Research on 3D Simulation Method of Nearshore Storm Surge Based on FLIP

Qiuyan Wang1,*, Hao Du1
1College of Information Science and Engineering, Ocean University of China, Laoshan Campus, Qingdao, China, 266100

*Corresponding author e-mail: wangqiuyan@ouc.edu.cn

Abstract. This paper proposes a FLIP-based three-dimensional storm surge simulation method. Based on the Fluid Implicit Particle (FLIP) method, the flow field is calculated by FLIP fluid, and finally through the standard coloring based on PBR in the Unity engine. The device renders the fluid state model generated at each moment. The experimental results show that the method in this paper not only meets the realistic requirements of nearshore storm surge simulation, but also effectively improves the efficiency of scene rendering. The result can be used not only in game production and movie special effects, but also in engineering simulations such as ocean engineering and environmental engineering, and has a wide range of application prospects and application values.

Keywords: Storm Surge Simulation, Fluid Implicit Particle (FLIP), Unity Engine

1. Introduction
Storm surge is a common marine environmental disaster phenomenon, especially in the context of global climate anomalies. The frequent occurrence of storm surges has brought immeasurable losses to various coastal countries. Therefore, storm surges have become the world’s marine research field. The key research object. Since the 1950s, with the development of computer technology, the research on storm surge at home and abroad has changed from theoretical research to numerical simulation research, and a large number of numerical models have emerged, such as MIKE 21, Delft3D and FVCOM. However, the results of these numerical models are mostly expressed in two-dimensional charts and two-dimensional images based on geographic information systems, and because most of these numerical models are empirical statistical models, the simulation results of storm surges lack a sense of reality. Until the emergence and development of computer graphics, a new perspective was provided for the study of storm surge phenomena. Through global illumination simulation technology, various ocean scenes can be rendered in real time or offline, making the rendered scenes extremely realistic.

This paper uses the physics-based fluid simulation method to solve the flow field, and renders the simulation results through the Unity engine to achieve a more realistic and rapid simulation of the storm surge phenomenon in the coastal zone.
2. Related work
The physics-based fluid simulation method is based on the Navier-Stokes equation which is the fluid motion equation, and the partial differential term is numerically solved to obtain the fluid motion state information at each moment. Euler's method is one of physics-based fluid simulation methods based on grid, which is a method to study the physical properties of a fixed point in the flow field over time. Foster et al. used the finite difference method to solve the three-dimensional N-S equation for the first time, and applied the results to the field of graphics to achieve a realistic three-dimensional fluid simulation [1]. Enright et al. proposed a particle level set method to extract the free surface of a fluid [2]. Stam used the semi-Lagrangian method to solve the convection term in the N-S equation on the basis of Foster et al., and realized unconditionally stable numerical simulation in the case of large time steps [3]. The FLIP method proposed by Brackbill et al. is a combination of the MAC method and the PIC method, and has extremely low numerical dissipation [4]. Boyd et al. adjusted the number of phases of the FLIP method, and introduced multiple sets of level set equations to track the phase boundary, and realized the air-water two-phase flow simulation [5]. Ferstl et al. proposed a method, NBFLIP, to store labeled particles in a narrow area near the free surface, which not only greatly reduces the amount of calculation, but also maintains the original accuracy and details [6].

3. Fluid simulation based on FLIP

3.1. Marker particles
The initialization of the marker particles is achieved by corresponding settings in the initial fluid grid. The specific process is to divide the fluid grid into 8 sub-grids. Each sub-grid contains a marker particle. At the same time, in order to maintain the rendering details of the fluid, Randomly shake the center position of the particle and give the initial velocity.

The initialization of the velocity field is achieved through the transfer of particle velocity and the grid velocity at grid index (i, j, k) is:

\[
\begin{align*}
 u_{i+1/2,j,k} &= \sum_p u_p k(\hat{X}_p - \hat{X}_{i+1/2,j,k}) \\
 v_{i,j+1/2,k} &= \sum_p v_p k(\hat{X}_p - \hat{X}_{i,j+1/2,k}) \\
 w_{i,j,k+1/2} &= \sum_p w_p k(\hat{X}_p - \hat{X}_{i,j,k+1/2})
\end{align*}
\]

(1)

In equation 1, \( \hat{X}_p \) is particle coordinate, \( k \) is weighting function and the equations as:

\[
k(x, y, z) = h(\frac{x}{\Delta d})h(\frac{y}{\Delta d})h(\frac{z}{\Delta d})
\]

(2)

\[
h(r) = \begin{cases} 
1 - r & (0 \leq r \leq 1) \\
1 + r & (-1 \leq r < 0) \\
0 & \text{other}
\end{cases}
\]

(3)

In equation 2, \( \Delta d \) is the scale in the X, Y, and Z directions of each grid. The velocity update of the marker particles depends on the velocity field of the fluid grid. For the FLIP method, the interpolation result of adding the original particle velocity to the grid velocity change value is adopted, so the updated particle velocity \( \vec{v}_{update}^p \) is:
\[ V_{update}^{update} = V_p + \text{interp}(V_{grid}^{update} - V_{grid}, \bar{X}_p) \] \hspace{1cm} (4)

In equation 4, \( V_p \) is the original velocity of particle \( p \), \( V_{update}^{update} \) is the updated velocity field by projection, \( V_{grid}^{grid} \) is the original velocity field, \( \bar{X}_p \) is the coordinate of particle \( p \) and interp is interpolation function.

3.2. Smooth surface

The idea of smooth surface construction is to polygonize the scalar field by level set method proposed by Zhu et al. [16] and matching cubes method proposed by Lorensen et al. [17].

The level set of point \( \bar{X} \) to the particle set of radius \( r \) is:

\[ \Phi(\bar{X}) = \left| \bar{X} - \bar{X} \right| - \bar{r} \] \hspace{1cm} (5)

\[ \bar{X} = \frac{\sum_{i} k\left( \frac{\left| \bar{X} - \bar{X}_i \right|}{h} \right) \bar{X}_i}{\sum_{i} k\left( \frac{\left| \bar{X} - \bar{X}_i \right|}{h} \right)} \] \hspace{1cm} (6)

\[ \bar{r} = \frac{\sum_{i} k\left( \frac{\left| \bar{X} - \bar{X}_i \right|}{h} \right) r_i}{\sum_{i} k\left( \frac{\left| \bar{X} - \bar{X}_i \right|}{h} \right)} \] \hspace{1cm} (7)

In the equations, \( \bar{X} \) is the weighted average of the coordinates of the surrounding particles, and \( \bar{r} \) is the weighted average of the radius of the surrounding particles. \( h \) is the range of each particle and is double the particle radius. \( k \) is smooth kernel function.

After advancing the velocity field step at each moment, the value in each fluid grid will change. Due to the particularity of grid construction, the value in the grid is the value represented by the vertex corresponding to the smallest coordinate in the grid. In order to filter out the fluid grids on the surface, it is necessary to compare this value with a preset grid threshold. For each surface grid, perform the steps as followed:

①Find the 7 grids around the current grid, their order and number are as shown in figure 1. In the above figure (a), the blocked mesh is the current mesh, numbered 0; (b) is the corresponding vertex number of the current mesh. The purpose of this step is to obtain the 8 vertices of the current mesh, and to map the 8 vertices and the mesh to an 8-bit binary variable Cubeindex. If there is a grid value greater than the grid threshold, the vertex corresponding to the grid is outside the threshold and the corresponding bit is set to 1, otherwise it is 0.

![Figure 1. The current grid and the 7 adjacent grids](image-url)
Initialize a lookup table Edgetable containing 256 elements and a triangle grid index table Triangletable. Each element in the Edgetable returns a 12-bit binary variable Vertexindex, which corresponds to the 12 edges of the current grid.

Substitute the calculated Cubeindex into Edgetable to locate the corresponding element. If a bit of the returned Vertexindex is 1, the corresponding edge will intersect the level set, otherwise it will not intersect; and the intersection point of the corresponding edge and the level set will be calculated to obtain the initial part The level set grid faces.

Substitute the calculated Cubeindex into the Triangletable to locate the corresponding triangle grid index to refine the initial local level set grid surface.

The local level set grid surface generated by each surface grid is integrated to obtain the approximate grid surface of the level set, which is the desired smooth surface.

3.3. Projection

In order to obtain the non-dispersive velocity field that satisfies the boundary conditions at the nth time, the velocity and pressure distribution at the solid boundary are set as MAC staggered grid mode, and the pressure of the free surface is 0. The projection at time step n is:

$$\frac{\Delta t}{\rho} \nabla P \cdot \hat{n} = (\overrightarrow{V}^n - \overrightarrow{V}_{solid}^n) \cdot \hat{n}$$

In equation 9, \(\overrightarrow{V}^n\) is the velocity field at time step n; \(\overrightarrow{V}_{solid}^n\) is the velocity at the solid boundary at the current time. Discretize \(\nabla P\) using forward difference method as follows:

$$\begin{align*}
    u_{n+1,i+1/2,j,k}^n &= u_{i+1/2,j,k}^n - \Delta t \frac{1}{\rho} \frac{P_{i+1,j,k} - P_{i,j,k}}{\Delta x} \\
    v_{n+1,i,j+1/2,k}^n &= v_{i,j+1/2,k}^n - \Delta t \frac{1}{\rho} \frac{P_{i,j+1,k} - P_{i,j,k}}{\Delta y} \\
    w_{n+1,i,j,k+1/2}^n &= w_{i,j,k+1/2}^n - \Delta t \frac{1}{\rho} \frac{P_{i,j,k+1} - P_{i,j,k}}{\Delta z}
\end{align*}$$

The incompressible condition is discretized by the backward difference method and brought back to equation (10) to obtain equation (11):

$$\begin{align*}
    \frac{u_{n+1,i+1/2,j,k}^n - u_{n+1,i-1/2,j,k}^n}{\Delta x} + \frac{v_{n+1,i,j+1/2,k}^n - v_{n+1,i,j-1/2,k}^n}{\Delta y} + \\
    \frac{w_{n+1,i,j,k+1/2}^n - w_{n+1,i,j,k-1/2}^n}{\Delta z} + \frac{2P_{t,j,k} - P_{t,j,k-1} - P_{t,j,k+1}}{(\Delta t)^2} = 0
\end{align*}$$

Equation (11) can also be expressed as:

$$\nabla \cdot \overrightarrow{V} = \frac{\Delta t}{\rho} (\nabla^2 P)$$

It is the Poisson equation of pressure. In order to solve the pressure field corresponding to the flow field, the Poisson equation is now linearly matrixed: Suppose there is a fluid grid in the flow field, and for each fluid grid formula (11), and It is stipulated that the coefficients of non-adjacent terms are set to 0, so a linear equation system can be obtained. b is the constant term of the equation system, P is the unknown term, and A is the positive definite symmetric coefficient matrix, so it is suitable for solving by the conjugate gradient method. In this paper, the Modified Incomplete Cholesky Decomposition -
Preconditioned Conjugate Gradient (MIC-PCG) is used to solve the system of equations. The algorithm of MIC-PCG is as followed:

① Initialize:

\[
\begin{align*}
P_0 &= 0 \\
r_0 &= b - AP_0 \\
z_0 &= (LL^T)^{-1}r_0 \\
s_0 &= z_0
\end{align*}
\]

(12)

S is the iterative direction vector, P is the pressure term to be solved, r is the residual, Z is the auxiliary vector, L is the approximate decomposition lower triangular matrix of the coefficient matrix A, its expression is \(L = FE^{-1} + E\), F is the strict lower triangular matrix of A, and the diagonal matrix E can be calculated by the following equation:

\[
E_{i,j,k} = \left( \frac{A_{i,j,k}}{E_{i,j,k}} \right)^2 - \left( \frac{A_{i,j\pm1,k}}{E_{i,j\pm1,k}} \right)^2 - \left( \frac{A_{i\pm1,j,k}}{E_{i\pm1,j,k}} \right)^2
\]

(13)

② For every iteration step:

\[
\alpha = \frac{z_i^T r_i}{s_i^T A s_i}
\]

\[
P_{i+1} = P_i + \alpha s_i
\]

\[
r_{i+1} = r_i - \alpha z_i
\]

\[
z_{i+1} = (LL^T)^{-1}r_{i+1}
\]

\[
s_{i+1} = z_{i+1} + \beta s_i
\]

\[
\beta = \frac{z_i^T r_{i+1}}{z_i^T r_i}
\]

(14)

\(\alpha\) is the iteration step.

③ If the calculated convergence tolerance \(|I|\) is less than the set convergence tolerance, the loop ends.

3.4. Advection

In FLIP method, the particle method is mainly used to solve the advection equation, that is, the velocity field at the next time step is obtained by advancing the fluid particles. Suppose the velocity of a certain fluid particle at the current time step is \(\vec{V}_r\), and the coordinate is \(\vec{X}_r\) and its velocity at the next time step is \(\vec{V}_r'\), and the coordinate is \(\vec{X}_r'\). From the advection equation \(D\vec{V}/Dt = 0\), the rate of change of the flow velocity \(\vec{V}\) with time is 0, so the flow velocity is \(\vec{V}_r = \vec{V}_r'\) before and after the time step \(\Delta t\). Therefore, it is only necessary to calculate the position \(\vec{X}_r\) and velocity \(\vec{V}_r\) of the particle forward. In order to maintain the stability of the time integration format and higher-order numerical accuracy, this paper adopts the fourth-order Runge-Kutta method to solve the problem. The equation is as followed:
\[
\hat{X}_\rho = \hat{X}_\rho - \frac{1}{6} \Delta k_1 - \frac{1}{3} \Delta k_2 - \frac{1}{3} \Delta k_3 - \frac{1}{6} \Delta k_4
\]

\[
\begin{align*}
  k_1 &= V(\hat{X}_\rho) \\
  k_2 &= V(\hat{X}_\rho + \frac{1}{2} \Delta k_1) \\
  k_3 &= V(\hat{X}_\rho + \frac{1}{2} \Delta k_2) \\
  k_4 &= V(\hat{X}_\rho + \Delta k_3)
\end{align*}
\]

(15)

3.5. Algorithm flow

① Initialize the sea area, impact model and flow field attributes. Set the scope of the flow field and the grid scale; convert the impact model into a solid boundary in the watershed; set the initial velocity, volume force, time step, etc. of the fluid; and specify the center coordinate point of each grid as the calculated point coordinates of the grid.

② Calculation of flow field advancement. For each time step, loop the following steps:

A. Smooth surface construction. Calculate the level set of the current fluid state according to equation (5)~(8), and then obtain a smooth surface similar to the level set according to the Cubematch algorithm, and store it in the .ply format.

B. Initialize the velocity field. The marker particle velocity is transferred to the corresponding flow field grid through equation (1), and the initialized velocity field \( \hat{V}_r \) is obtained.

C. Make the velocity field no divergent. The preconditioning conjugate gradient method MIC-PCG based on the improved Cholesky decomposition is used to solve the Poisson equation for pressure, and the calculated pressure field is substituted into the projection equation to obtain the dispersion-free velocity field \( \hat{V}_{adv} \).

D. Update the speed of the marker particles. Use equation (4) to get the marker particle velocity in the non-dispersive velocity field \( \hat{V}_{update} \).

E. Advance the velocity field. The velocity of each marker particle at the next moment is calculated by the fourth-order Runge-Kutta method (equations (16) ~ (17)).

4. Result

In order to obtain higher rendering efficiency and rendering effects, this paper selects the built-in rendering pipeline of the Unity engine to render the generated fluid state model and the results are as followed:
Figure 2. The rendering effect of the storm surge movement process

The figures (a) ~ (d) show the movement state of storm surge from generation to development to extinction. It can be seen from the figure that the movement process of the storm surge is very smooth, and the details of its fragmentation can be drawn perfectly, indicating that the FLIP fluid simulation method used in this paper has obtained the correct simulation results. However, the shape of the broken part is slightly sharp, indicating that the selected flow field grid scale is not fine enough.

5. Conclusion

This paper mainly realizes the three-dimensional simulation of offshore storm surge based on the grid-based fluid implicit particle method FLIP. The experimental results show that the 3D storm surge simulation method proposed in this paper not only meets the realistic requirements of nearshore storm surge simulation, but also effectively improves the efficiency of scene rendering and rendering. The result can be used not only in game production and movie special effects, but also in engineering simulations such as ocean engineering and environmental engineering, and has a wide range of application prospects and application values.

Consider in future work: 1) Use Nvidia Optix real-time rendering platform to perform simulation rendering to ensure simulation efficiency and rendering efficiency; 2) Use volume rendering to render each fluid mesh to achieve underwater godray and underwater Lighting effects such as caustics.

References
[1] Foster N, Metaxas D N. Realistic Animation of Liquids [J]. Graphical Models & Image Processing, 1995, 58: 471-483.
[2] Enright D, Fedkiw S M R. Animation and Rendering of Complex Water Surfaces [J]. ACM TransactioN-S on Graphics, 2002, 21 (3): p. 736-744.
[3] Stam J. Stable Fluids [J]. Acm TransactioN-S on Graphics, 1999, 1999.
[4] Brackbill J U, Kothe D B, Ruppel H M. Flip: A low-dissipation, particle-in-cell method for fluid flow [J]. Computer Physics CommunicatioN-S, 1988, 48 (1): 25-38.
[5] Landon, Boyd, Robert, et al. MultiFLIP for energetic two-phase fluid simulation [J]. Acm
TraN-SactioN-S on Graphics, 2012.

[6] Ferstl F, Ando R, Wojtan C, et al. Narrow Band FLIP for Liquid SimulatioN-S [J]. Computer Graphics Forum, 2016.