Introduction:

Consider a rectangular area with all sides enclosed.
There are two points in the area that will periodically generate waves.
These points can be considered source points and they both will generate a wave with the exact same amplitude at any given time.
Next, consider a ball of some mass and initial velocity.
This project will focus on the movement of the ball through the wave field using simple momentum equations to derive the interaction between the water and the ball.

Method: Shallow water wave field

Start off by figuring out the wave field.
Assuming constant density, a flat bottom topography, ignoring surface tension, having a large Rossby radius so coriolis force is not important, and considering waves with much longer wavelengths than the depth of the fluid, the shallow water equations are:

$$\frac{\partial \eta}{\partial t} + \frac{\partial}{\partial x}(\eta u) + H\frac{\partial u}{\partial x} + \frac{\partial}{\partial y}(\eta v) + H\frac{\partial v}{\partial y} = 0$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + g\frac{\partial \eta}{\partial x} = 0$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + g\frac{\partial \eta}{\partial y} = 0$$

$\eta(x, y, t)$ is the displacement height of the water from the mean undisturbed surface height
$u(x, y, t)$ is the east-west water velocity
$v(x, y, t)$ is the north-south water velocity
H is the mean height of the horizontal pressure surface (mean undisturbed surface height)
g is the gravitational acceleration
We do not assume that perturbations have small amplitudes, so that nonlinear terms are not neglected.

Next, we have to discretize our equations. We will be using the leapfrog scheme for all equations in space and time.
The chosen grid is a non-staggered grid for no other reason than that we are familiar with it.

Let the spatial steps be x_i, y_j where $0 \le i \le n$, $0 \le j \le m$ and can be represented like A_i^j
Let the time step be t_k where $0 \le k \le l$ and the value of each step is dx, dy or dt.
Discretizing our shallow water equations we get:

$$\frac{\eta_i^j(t_{k+1}) - \eta_i^j(t_{k-1})}{2dt} + u_i^j(t_k)\left(\frac{\eta_{i+1}^j(t_k) - \eta_{i-1}^j(t_k)}{2dx}\right) + \left(\eta_i^j(t_k) + H\right)\frac{u_{i+1}^j(t_k) - u_{i-1}^j(t_k)}{2dx}$$
$$+ v_i^j(t_k)\left(\frac{\eta_i^{j+1}(t_k) - \eta_i^{j-1}(t_k)}{2dy}\right) + \left(\eta_i^j(t_k) + H\right)\frac{v_i^{j+1}(t_k) - v_i^{j-1}(t_k)}{2dy} = 0$$

$$\frac{u_i^j(t_{k+1}) - u_i^j(t_{k-1})}{2dt} + u_i^j(t_k)\left(\frac{u_{i+1}^j(t_k) - u_{i-1}^j(t_k)}{2dx}\right) + v_i^j(t_k)\left(\frac{u_i^{j+1}(t_k) - u_i^{j-1}(t_k)}{2dy}\right)$$
$$+ g\left(\frac{\eta_{i+1}^j(t_k) - \eta_{i-1}^j(t_k)}{2dx}\right) = 0$$

$$\frac{v_i^j(t_{k+1}) - v_i^j(t_{k-1})}{2dt} + u_i^j(t_k)\left(\frac{v_{i+1}^j(t_k) - v_{i-1}^j(t_k)}{2dx}\right) + v_i^j(t_k)\left(\frac{v_i^{j+1}(t_k) - v_i^{j-1}(t_k)}{2dy}\right)$$
$$+ g\left(\frac{\eta_i^{j+1}(t_k) - \eta_i^{j-1}(t_k)}{2dy}\right) = 0$$

Isolate for the 'next' time values:

$$\eta_i^j(t_{k+1}) = \eta_i^j(t_{k-1}) - dtu_i^j(t_k)\left(\frac{\eta_{i+1}^j(t_k) - \eta_{i-1}^j(t_k)}{dx}\right) - dt\left(\eta_i^j(t_k) + H\right)\frac{u_{i+1}^j(t_k) - u_{i-1}^j(t_k)}{dx}$$
$$- dtv_i^j(t_k)\left(\frac{\eta_i^{j+1}(t_k) - \eta_i^{j-1}(t_k)}{dy}\right) - dt\left(\eta_i^j(t_k) + H\right)\frac{v_i^{j+1}(t_k) - v_i^{j-1}(t_k)}{dy}$$

$$u_i^j(t_{k+1}) = u_i^j(t_{k-1}) - dtu_i^j(t_k)\left(\frac{u_{i+1}^j(t_k) - u_{i-1}^j(t_k)}{dx}\right) - dtv_i^j(t_k)\left(\frac{u_i^{j+1}(t_k) - u_i^{j-1}(t_k)}{dy}\right)$$
$$- dtg\left(\frac{\eta_{i+1}^j(t_k) - \eta_{i-1}^j(t_k)}{dx}\right)$$

$$v_i^j(t_{k+1}) = v_i^j(t_{k-1}) - dtu_i^j(t_k)\left(\frac{v_{i+1}^j(t_k) - v_{i-1}^j(t_k)}{dx}\right) - dtv_i^j(t_k)\left(\frac{v_i^{j+1}(t_k) - v_i^{j-1}(t_k)}{dy}\right)$$
$$- dtg\left(\frac{\eta_i^{j+1}(t_k) - \eta_i^{j-1}(t_k)}{dy}\right)$$

Now we can solve for the next time values except at the first time step because we will need a t(k = -1) value which doesn't exist.
So using a predictor-corrector based on the forward Euler scheme in time will be used to give us the values at the first time step:

$$\frac{\eta_i^j(t_{k+1}) - \eta_i^j(t_k)}{dt} + u_i^j(t_k)\left(\frac{\eta_{i+1}^j(t_k) - \eta_{i-1}^j(t_k)}{2dx}\right) + \left(\eta_i^j(t_k) + H\right)\frac{u_{i+1}^j(t_k) - u_{i-1}^j(t_k)}{2dx}$$
$$+ v_i^j(t_k)\left(\frac{\eta_i^{j+1}(t_k) - \eta_i^{j-1}(t_k)}{2dy}\right) + \left(\eta_i^j(t_k) + H\right)\frac{v_i^{j+1}(t_k) - v_i^{j-1}(t_k)}{2dy} = 0$$

$$\frac{u_i^j(t_{k+1}) - u_i^j(t_k)}{dt} + u_i^j(t_k)\left(\frac{u_{i+1}^j(t_k) - u_{i-1}^j(t_k)}{2dx}\right) + v_i^j(t_k)\left(\frac{u_i^{j+1}(t_k) - u_i^{j-1}(t_k)}{2dy}\right)$$
$$+ g\left(\frac{\eta_{i+1}^j(t_k) - \eta_{i-1}^j(t_k)}{2dx}\right) = 0$$

$$\frac{v_i^j(t_{k+1}) - v_i^j(t_k)}{dt} + u_i^j(t_k)\left(\frac{v_{i+1}^j(t_k) - v_{i-1}^j(t_k)}{2dx}\right) + v_i^j(t_k)\left(\frac{v_i^{j+1}(t_k) - v_i^{j-1}(t_k)}{2dy}\right)$$
$$+ g\left(\frac{\eta_i^{j+1}(t_k) - \eta_i^{j-1}(t_k)}{2dy}\right) = 0$$

Then isolate them for the next time step values so we can solve:

$$\eta_i^j(t_{k+1}) = \eta_i^j(t_k) - dtu_i^j(t_k)\left(\frac{\eta_{i+1}^j(t_k) - \eta_{i-1}^j(t_k)}{2dx}\right) - dt\left(\eta_i^j(t_k) + H\right)\frac{u_{i+1}^j(t_k) - u_{i-1}^j(t_k)}{2dx}$$
$$- dtv_i^j(t_k)\left(\frac{\eta_i^{j+1}(t_k) - \eta_i^{j-1}(t_k)}{2dy}\right) - dt\left(\eta_i^j(t_k) + H\right)\frac{v_i^{j+1}(t_k) - v_i^{j-1}(t_k)}{2dy}$$

$$u_i^j(t_{k+1}) = u_i^j(t_k) - dtu_i^j(t_k)\left(\frac{u_{i+1}^j(t_k) - u_{i-1}^j(t_k)}{2dx}\right) - dtv_i^j(t_k)\left(\frac{u_i^{j+1}(t_k) - u_i^{j-1}(t_k)}{2dy}\right)$$
$$- dtg\left(\frac{\eta_{i+1}^j(t_k) - \eta_{i-1}^j(t_k)}{2dx}\right)$$

$$v_i^j(t_{k+1}) = v_i^j(t_k) - dtu_i^j(t_k)\left(\frac{v_{i+1}^j(t_k) - v_{i-1}^j(t_k)}{2dx}\right) - dtv_i^j(t_k)\left(\frac{v_i^{j+1}(t_k) - v_i^{j-1}(t_k)}{2dy}\right)$$
$$- dtg\left(\frac{\eta_i^{j+1}(t_k) - \eta_i^{j-1}(t_k)}{2dy}\right)$$

After solving for t = dt, we average the values with those at t = 0 from our initial conditions.
The averaged values will be the estimated values at t = dt/2.
Finally, we use t = dt/2 in our usual centred difference scheme to get values for t = dt.

What our non-staggered grid looks like:
Consider a 2d grid with x in the horizontal and y in the vertical. $0 \leq x \leq L,\ 0 \leq y \leq W$
Each point will consist of a $u,\ v,\ \eta,\ t$ value.
Ex. some arbitrary point $(\frac{L}{2}, \frac{W}{3})$ will have $u(\frac{L}{2}, \frac{W}{3}, t),\ v(\frac{L}{2}, \frac{W}{3}, t),\ \eta(\frac{L}{2}, \frac{W}{3}, t)$
We can solve our equations for the t_{k+1}values and generate our field.
Initial conditions:
$\eta(\frac{L}{5}, \frac{3W}{4}, 0) = \eta_0$
$\eta(\frac{4L}{5}, \frac{W}{2}, 0) = \eta_0$
$u(x, y, 0) = 0$
$v(x, y, 0) = 0$
Boundary conditions:
Got walls on all sides so,
$u(0, y, t) = u(L, y, t) = u(x, 0, t) = u(x, W, 0) = 0$
$v(0, y, t) = v(L, y, t) = v(x, 0, t) = v(x, W, 0) = 0$
To simulate reflecting boundary conditions for the waves set η at the walls to be zero.
$\eta(0, y, t) = \eta(L, y, t) = \eta(x, 0, t) = \eta(x, W, 0) = 0$

Method: The Object in the wave field

Consider a ball with some mass and initial velocity.
Ignore drag in the water.
The ball also stays at the same vertical height for the entire duration.
We will treat the water as round packets with dimensions dx (width), dy (height). If we use the same spatial step size (can be different total steps for rectangle), then we have a sphere.
Since we ignore drag in the water, if the ball moves through an area with no flow, then we can view it as just passing through the spherical water packets.
If the ball encounters a water with u, v-velocity, then we will consider the momentum equations:
m1v1(initial) + m2v2(initial) = m1v1(final) + m2v2(final)
Where 1 denotes the ball and 2 denotes the water packets.
For simplicity, we will also not deal with the momentum and thus velocity change of the water packets after collision with the ball. This would become far too complicated as the wave field would be changing locally around the ball at each timestep. Only the ball will change.
Lastly, to actually solve for the momentum of the ball, we consider that after a collision, some energy will be lost to heat or whatever else.
Therefore, we can just say the momentum m2v2(final) is just some scalar multiple (<1) of m2v2(initial). This ideally should change with every collision but we kept it the same in this project.
So finally, we are solving for both **x and y direction separately:**

m1v1(final) = m1v1(initial) + m2v2(initial) - c*m1v1(initial)

Where m1, v1 are given and v2 is obtained from the shallow water wave field and it can be either u-velocity (+ve is east, -ve is west) or v-velocity (+ve is north, -ve is south).
m2 is just $4/3\pi r^3*\rho$
where r is just dx/2 or dy/2 since it is just a sphere and density ρis taken as 1000 kg/m^3
We can now solve for each new v1 value of the ball for as many timesteps as we want by taking v1 * dt and moving the ball by that much.

Results and Code summary:

Create a class that gives us empty 2D matrices to store our u, v, h values (h is now elevation η).
Store them in a 3D matrix with each layer corresponding to the timestep.
Shift the old u, v, h values to become the new values which will let us generate the next values.
The ball_water function gives us the velocity of the ball using the momentum equations mentioned in the section above.
To move our ball we simply let the new position be:
Old position + v1*dt
To generate the animation, we let each timestep be a new frame in the animation and use the surface plots from mpl_tookits to generate both a surface for the ball and the waves.
Finally, we attempted to smooth out our wave field by using a 2D spline interpolation from SciPy but it didn't work out.

As for the behaviour and movement of the ball, it is quite straightforward.
From our equations it is obvious that if we let our ball have a large mass compared with the water packets, then it will not change momentum as easily and stay on its initial trajectory.
On the flipside, a light ball will be pushed around by the waves.
In addition, because our boundary conditions have u-velocity and v-velocity at the walls equal to zero, once our ball reaches a wall, it will only move due to its current momentum and should eventually reach a corner because we have assumed zero drag in water. Once at a corner, it will remain there indefinitely.

References

Allen, S., Guo, C., Yang, L. (2020). *Laboratory 7: Solving partial differential equations using an explicit, finite difference method.* Numeric course.

"Inelastic Collision." Wikipedia, Wikimedia Foundation, 23 Jan. 2020, en.wikipedia.org/wiki/Inelastic_collision.

Mei, C., C. (Fall 2004). CHAPTER FOUR. WAVES IN WATER. Retrieved from http://web.mit.edu/1.138j/www/material/chap-4.pdf.

SciPy.org. (2014, May 11). ENTHOUGHT. "Interpolation (scipy.interpolate)". Retrieved from https://docs.scipy.org/doc/scipy-0.14.0/reference/tutorial/interpolate.html#two-dimensional-spline-representation-object-oriented-bivariatespline

Segur, H., (Lecturer). Yamamoto, H., (Write-up). (2009, June 18). "Lecture 8: The Shallow-Water Equations".

"Shallow Water Equations." Wikipedia, Wikimedia Foundation, 20 Mar. 2020, en.wikipedia.org/wiki/Shallow_water_equations#Kinematic_wave

Calculations for variables at timestep 1, t = dt:

Initial conditions:

$\eta(\frac{L}{5}, \frac{3W}{4}, 0) = \eta_0$ - using our x, y parameters: $\eta(\frac{n+1}{5}, \frac{3(m+1)}{4}, 0) = \eta_0$

$\eta(\frac{4L}{5}, \frac{W}{2}, 0) = \eta_0$ - using our x, y parameters: $\eta(\frac{4(n+1)}{5}, \frac{m+1}{2}, 0) = \eta_0$

$u(x, y, 0) = 0$

$v(x, y, 0) = 0$

We have designed our model so that:

# of timesteps = l + 1

Rows = m + 1

Cols = n + 1

I don't know why we did this but let us continue:

Let (n+1)/5 = x1, 3(m+1)/4 = y1

4(n+1)/5 = x2, (m+1)/2 = y2

Forward euler equations:

$$\eta_i^j(t_{k+1}) = \eta_i^j(t_k) - dtu_i^j(t_k)\left(\frac{\eta_{i+1}^j(t_k) - \eta_{i-1}^j(t_k)}{2dx}\right) - dt\left(\eta_i^j(t_k) + H\right)\frac{u_{i+1}^j(t_k) - u_{i-1}^j(t_k)}{2dx}$$

$$- dtv_i^j(t_k)\left(\frac{\eta_i^{j+1}(t_k) - \eta_i^{j-1}(t_k)}{2dy}\right) - dt\left(\eta_i^j(t_k) + H\right)\frac{v_i^{j+1}(t_k) - v_i^{j-1}(t_k)}{2dy}$$

Code Walkthrough

Class Wave_field():

- With *__init__* we first give our class several attributes for a simple grid
    - *n* as the columns or x-axis
    - *m* as the rows or y-axis
    - *l* as the timestep
    - *prev, now, next* -> creates a grid with *n, m* dimensions
    - *field* -> includes the timestep for the grid
- Note that our model takes columns, rows, timesteps as n+1, m+1, l+1 respectively

- Method *fullgrid_time(self, timestep, attr='next'):*
    - Creates and assigns a grid value to *self.field* at the *timestep* specified
    - Grid value is determined by the *attr* with a default value of *self.next*

- Method *shift(self):*
    - When called, we are shifting the grid values 1 timestep forward

Function initial_cond():

- Sets initial condition *for* all *self.prev* grids
- All velocities and mean surface heights at 0 except our drum amplitudes

Function b_cond():

- Inputs the boundary conditions; to be called at each timestep
- We are using 'no-slip' and 'no-penetration' conditions so fluid particles on the wall move with the velocity of the wall (zero in this case)
- Lastly the two drums have h_ampltitude which follows a cosine function

Function t1():

- Our leapfrog discretization for the shallow water wave equations require a 'previous' timestep to start the calculations
- So *t1* will calculate the values at the first timestep using the *Forward Euler* method.
- Not all points need to be considered. Only those closest to the drum will initially see any effect.

Function leap_frog():

- Our primary discretization method that we will use to iterate through the timesteps to get the water velocities and the surface heights

Function ball_water():

- Tracks a ball through our wave field using simple (not-realistic) momentum equations
- Trajectory affected by velocities *u, v,* and by height *h*
- Returns the x and y positions on our field for the ball at each timestep

Function model():

- Puts everything together as one call function.
- Includes an **animation** plot to visualize the numbers calculated for the wave field