水面模拟算法理论

段元兴

2019 年 4 月 30 日

1 公式推导

在简化近似下,水面的波动可以近似为一个二维弹性膜的振动.于是可以由二维波动方程描述:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u. \tag{1}$$

在笛卡尔直角坐标系下可以写作:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right). \tag{2}$$

而对于我们的正方形游泳池模型,可以得到如下边界条件:

$$\begin{cases} \frac{\partial^2 u}{\partial t^2} - c^2 \frac{\partial u}{\partial x}\Big|_{x=0} = 0 \\ \frac{\partial^2 u}{\partial t^2} + c^2 \frac{\partial u}{\partial x}\Big|_{x=a} = 0 \\ \frac{\partial^2 u}{\partial t^2} - c^2 \frac{\partial u}{\partial y}\Big|_{y=0} = 0 \\ \frac{\partial^2 u}{\partial t^2} + c^2 \frac{\partial u}{\partial y}\Big|_{y=a} = 0 \end{cases}. \tag{3}$$

将方程差分化可得:

$$\begin{cases} a_{ijk} = \frac{c^2}{(\Delta a)^2} * \begin{cases} u_{i+1,jk} + u_{i-1,jk} + u_{i,j+1,k} + u_{i,j-1,k} - 4u_{ijk} & 0 < i < n_x - 1, \quad 0 < j < n_y - 1 \\ u_{i+1,jk} + u_{i,j+1,k} + u_{i,j-1,k} - 3u_{ijk} & i = 0, \quad 0 < j < n_y - 1 \\ u_{i-1,jk} + u_{i,j+1,k} + u_{i,j-1,k} - 3u_{ijk} & i = n_x - 1, \quad 0 < j < n_y - 1 \\ u_{i,j+1,k} + u_{i+1,jk} + u_{i-1,jk} - 3u_{ijk} & 0 < i < n_x - 1, \quad j = 0 \\ u_{i,j-1,k} + u_{i+1,jk} + u_{i-1,jk} - 3u_{ijk} & 0 < i < n_x - 1, \quad j = n_y - 1 \\ u_{i+1,jk} + u_{i,j+1,k} - 2u_{ijk} & i = 0, \quad j = 0 \\ u_{i+1,jk} + u_{i,j-1,k} - 2u_{ijk} & i = 0, \quad j = n_y - 1 \\ u_{i-1,jk} + u_{i,j+1,k} - 2u_{ijk} & i = n_x - 1, \quad j = 0 \\ u_{i-1,jk} + u_{i,j-1,k} - 2u_{ijk} & i = n_x - 1, \quad j = n_y - 1 \end{cases} \\ u_{ij,k+1} = v_{ijk}\Delta t + \frac{(\Delta t)^2}{2} a_{ijk} \\ v_{ij,k+1} = v_{ijk} + a_{ijk}\Delta t \end{cases}. \tag{4}$$

最后将方程进行多次迭代就能近似解出水面的波动形式.

2 与光线追踪算法结合

为了将这些点渲染出来,首先将这些点相互连接形成许多三角形,然后将其渲染出来就行了. 传统的光栅化算法当然十分高效,基本可以忽略三角形绘制所需的时间,但是这样得到的画质也是惨不忍睹的. 因此我们采用光线追踪算法来渲染.由于在CPU上构建BVH需要耗费大量时间,这与所需要的高性能是相悖的, 考虑到水面有一定的波动范围,因此在构建BVH的时候就将其z方向(振动方向)设为该阈值,这样就无需重复构造没有必要的包围盒.而三角形的预计算本来就是在GPU上以并行化的方式进行,基本不需要担心这一部分的计算速度. 而且可以判断哪些三角形面片是属于水面的,在三角形预计算的时候判断一下就能将静态的三角形剔除从而加速.