Water Wave Particles with Simple Boundaries in Unity3D

Project Report

Keren He
McGill University
ke.he@mail.mcgill.ca
Supervisor: Paul Kry

November 22, 2018

Abstract

In this project report we demonstrate water surface simulation in Unity3D. We adapt the water particle method proposed by Yuksel, House and Keyser (2007). This method involves height field generation which allows water surface to be simulated in real-time with comparatively low computation cost. We show that this method can be implemented in Unity3D, and can effectively simulate the propagation and reflection behavior of water waves with computational efficiency even in large scale water surface scenarios. Through this study, we can conclude that the water particle method can be implemented in other game engines and can be applied to the game industry without burdening the overall simulation speed.

1 Introduction

Water is one of the most beautiful and important natural phenomena. Throughout the years scientists try to model and simulate water offline and in real-time. Water, like all other fluids, can be modeled by Navior-Stokes equations. However such equations are hard to solve. In the game industry, water is often treated as environmental elements. This is because large scale water model that can interact with objects is computational complex, yet all the simulations are often required to be computed in real time. Although particle based water simulation can handle interactions pretty well, when it comes to large scaled water scene the computation cost increase rapidly. However, this can be simplified since game water simulation is more focused on the water surface, what happened under the surface is usually ignored. Height field generation can play an important role in dealing with this situation. In this project, we present a water particle system introduced by Yuksel, House and Keyser along with a height field formula which can model large-scaled water scene in real time efficiently [14]. In the paper, water interaction with object simulation can also be solved by the particle system with other techniques, more details can be found [14]. We implement the wave particle system with simple boundary conditions in Unity3D without interaction with objects. With this project,the goal is to build intuition about water surface simulation in games for beginners by implementing water particle system on CPU in Unity3D.

In Section 2 we present some related works. Then we will brief explain the wave particle system in section 3. Later in Section 4, implementation details are presented. Section 5 shows our results achieved using our implementation method. In Section 6, we conclude the project and discuss about possible future work directions.

2 Related Work

In this section we present some existing literature on water simulation. Two major approaches, Lagrange viewpoint (also known as particle based simulation) and Eulerian viewpoint (also known as grid based simulation), are commonly used to simulate water from various aspects. Both approaches involve solving a version of incompressible Navier-Stokes equations (NSE) that model the various fluid phenomena. Comparing to particle based approach, grid based simulation is better at tracking smooth surface but is also relatively slower and suffers from mass loss. The particle based simulation treats fluid as small particles. This makes it easier to account for the conservation of mass. However, it will not look as smooth as grid

based approach due to the issues with a smooth surface [2]. Bridson released the book Fluid Simulation for computer graphics in 2008 [3] which presents a complete grid based method to simulate fluid such as water and gas. In 1999, Stam published stable fluids [10] and introduced the idea of semi-Lagrange where we can advect the velocity for each grid using particles. He continued to publish Real-time Fluid Dynamics for Games [11] in 2003. It is a simpler version of his paper in 1999, where he presented a simple and rapid simulation of 2D fluid for the game engine.

Smoothed-particle hydrodynamics (SPH) [6] was first introduced in 1977 for simulation of stellar. Later in 2003, Müller, Charypar and Gross [9] presented a method that obtains an extension on SPH. In this method, the authors were able to efficiently simulate the fluid system with a free surface in order to model the interaction of fluid and other objects while ignoring mass conservation and conservation in NSE.

Several hybrid approaches attempts to combine the Eulerian and Lagrange viewpoints together to achieve a better simulation effect. Particle level sets were introduced in 2001, where a general, efficient and flexible approach was presented to model a fluid system interacting with objects while maintaining a smooth free surface. Lagrangine view is used to simulate the fluid movement for the purpose of mass conservation, while on the other hand, Eularian view is used to generate the fluid surface to achieve smoothness. Particle-in-cell (PIC) method [5] was first presented in 1957 to solve hydrodynamic problems. This method also involves a combination of Lagrangine and Eularian view, where the fluid space is divided into multiple grids, and particles carrying mass were created to present the fluid within each grid cell. Later in 1986, an improved version of PIC which is called fluid implicit particle (FLIP) was presented to model a large scale of fluid by simulating particles in zoned gird [1]. Another important approach to model fluid is using heightfields.

Heightfields are often used to simulate large-scaled fluids in real-time. In this method, the z-coordinates of the fluid is a function of x, y coordinates. With this approach, 3D simulation of fluids is downsized to a 2D simulation of water columns, accelerating the speed of simulation. Comparing to the full three dimensional model of fluids, Heightfields method loses some natural fluid phenomena, such as ignoring the vertical velocity of the fluid or use the assumption that vertical velocity is only affected by the gravity and pressure. Shallow water equation (SWE) is a common approximation to build heightfields [8]. In 1990, Kass and Miller [7] presented a method based on SWE to model heightfields where the wave velocity is proportional to the square root of fluid depth. In their implementation, refraction, reflection and boundary condition are well handled. Later in 2006, a hybrid simulation method

was presented by Thurey [13]. The authors couple a two dimensional SWE simulation with a full three dimensional free surface simulation to achieve a water drop propagating to a large wave.

In many current games, the fluid volume underneath the surface is neglected. Simulating realistic waves, especially when interacting with floating objects is very important in many modern computer games. To accomplish this, some methods which assume that no terrain exists under the water surfaces were introduced. In 2001, a method of synthesizing a patch of ocean waves from a fast Fourier transform (FFT) prescription was presented by Tessendorf [12]. The height of the ocean wave was produced by f sinusoidal functions. Later in 2007, a new method called wave particles [14] was introduced to model water waves and their interactions with floating objects in both ocean views and bounded environments. Yuksel, House and Keyser presented how they convert wave particles to a height field surface. The detailed methodology will be discussed in the following section as our project is implemented based on this paper.

3 Methodology

In this section we will present the methodology used in this project which is fully based on water wave particles [14]. A discrete analytically solution to the wave equation was provided by the paper. The wave equation is shown as

$$\nabla^2 Z = \frac{1}{v^2}\frac{\partial^2 Z}{\partial t^2} \tag{1}$$

where Z is the height function of the wave. The value of this function varies according to the surface point location and time. This function shows that the shape of the wave determines the change of vertical deviation. According to the principle of superposition of waves, the sum of two valid solutions of the wave equation also satisfies the wave equation. Hence we can simulate different groups of water wave independently. We can reform the wave equation to

$$\frac{\partial^2 Z}{\partial x^2} + \frac{\partial^2 Z}{\partial y^2} = \frac{1}{v^2}\frac{\partial^2 Z}{\partial t^2} \tag{2}$$

to fit the three dimension water surface wave model. 3D water waves can propagate in many directions as they travel in the X, Y plane, precisely infinite many directions thus it is hard to track the exact propagation direction of a surface wave in 3D.

Yuksel, House and Keyser introduced the concept of radial wave particle to solve

this problem. A simple wave is represented by a collection of wave particles on the wave surface propagating with wave velocity. The height displacement of a point on the surface can be modeled by the summation of local deviation functions and each wave particle has a local deviation function. The height function allows the 3D wave simulation to be represented by a 2D particle system thus reducing the complexity of the simulation process. The formulas are

$$Z(x,t) = z_0 + \eta_z(x,t) \qquad \text{and,} \tag{3}$$

$$\eta_z(x,t) = \sum_i D_i(x,t) \tag{4}$$

where z_0 is the ground level of the water surface and η_z is total deviation caused by waves. The function D_i presents the local deviation of each particle. The local deviation function for particle i is formulated as

$$D_i(x,t) = a_i W_i(x - x_i(t)) \tag{5}$$

where a_i is the amplitude of particle i, W_i a constant waveform function and $x_i(t)$ the particles position at time t.

Yuksel, House and Keyser choose the waveform function to be

$$W_i(u) = \frac{1}{2}\left(\cos\left(\frac{\pi|x - x_i(t)|}{r_i}\right) + 1\right)\Pi\left(\frac{|x - x_i(t)|}{2r_i}\right) \tag{6}$$

where r_i is the radius of the wave particle. This function satisfies many requirements for simulating water waves. Sinusoidal functions match the shape of water wave and the above formula is a solution to the wave equation in 2D. Moreover, Equation 6 is non-zero over a finite area and at the end points, the first derivative is zero. Using this equation is also easy to create continuous waves with alternating amplitudes. Another advantage of using a Radial definition of waves particles is that expanding wave simulation becomes easier. Since r_i is constant for each particles, this means that the width of each particle remains the same. In order to preserve the energy that each particle carries, the amplitude of each particle has to stay constant. With a constant amplitude, each particle is able to preserve its shape as it travels. As a matter of fact, when a wave front is expanding, due to conservation of energy, the amplitude of the wave front would decrease. Wave particle systems can achieve this property without any further adjustment. As a wave particle propagates away from the origin, the distance between the neighbouring particles becomes larger, hence the total amplitude of the wave decreases. However, Yuksel, House and Keyser showed

that the error of amplitude and shape of the wave front is bounded by 0.8% and 7.1% respectively if the distances between the neighbouring particles are kept less than half of the particle radius. Hence, each particle is subdivided into three particles once the distances between neighbouring particles reach half radius. This effectively keeps the distances between neighbouring particles always below half radius, and also controls the errors of amplitude and shape of simulated wave front to be always below 0.8% and 7.1% respectively as a result.

4 Implementation Details

In this section we will present in detail about the simulation of the water wave. The implementation steps are listed below.

```
StartUp ()
   Water Environment Creation ()
   Filter Value Calculation ()
FixedUpdate ()
   Particle Iteration ()
   Height Field Generation ()
```

4.1 Unity setups and some Assumptions

Each `FixedUpdate()` iteration is 0.05 seconds in order to best present the wave front propagation. There is no damping in the system. When the particle has amplitude less then 0.001 unit we kill the particle.

4.2 Water environment creation

The surface of the swimming pool is generated by Unity triangular mesh API, so we dont have to calculate the normal of each face on the mesh surface by ourself. For a simple rectangle swimming pool of size 50 by 50, we can create a mesh of size 201 by 201 in both X and Y direction and each neighboring vertices are 0.25 unit apart. This method simulates a less accurate shape of the wave but is computational efficient. For more accurate wave shapes, we need to combine several surface mesh plans together due to the limitation of Unity. In this situation, the computation time for each iteration of fixed update increases. During this step, we also load in a text file which indicates the boundary vertices of a regular shaped swimming pool and these vertices are determined by MATLAB. According to the boundary

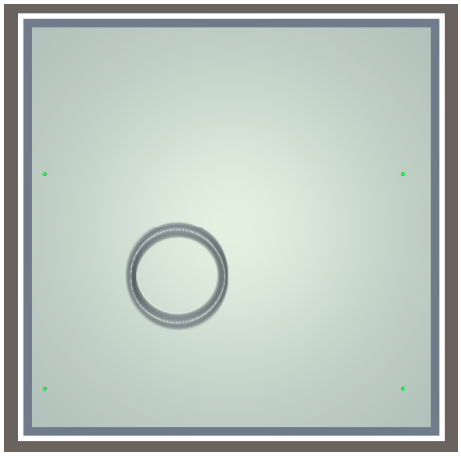


Figure 1: Image of a simple pool with size 50 unit by 50 unit

vertices, we can use Unity's 3D Gameobject cube to create the boundary wall of the swimming pool.

4.3 Initialize Water Wave Particles

For simplicity, water wave will be created when the mouse left key is clicked or while dragging. While mouse clicked, 12 wave particles will be initialized and form a circle shape around the origin. By saying create we do not mean create a 3D GameObject inside unity working space. We should generate virtual wave particles which contain the properties of this water wave. The mouse position is immediately the origin for these water wave particles. Particles are placed 1 unit away from the origin and amplitude is set to 4. To couple with this initial distance, the radii of the wave particles are set to 2 so that these particles would not subdivide immediately after creation and the radii will remain constant for each wave particle as the wave propagate. The wave is initially propagating away from the origin with a speed of 15 units per second. It is obvious that the curvature of the wave changes constantly, hence we introduce the dispersion angle α to describe the shape of a given wave [14]. Dispersion angle α is the angle between the two side edges of a curved wave particle. This angle does not change during the wave particle propagation. The relationship between the dispersion angle and the curvature is expressed as

$$\kappa = \frac{\alpha}{w} \tag{7}$$

where κ is the curvature and w is the width of the water particle. The value of the dispersion angle is stored in each water wave particle as a property.

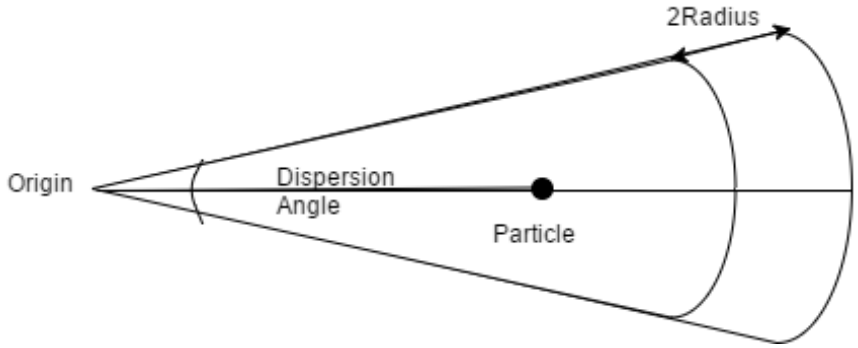


Figure 2: Image of dispersion angle

4.4 Particle Iteration

In this section we present how each water wave particle moves, subdivides and reflects from boundaries in the particle iteration step.

4.4.1 Water Particle Movement

While ignoring the objects that are interacting with the water, the wave particles move at a constant speed away from the origin. Without any interference, Given the starting point of a wave particle, x_0 at time t_0, and its origin, we are able to calculate its current position x at time t,

$$x + uv(t - t_0) \tag{8}$$

where v is the constant speed and u is the propagating direction of the wave particle. Because Unity updates the scene every 0.05 seconds, thus we assign the time interval between time t and $t + 1$ to be 0.05 seconds.

$$x_{t+1} = x_t + uv\Delta t \tag{9}$$

Each frame, every particle moves a distance of $uv\Delta t$ away from the origin without any interference.

4.4.2 Water Particle Subdivision

Particle subdivision occurs as the wave propagates away from the origin. This step is to make sure that when the wave expands there will be no cut off on the wave front.

Figure3 shows that without subdivision, wave front will be disconnected as the water wave particles advances and the distances between neighboring particles increase.

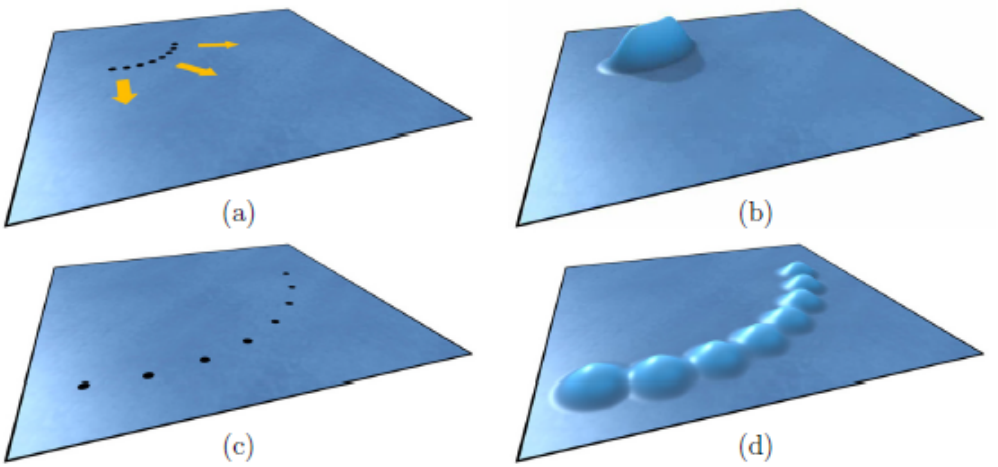


Figure 3: (a)(b)show a wave front (c)(d) show wave particles [14]

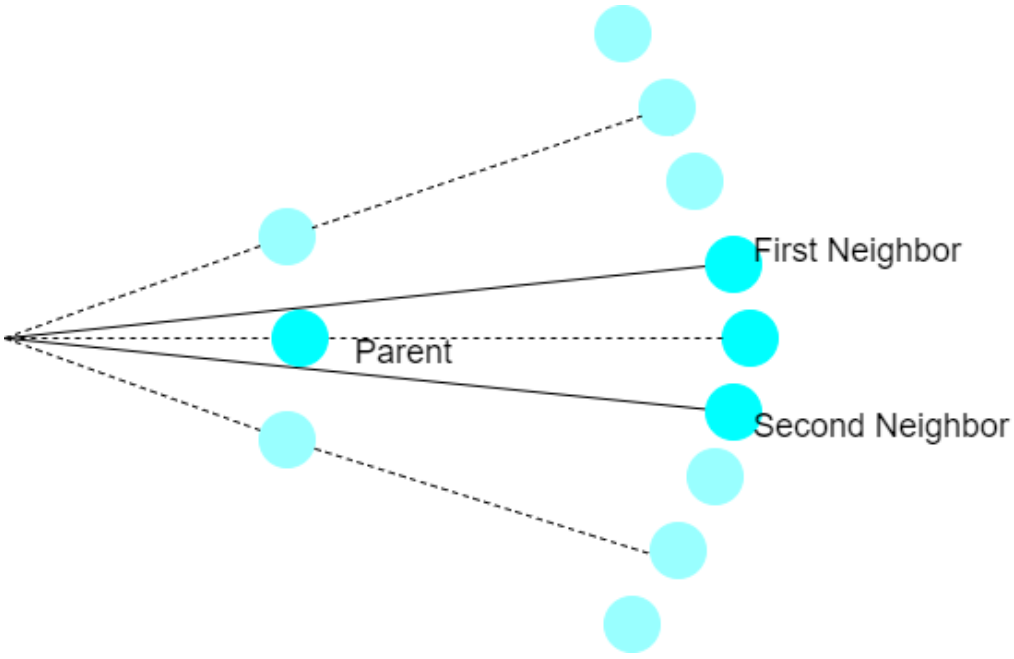


Figure 4: Example of wave particle subdivision

In order to keep the water wave front connected, we subdivide the particles when the distance between the two neighboring wave particles becomes greater than half of the particle radius. We assume that for a given wave particle, it shares the same origin with its two sides neighbours. Moreover, these three particles have equal dispersion angle. Under this assumption, it is straightforward to calculate the current distance d_t between the two neighboring wave particles. The calculation is based on the previous distance d_0, particle propagation speed, and its dispersion

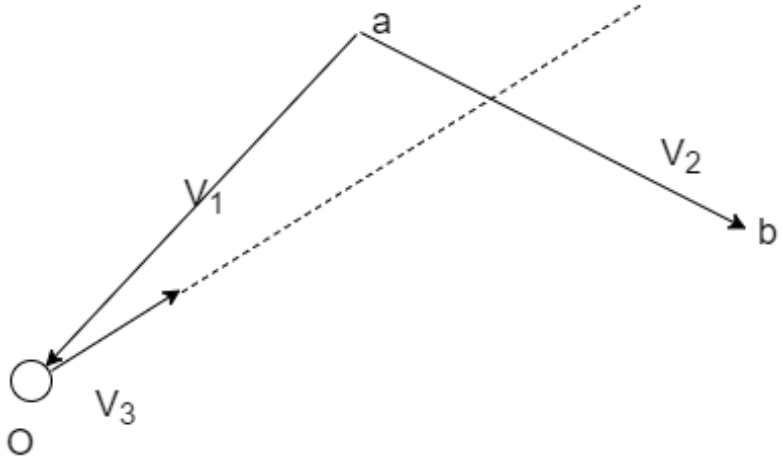


Figure 5: Ray intersects with line segment.

angle. The distance calculation function is as simple as

$$d_t = d_0 + \alpha v \left(t - t_0\right). \tag{10}$$

In order to build an efficient particle system, it is important to keep each particle independent. Therefore, instead of inserting a new particle between the two neighboring particles, we generate two child particles on the two sides of the parent particle during subdivision. These new children wave particles are placed $\frac{d_t}{3}$ away from the parent wave particle. Children and parent wave particles amplitude and dispersion angle becomes a third of the parent particle's amplitude and dispersion angle values before subdivision respectively.

4.4.3 Reflection

For a bounded water space, we can use ray line intersection functions to detect the collision between the wave particles and boundary line segments. Let the end points of a line segment be point a and point b, and the current position of a wave particle be O. We can identify the vector from a to O to be V_1, the vector from a to b be V_2 and the velocity vector of the wave particle to be V_3 shown in Figure 5. If the value of

$$\frac{v_1 \cdot v_3}{v_2 \cdot v_3} \tag{11}$$

is between 0 and 1 and the value of

$$\frac{|v_1 \times v_3|}{v_2 \cdot v_3} \tag{12}$$

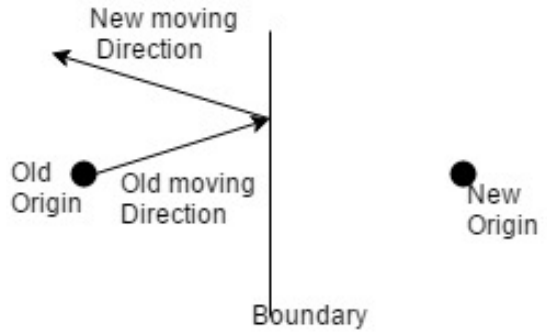


Figure 6: Origin change after reflection.

is greater or equal to 1, the water wave particle will hit the boundary. Additionally it indicates where on the boundary segment the particle will hit. Once the wave particle intersects the boundary, we instantly reflect it. At the reflection point, the particle's new origin is assigned to be the reflection point of the current origin against the boundary segment which the particle will collide with, shown in Figure 6. All the properties of the wave particle will be reassigned, such as dispersion angle and propagating direction, according to the new origin as presented in Section 4.3.

4.5 Height Field Updating

Height field map contains the vertical deformation of the water waves. The vertical deformation over here represent just the amplitude of the water wave. In this updating step we need to convert this deformation from the wave particles to the vertices of water surface mesh. We first need to render each wave particle on the height field while applying some interpolation.A sample picture is show in Figure 7 of show how we determine the exact position of each particle on the water surface mesh space and distribute its amplitude to the four nearest vertices of the surface mesh that are around the particle. In this step, vertices of surface mesh may receive multiple amplitudes from different wave particles, and we can directly add the amplitudes up. The vertices on the surface mesh whose amplitude is higher than 0 are called wave particle points. Next, we need to apply a filter so that wave particles points form a water wave shape. The vertical filter function is

$$d_z(p) = \frac{1}{2}\left(\cos\left(\frac{\pi|p|}{r}\right) + 1\right)\Pi\left(\frac{|p|}{2r}\right) \tag{13}$$

where $|P|$ is the distance between the surface mesh vertex and the computed wave particle point. The above 2D filter smooth out the amplitude in both X and Y direction of the surface. We can approximate this 2D filter function as a tensor

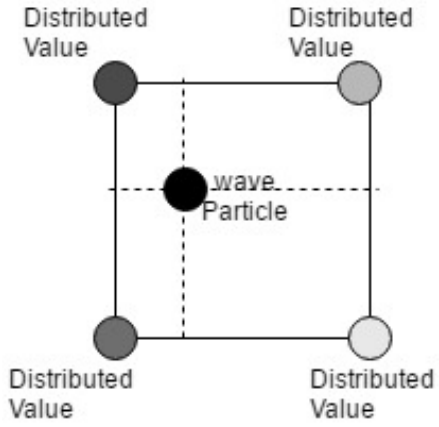
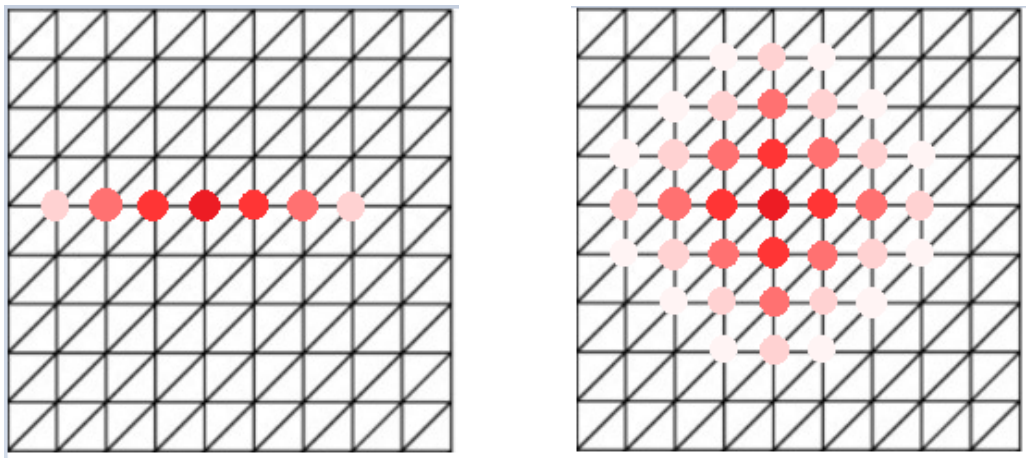


Figure 7: Interpolation of a wave particle



(a) Filtered by X direction filter

(b) Image (a) Filtered by Y direction filter

Figure 8: Example of a single wave particle point after filtering. The most red colored dot which is located at the center is the original wave particle point.

product of two 1D functions. The two 1D functions are

$$d_z(x) = \frac{1}{2}\left(\cos\left(\frac{\pi x}{r}\right) + 1\right)\Pi\left(\frac{x}{2r}\right) \tag{14}$$

$$d_z(y) = \frac{1}{2}\left(\cos\left(\frac{\pi y}{r}\right) + 1\right)\Pi\left(\frac{y}{2r}\right) \tag{15}$$

where x, y is the position of each mesh vertex. In this case we reduce the computation cost [4]. Rectangle function Π works as a cut off function that determines whether a vertex gets its amplitude induced by the neighboring wave particle points. For faster computation, only the vertex with distance to a particular wave particle point that is within two times the radius will be distributed some portion of the wave particle

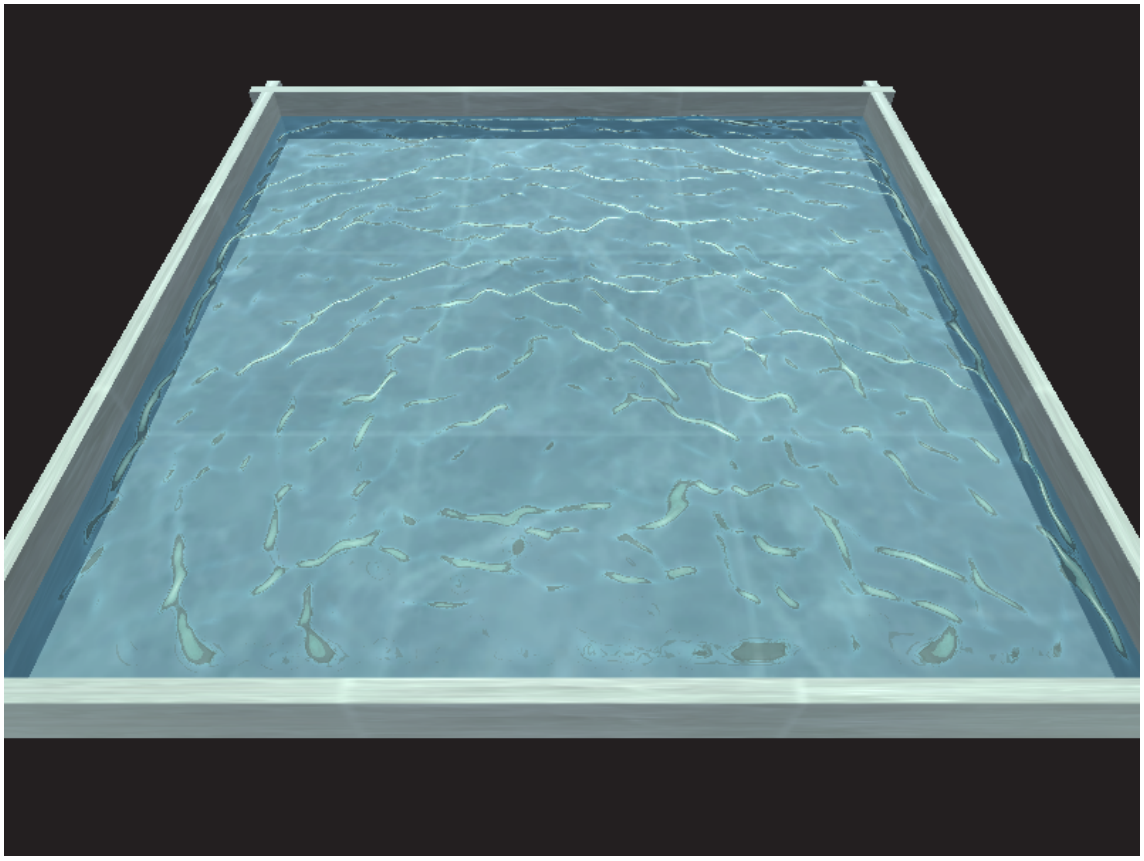
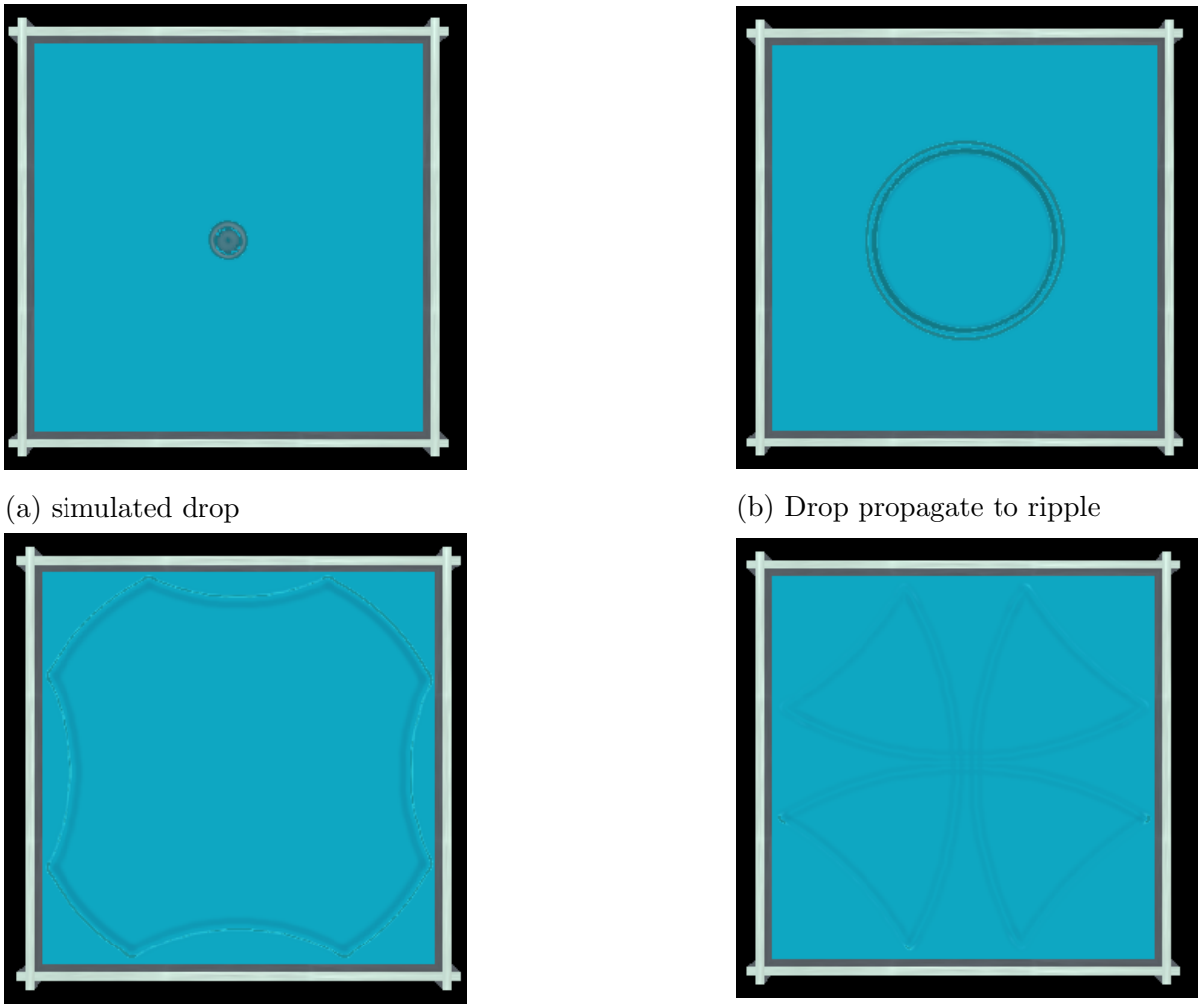


Figure 9: Swimming pool test scene

points amplitude. The exact portion a vertex gets from an existing wave particle point p depends on the distance between this vertex and p on a cosine function. This filter value is calculated in the start up step since the distance between two neighboring mesh vertices is the same across the water surface in both X and Y direction.We apply the filter function in X direction to each vertex, then we apply the filter function in Y direction to each vertex for each column and a sample result is given in Figure 8 for a single wave particle point.

5 Results

Unity reflection prob which acts as a camera that captures surrounding views in all direction and transparent material are used to improve the visualization of the simulated scene. All the computations except for mesh rendering are run on a normal CPU. Figure 9 shows a test scene of a swimming pool with resolution of 800 by 600.The mesh size of the pool surface is 200 by 200 and around 20000 active water wave particles exist in the figure scene. Figure 10 shows how waves once generated from a small drop by left mouse click expands to ripple. Once the waves hit the



(a) simulated drop

(b) Drop propagate to ripple

(c) Ripple reflect from boundaries

(d) Ripple after reflection

Figure 10: Example of a single drop expanding and reflected from boundaries.

boundary, they reflect from the boundary with same speed but opposite direction.

To test the speed of our project, we run swimming pool scene with different counts of active water wave particles on a surface mesh of size 200x200 and recorded data is shown in Table 1.Notice that the simulation time of particle iteration is linearly dependent on the number of active water wave particles. On the other hand the dependency of height field generation time and active particle size is less obvious. Other than the interpolation step which loops over all the active wave particles in the scene, height field generation loops over all the vertices on the mesh. Therefore, if the particle size is greatly smaller than the vertex size, the height field generation time should not vary a lot. It is clear that increasing the mesh size also increases the time of simulation. Table 2 shows the simulation time for mesh size of 400 by 400.

Particle Count	Particle Iteration Time in (ms)	Height Fields Generation Time in (ms)	Total Simulation Time in (ms)	Simulation Frame Per Seconds
5724	7.18	30.48	37.61	21.80
21083	26.80	44.95	71.75	13.20
50265	62.08	62.72	124.80	7.10
101059	130.79	102.58	233.37	4.40

Table 1: Performance data on 200 by 200 mesh

Particle Count	Particle Iteration Time in (ms)	Height Fields Generation Time in (ms)	Total Simulation Time in (ms)	Simulation Frame Per Seconds
5940	11.55	200.34	211.89	4.1
20988	29.48	217.09	246.57	3.9
51175	68.80	241.05	309.85	3.4
100283	131.86	286.73	418.59	2.4

Table 2: Performance data on 400 by 400 mesh

6 Conclusion

We briefly present the wave particle system which is invented by Yuksel, House and Keyser [14] and show how to use this system to simulate water waves in Unity3D. The simulation of water surface interacting with objects is not handled in this project.

Further work could be done to this project to add more effect and to optimize the computation speed. We can generate the height field by using texture mapping to accelerate the process. In this case, the computation will be assembled on a GPU instead of a CPU. Other techniques can also be applied to speed up the simulation time as described in the paper Wave particle [14]. We can calculate the subdividing and reflection occurrence time of each particle instantly after its creation and put this information into a linked list table. We can then update the status of each particle according to the linked list, and perform the previous step again after changing its status. In this way, we avoid the repetitive checks of whether a particle needs to reflect or subdivide in each iteration, thus reducing the cost of computation.

References

[1] Brackbill, J., and Ruppel, H. Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational physics 65*, 2 (1986), 314–343.

[2] Braley, C., and Sandu, A. Fluid simulation for computer graphics: A tutorial in grid based and particle based methods. *Virginia Tech, Blacksburg* (2010).

[3] Bridson, R. *Fluid simulation for computer graphics*. CRC Press, 2008.

[4] Darkness, K. Definition and properties of tensor products. https://www.uio.no/studier/emner/matnat/math/nedlagte-emner/MAT-INF2360/v12/tensortheory.pdf., 2012.

[5] Evans, M. W., Harlow, F. H., and Bromberg, E. The particle-in-cell method for hydrodynamic calculations. Tech. rep., Los alamos mational lab NM, 1957.

[6] Gingold, R. A., and Monaghan, J. J. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly notices of the royal astronomical society 181*, 3 (1977), 375–389.

[7] Kass, M., and Miller, G. Rapid, stable fluid dynamics for computer graphics. In *ACM Siggraph Computer Graphics* (1990), vol. 24, ACM, pp. 49–57.

[8] Kellom, T. *Large-Scale Water Simulation in Games*. Tampere University of Technology. Publication. Tampere University of Technology, 12 2015. Awarding institution:Tampere University of Technology.

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

[10] Stam, J. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 121–128.

[11] Stam, J. Real-time fluid dynamics for games. In *Proceedings of the game developer conference*, vol. 18. 2003, pp. 25–41.

[12] Tessendorf, J., et al. Simulating ocean water. *Simulating nature: realistic and interactive techniques. SIGGRAPH 1*, 2 (2001), 5–23.

[13] Thürey, N., Rüde, U., and Stamminger, M. Animation of open water phenomena with coupled shallow water and free surface simulations. In *Pro-*

ceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation (2006), Eurographics Association, pp. 157–164.

[14] Yuksel, C., House, D. H., and Keyser, J. Wave particles. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 26*, 3 (2007), 99–107.