Ripple Simulation on Water Surface

Chujie Chen, Siying Cen, Shuran Wen

https://colab.research.google.com/drive/18RBOzwB4vzlnpgpMqJgie7MUNoBf9iMG#scrollTo=AGMOeAe25AyT
https://github.gatech.edu/scen9/CSE6730

Abstract

In this project, we aim to simulate the formation procedure of ripples when droplets drop on a water surface. The dynamics of the water surface is represented by the amplitude change of each point.

Introduction

Wave is propagation of vibration. The wave propagation direction which is perpendicular to the particle vibration direction is called a transverse wave, while the same is called a longitudinal wave, and a water wave is a superposition of a transverse wave and a longitudinal wave.

There have been many methods proposed for water wave simulation. One classic class of simulation methods is wave superposition, including *Sinusoids* wave superposition and *Gerstner*'s wave [1] superposition. Other classes include FFT methods [2], wave particle methods [3] and flow map methods.

The nature of waves are vibrations and energy transmission. As a result, the distribution of energy is a representation of the distribution of waves, which can be represented by wave amplitudes.

Conceptual model of the system

Due to the wave propagation characteristics, the vibration at a certain point of the next moment is affected by the combination of the vibration of the surrounding particles and its own vibration. To simplify this, we assume that the amplitude *Ai* of point *Xi* is not only affected by itself, but also affected by four points (*Xi−W, Xi+W, Xi−1, Xi+1*) around it. We also assume that the influence of the four points on the Xi point is equal and linearly superposition.

Thus, the formula of calculating amplitude Ai of point Xi after one step can be summarized as:

$$A_i^{'} = \alpha(A_{i-w} + A_{i+w} + A_{i-1} + A_{i+1}) + bA_i$$

After approximation, this formula can be described as [4]:

$$A_i^{'} = \frac{1}{2}(A_{i-w} + A_{i+w} + A_{i-1} + A_{i+1}) - A_i$$

Besides, we need to take attenuation into consider, or the water surface will keep waving when a wave source is added. Therefore, we also add a dampening factor in our model, so that after each iteration, the amplitude will be reduced by a certain percentage from the ideal value:

$$A_i^{'} = dampening\ factor\ *\ A_i^{'}$$

Repeating this one-step update, we can simulate the amplitudes of each point on the water surface after *n* steps.

When visualizing, because of the refraction of water, when the water surface is not perpendicular to our line of sight, the underwater scenery we see is not directly below the observation point, but has a certain offset. To handle this offset, we simply do a linear approximation, which in particular means by calculating the difference in amplitude between the up,down, left and right points of a certain point to represent the deviation of the underwater scenery shift.

We store the pixel information of the original image in two arrays, one is used to store the original image data, and the other is used to store the updated data in each iteration. We also use an inverter to handle the image restoration problem when updating the image by giving each point an offset in the opposite direction. At last, copy each pixel on the original image to the updated data according to the offset.

In order to form a wave, we add a wave source on a calm water surface like throwing a stone into a pool, and the size and energy of the wave source formed are related to the radius of the stone and the force you throw at it. In order to simulate the wave source, we modify the wave amplitude array around the center point at the beginning.The radius of the wave source is controlled by modifying the amplitude in the circle around the center point with a certain radius.

Platform

We use Python to develop our model and visualize the result. In particular, we implement the data structure definition and detail logic in python files, and use the jupyter notebook to initialize the objects and call the functions.

Experiments

In our system, we simulate on a water surface of 500 x 300 points. 10 random droplets are generated on this surface, which means that 10 ripples will be created from these 10 start points. A dampening factor (0.95) is assigned to each point, considering the existence of dampening effect when water wave broadcasts.

Finally, we display the amplitudes array as a matrix in a new figure window after certain simulation steps.

We defined two classes (‘liquid.py’ and ‘drop.py.) and functions under “src” folder in our github repo (https://github.com/wenshuran/CSE6730Proj) and run simulations in google colab

(https://colab.research.google.com/drive/18RBOzwB4vzlnpgpMqJgie7MUNoBf9iMG#scrollTo=AGMOeAe25AyT) .

Progress

- Codes for our model

```
class drop:
    def __init__(self, x, y, scale=1.0, purity=1.0):
        scale = max(scale, 0.0)
        scale = min(scale, 1.0)
        self.x = x
        self.y = y
        self.amplitude = 255 * scale
        # TODO (low priority):
        # purity: when drop is not pure water, surface's dampening factor
        # gets affected locally?
        self.purity = purity
```

The “**drop**” class defines drops.

```
import numpy as np
class liquid:
    def __init__(self, height, width, dampening=0.95):
        self.height = height
        self.width = width
        self.dampening = 0.95
        self.value = np.zeros((self.height, self.width), dtype=float)
        # TODO: store self.value in self.history everytime it gets update
        # scipy.sparse may be needed to save memory
        # self.history = np.array([])


    def clear(self):
        self.value = np.zeros(self.height, self.width)
        return self.value

    def clear_region(self, hstart, wstart, hend, wend):
        # TODO: clear rectangular region
        pass

    def shape(self):
        return (self.height, self.width)

    def inspect(self):
        return self.value
```

```
    def take_drops(self, drops):
```

```
        for drop in drops:
            try:
                self.value[drop.x][drop.y] = drop.amplitude
            except:
                raise Exception("can not set drop at {},{} with value {}".format(drop.x, drop.y, drop.amplitude))

    def update_one_step(self):
        # TODO: 4 edges with special care
        # TODO: vectorization to speed things up
        curr = np.copy(self.inspect())
        nxt = np.copy(self.inspect())
        for i in range(1, self.height - 1):
            for j in range(1, self.width - 1):
                nxt[i][j] = (curr[i-1][j]+curr[i+1][j]+curr[i][j-1]+curr[i][j+1])/2 - curr[i][j]
                nxt[i][j] = nxt[i][j] * self.dampening
        self.value = nxt
        return self.inspect()


    def update_n_step(self, n=1):
        for i in range(n):
            self.update_one_step()
        return self.inspect()
```

The ”**liquid**” class defines the initialization of water surface and ‘update_one_step’, ‘update_n_step’ methods.

- Codes for running simulation tests

```
[5]  import sys
     sys.path.append("H2O-Sim/src")
     from water.drop import drop
     from surface.liquid import liquid
```

```
[6]  import numpy as np
     import scipy as sp
     import scipy.sparse
     import matplotlib.pyplot as plt
     %matplotlib inline
     from ipywidgets import interact, IntSlider
```

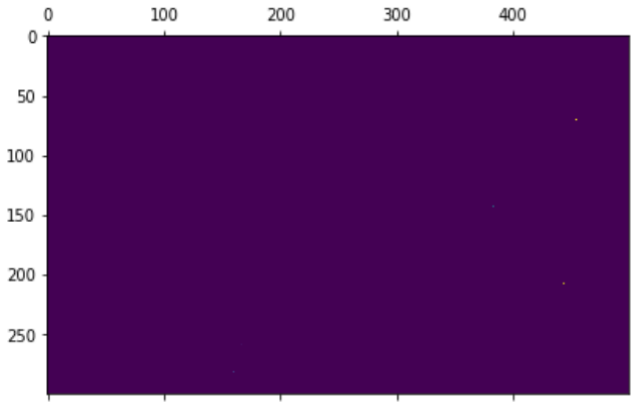
```
[7]  canvas = liquid(300,500)
```

```
[8]  import random
     drops = []
     for i in range(10):
         drops.append(drop(random.randint(0,300-1), random.randint(0,500-1), random.uniform(0, 1.0)))
     canvas.take_drops(drops)
```

- Preliminary results

```
[18] plt.matshow(canvas.inspect())
```

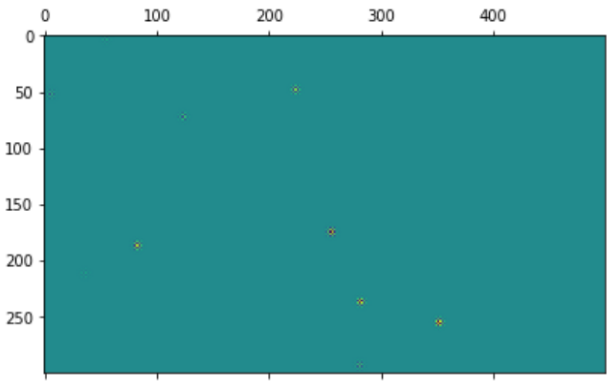
```
<matplotlib.image.AxesImage at 0x7f15e0b907d0>
```



Initial drops on water surface of 300 x 500.

```
[12] canvas.update_n_step(20)
     plt.matshow(canvas.inspect())
```

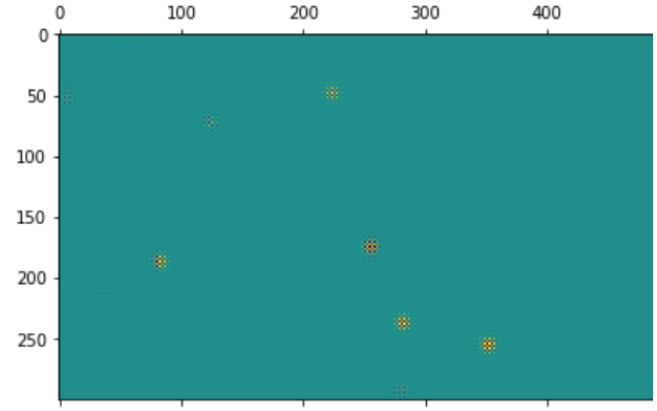
```
<matplotlib.image.AxesImage at 0x7f15e0d84210>
```



Amplitudes after 20 steps' simulation.

```
[13] canvas.update_n_step(50)
     plt.matshow(canvas.inspect())
```

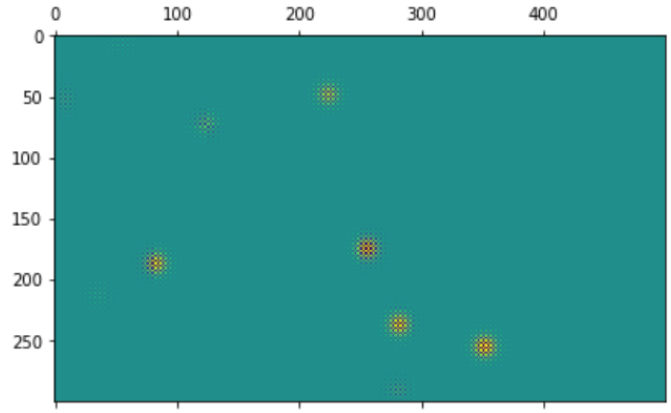
```
<matplotlib.image.AxesImage at 0x7f15e0cb0e50>
```



Amplitudes after 50 steps' simulation.

```
[14] canvas.update_n_step(100)
     plt.matshow(canvas.inspect())
```

```
<matplotlib.image.AxesImage at 0x7f15e0c2d9d0>
```



Amplitudes after 100 steps' simulation.

Changes in direction

After discussion, we determine to keep working on simulation of ripple formation on water surface, not simulation of drops on solid surface.

Division of labor

- Chujie Chen: improve ripple simulation model on water surface; create tests in google colab
- Shuran Wen: explore other methods
- Siying Cen: explore other methods; write report

Reference

[1] Fournier, Alain, and William T. Reeves. "A simple model of ocean waves." Proceedings of the 13th annual conference on Computer graphics and interactive techniques. 1986.
[2] Mastin, Gary A., Peter A. Watterberg, and John F. Mareda. "Fourier synthesis of ocean scenes." IEEE Computer graphics and Applications 7.3 (1987): 16-23.
[3] 2010, Yuksel C. Real-time water waves with wave particles[M]. Texas A&M University

[4] https://www.cnblogs.com/youyoui/p/8546178.html