 Navier-Stokes(N-S)方程为基础，又可分为两种[2]：一种是欧拉法(基于网格的方法)；另一种是拉格朗日法(基于粒子的方法)。基于物理模型的方法主要用于小范围精细尺度或特定场景的模拟。例如：水花飞溅、泡沫、喷泉、流体倾倒等。基于几何构造的方法主要是利用数学函数构造出海浪的轮廓，例如：正弦函数、随机中点法、Beta-样条曲线等。基于海浪谱的方法主要是以微小振幅波理论和有限振幅波理论为基础，把真实的海面波动看作是由许多不同振幅、周期、相位波动的叠加，利用波浪谱和方向谱对波浪模型设定参数，一般用快速傅里叶变换(FFT)对简单波动进行叠加。波浪谱一般采用 Phillips 波浪谱或者 P-M 谱和方向谱或者文氏方向谱等。其中基于几何构造方法中，随机点生成高度的海面模拟中浪尖处容易失真。针对此问题，本文利用海面网格建模[3]，再利用 Perlin 噪声生成高度后，用缓和曲线插值函数进行插值来模拟海浪，从而解决了因浪尖过于尖凸出现的失真问题。

1 基于网格的海面建模

网格是计算机图形学中表示三维图形最基础的方法，而构成网格模型的基础单元是一个个三角面。网格模型的核心数据是点表、UV 表和三角面表。其中，点表是构成整个网格模型顶点的坐标集合，每个坐标都是一个三维向量，表示该顶点在坐标系中的空间位置；UV 表是构成网格模型顶点在进行贴图纹理映射时的坐标，每个坐标都是一个二维向量，

作者简介：孙攀攀(1998-)，女，硕士研究生，主要研究方向：图形图像处理、自然景物模拟；郑凯东(1964-)，男，硕士，副教授，主要研究方向：图形学与虚拟现实、程序设计、计算机基础教育。
通讯作者：郑凯东 Email：kdzheng@ xsyu.edu.cn

收稿日期：2021-08-12

并与点表中的顶点一一对应,表示该顶点在进行贴图纹理映射时在二维贴图中的平面位置;三角面表是构成网格模型三角面顶点序号的集合,其是一个整数集合,其中的数值表示该顶点在点表中的序号,每 3 个值为一组,表示一个三角面。

在 Unity 中创建一个网格模型的具体步骤如下:

Step 1 在 Unity 中新建一个 GameObject,命名为 OceanSurface,并为其添加 MeshFilter 组件和 MeshRenderer 组件。MeshFilter 组件负责为渲染器提供网格模型数据,后续的网格模型创建工作主要是基于此组件;MeshRenderer 组件则是负责网格模型的渲染。

Step 2 在 Unity 工程目录中创建 OceanSurface MeshGenerator 脚本文件,并为 GameObject 挂载此脚本。本文中海面网格模型生成的主要工作将在此脚本中完成。

Step 3 在代码编辑器中打开 OceanSurfaceMesh Generator 脚本文件,新建函数 GenerateOceanSurface Mesh,并将其引用在脚本中的默认函数 *Start* 中。在 *Start* 函数中引用此函数的目的是为了在程序启动时立即调用此函数来生成海面网格模型。

Step 4 在脚本中定义海面网格模型计算所需变量。其中包括:海面网格模型的宽度 x、海面网格模型的长度 y、海面网格模型的精度 *precision*。*precision* 变量将决定生成海面网格模型的平滑程度,其值越大则越平滑,效果也越细腻,但相应地也会消耗更多的计算机算力。

Step 5 计算网格模型。根据 x、y、*precision* 变量的值,计算出网格模型的点表、*UV* 表以及三角面表。

网格数据计算公式如下:

(1)顶点索引:

$$n = y * (precision + 1) + x \tag{1}$$

顶点索引按照先 X 方向后 Z 方向的顺序计算。

(2)顶点坐标:

$$vertices[n] = \text{new Vector3}(-\text{size}.x/2 + \text{size}.x/precision * x, 0, -\text{size}.y/2 + \text{size}.y/precision * y); \tag{2}$$

Unity 中的坐标系统采用的是左手坐标系统,X 轴表示水平方向,Y 轴表示垂直方向,Z 轴表示深度。因需要构建的是水平平面海面,所以 Y 轴的值为 0;而 X 轴和 Z 轴坐标分别减去 1/2 倍总长度的作用,是为了使最终生成平面的网格中心仍位于原点处。

(3) *uv* 坐标

$$uv = new\ Vector2[(precision + 1) * (precision + 1)] \tag{3}$$

$$n = y * (precision + 1) + x \tag{4}$$

$$uv[n] = new\ Vector2(1f/precision * x, 1f/precision * y) \tag{5}$$

由于 *uv* 坐标与网格中的顶点一一对应,所以 *uv* 坐标中的数量和网格的顶点数相同。

(4)三角形:

$$\text{triangles} = \text{new int}[precision * precision * 6] \tag{6}$$

Unity 中的网格都是三角形构成的,把所需要的三角形保存在 Mesh 组件的 triangles 中。三角形的总数为 *precision* * *precision* * 6。

Step 6 网格的生成。对于分辨率为 *precision* * *precision* 的网格构建流程如图 1 所示。

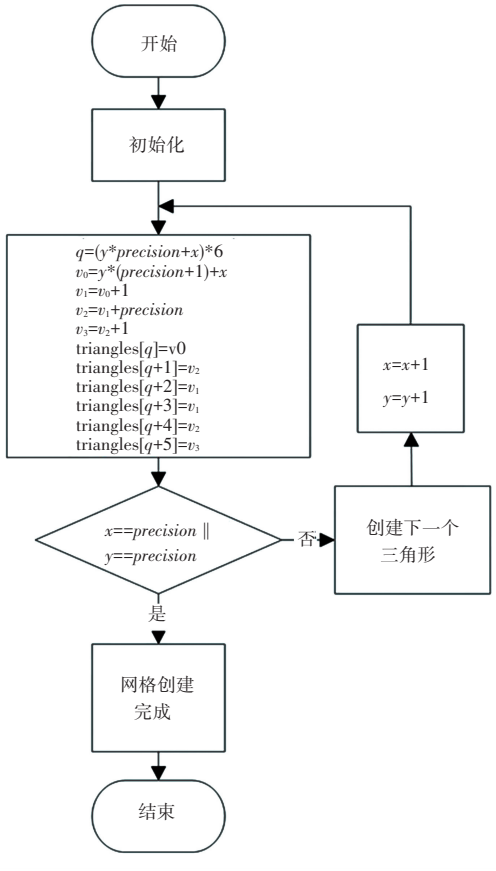

图 1 网格构建流程图

Fig. 1 Grid Construction Flowchart

图 1 中,q 表示构成三角形的顶点序列,V_0、V_1、V_2、V_3 表示四边形的顶点。

Step 7 提交网格模型数据。其实现过程如下所示。

定义 Unity 中的网格模型对象：

Mesh mesh = new Mesh()　　(7)

将计算得出的网格模型数据赋值给 mesh 对象：

mesh.vertices = vertices; mesh.uvs = uvs; mesh.triangles = triangles;　　(8)

将网格模型提交 MeshFilter 组件：

gameObject.GetComponent<MeshFilter>().mesh = mesh;　　(9)

在完成海面基础网格模型的生成后运行 Unity，将在 Unity 的 Game 视图看到生成的海面网格模型。因为还没有加入海面高度场，所以此时的海面还是一个没有任何波澜的平面。

2　海面高度场生成

2.1　Perlin 噪声算法

Perlin 噪声函数是伦敦大学的 Perlin.K 先生提出的自然噪声模拟算法[4]。该噪声函数的功能类似于随机数生成器，其每次返回的结果都是随机的，没有任何规律可寻。*Perlin* 噪声函数用一个整数作为参数，然后返回一个基于这个参数的随机数。如果把同样的参数传递两次，其会产生相同的随机数[5]。由于随机中存在一定的规律，所以被称为伪随机。正是因为其这个特征，所以被用于模拟一些有规律的场景中。如：火焰、海浪、云、木头等。

2.2　生成海面高度场

利用二维 *Perlin* 噪声模拟海面的三维高度场的方法如下：

首先，生成梯度向量。根据生成的网格划分晶格，在晶格内任选一点 $P(x,y)$，则 P 所在晶格的顶点坐标为 $P_0(i, j+1)$、$P_1(i,j)$、$P_2(i+1, j)$、$P_3(i+1, j+1)$；之后在 P 的 4 个顶点处随机生成梯度向量，用 g_{01}、g_{00}、g_{10}、g_{11} 代替。具体见图 2。

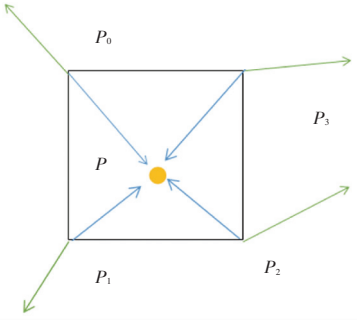

图 2　梯度向量生成图

Fig. 2　Generation of gradient vector

其中，梯度向量一般用(-1,1)、(0,1)、(1,1)、(-1,0)、(1,0)、(-1,-1)、(0,-1)、(1,-1)等 8 个向量表示。梯度向量也是伪随机的，同一个点的梯度向量是固定的。

其次，两两向量点乘。分别求出 4 个顶点到 P 的向量。即

$$\begin{aligned}\overrightarrow{p_0p} &= (x,y) - (i,j+1)\\ \overrightarrow{p_1p} &= (x,y) - (i,j)\\ \overrightarrow{p_2p} &= (x,y) - (i+1,j)\\ \overrightarrow{p_3p} &= (x,y) - (i+1,j+1)\end{aligned}\tag{10}$$

然后，分别用 4 个梯度向量与指向 P 的 4 个向量进行点乘，此时得到四个顶点对 P 的影响值。

$$\begin{cases}a = g_{01} * \overrightarrow{p_0p}\\ b = g_{00} * \overrightarrow{p_1p}\\ c = g_{10} * \overrightarrow{p_2p}\\ d = g_{11} * \overrightarrow{P_3P}\end{cases}\tag{11}$$

式(8)中，g_{01} 为点 $P_0(i, j+1)$ 的梯度；g_{00} 为点 $P_1(i, j)$ 的梯度；g_{10} 为点 $P_2(i+1, j)$ 的梯度；g_{11} 为点 $P_3(i+1, j+1)$ 的梯度。

最后，用缓和曲线代入坐标进行线性插值，即可得到 P 点的值。其中，缓和曲线有 $3x^2-2x^3$ 和 $6x^5-15x^4+10x^3$ 两种。缓和曲线项的次数越高，生成的海浪就越平滑，本文采用 $6x^5-15x^4+10x^3$ 插值公式。二维柏林噪声函数需要进行 3 次插值计算。首先需要对 P_0 和 P_3 两点在 x 方向进行插值，得到插值结果 s_1 后，对 P_1 和 P_2 两点在 x 方向进行插值，得到插值结果 s_2；最后对 s_1 和 s_2 在 y 方向插值，得到 P 的噪声函数值。具体算法如下：

设：$p = x - i$、$q = y - j$，在 x 方向上对 a 和 d、b 和 c 分别进行调和插值，插值结果为 s_1 和 s_2。则：

$$s_1 = a - (6p^5 - 15p^4 + 10p^3) * (a - d)\tag{12}$$

$$s_2 = b - (6p^5 - 15p^4 + 10p^3) * (b - c)\tag{13}$$

在 y 方向上对 s_1 和 s_2 进行调和插值，得到 P 的噪声函数值为：

$$s = s_1 - (6q^5 - 15q^4 + 10q^3) * (s_1 - s_2)\tag{14}$$

3　仿真结果与分析

为了验证 Perlin 噪声算法生成海面的效果，本文进行了模拟仿真实验。

实验硬件环境为：英特尔酷睿处理器，4 G 内存，英伟达显卡，Window10 64 位操作系统。

实验参数设定为：范围长 20 m、宽 20 m；海水颜

（下转第 179 页）