Terrain Modificaiton with fluid simulation using doubly-coupled PCISPH

JIWEI WANG CHENGHUA YANG YIHANG ZHU

ACM Reference Format:
Jiwei Wang Chenghua Yang Yihang Zhu. 2021. Terrain Modificaiton with fluid simulation using doubly-coupled PCISPH. 1, 1 (June 2021), 1 page. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In this project, we make an attempt to realize erosion phenomenon applyed on the terrain modification, especially on the mountain peak formation in the plate movement. Detailedly, we use PCISPH to simulate fluid behavior. We utilize marching cube technique to generate mesh by passing volume data. The coupling, including single-sided solid coupling and double coupling, is implemented to simulate the fluid behavior and terrain change in the erosion process.

2 IMPLEMENTATION DETAILS

2.1 PCISPH Fluid Simulation

PCISPH, known as Predictive–Corrective Incompressible SPH, is an incompressible fluid simulation method based on SPH model. This method first finds neighbours for each particles and initialize pressure and pressure force accelaration to 0, as well as calculating the force accelaration including gravity and viscosity accelaration. Then for a given interation number, we repeatedly perform prediction, correction and pressure accelaration compution.
In detail, in each iteration, we first predict the velocity and position of each particle at time $t + 1$ according to the velocity and position at time t,

$$pre\_v_{t+1} = v_t + (a_t^{ext} + a_t^{p}) * dt$$
$$pre\_p_{t+1} = p_t + pre\_v_{t+1} * dt$$

It is followed that we predict the density $\rho_i^*(t+1)$ of each particle, the postion varation $\rho_{err}^*(t+1)$. We then can update the pressure $p_i(t)$ by add $f(\rho_{err}^*(t+1))$. The $f()$ is the smooth kernel function that satisfying the following properties: f returns 0 if input r is larger that the kernel radius h and f reduces to delta function when kernel radius h goes to 0.
After the interation, we finally compute the new velocity $v_i(t+1)$ and new position $p_i(t+1)$ with the intered pressure and density. These two data can be applied into simulation in the following *simulation*. For the range search part to find neighbours of each particle, we use kd tree structure to speed up query with endurable building cost.

2.2 Particle Rendering

During the fluid simulation, we apply particle rendering to show the spatial shape of the entire fluid. In the implementation, we utilize the unity compute shader by passing the particle position, which runs on GPU, to get the rendered result in the scene.

2.3 Marching Cube Rendering

We use a data structure to store volume data, which is called volume matrix. The volume matrix store a $Vector3Int$ variable $size$ to record the size of the entire coordinate space, and a $float$ array $data$ to store the volulme value.

2.4 Terrain Generation With Noise

We apply 3 dimensional Perlin Noise to generate a volume data at each spatial coordinate in the bounding box we set. It is naturally that we package the volume data into the volume matrix in the purpose of building the terrain. In the practice, we use noise layers to generate different terrain by changing the weight of each layer in the entire part. We also set parameters $sharpness$ and $magnitude$ to restrain the spatial distribution of the terrain.
We generate the perlin noise of each spatial coordinate in the computer shader at differrent threads, which runs on the GPU.

2.5 Terrain Coloring

We designed a color curve to approximately simulate the surface appearance of real terrain. The color is mapped according to the height and normal direction. In general, the larger the height and the closer the normal direction with the positive y-axis direciton, the larger the value will be in the color curve. As a result, we can generate a terrian with river valley, meadow in the midium to height altitude and the ice sheet on the peak.

3 RESULTS

4 CONCLUSION

Author's address: Jiwei Wang Chenghua Yang Yihang Zhu.

XXXX-XXXX/2021/6-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn