GAMES103 Lab 4: Pool Ripples

Due Date: 1/17/2022, 11:59PM

1 Requirements

In this project, you will learn how to solve a simplified shallow wave model to simulate ripples of a water pool and experience the two-way coupling technique. In the provided example Unity package, you will find a planar mesh resized into $size \times size$. You will also see two blocks inside of the pool and you can click and drag the blocks to move them around.

You have two jobs: to simulate ripple effects by adjusting the y values of the mesh vertices; and to simulate proper interactions between blocks and water surface.

1.a. Basic setup (1 Point) At the beginning of the `update` function, load the heights (the y values) of the vertices into `h`. At the end of the `update` function, set `h` back into the heights of the vertices. Remember to recalculate the mesh normal in the end.

1.b. User Interaction (1 Point) When the player hits the 'r' key, add a random water drop into the pool. To do so, you can increase the column `h[i,j]` by r, in which i, j and r are all randomly determined. One problem is that it can cause more water volume to be poured into the pool over time, according to the shallow wave model. To solve this problem, simply deduce the same amount (r) from the surrounding columns so the total volume stays the same.

After implementing the above procedure, you should observe a bump on the surface whenever the key gets pressed, but no wave propagation yet.

1.c. Ripples (3 Points) In the `update` function, call the `Shallow_Wave` function eight times. In the `Shallow_Wave` function, compute the new height `new_h` as:

$$h_{i,j}^{new} \leftarrow h_{i,j} + (h_{i,j} - h_{i,j}^{old}) * damping + (h_{i-1,j} + h_{i+1,j} + h_{i,j-1} + h_{i,j+1} - 4h_{i,j}) * rate. \quad (1)$$

in which $damping = 0.996$ and $rate = 0.005$. Specify Neumann boundary conditions on the grid boundary. After you compute $h_{i,j}^{new}$ for all of the columns, set $h_{i,j} = h_{i,j}^{new}$ and $h_{i,j}^{old} = h_{i,j}$. Now you should see waves being simulated.

1.d. One-way coupling (4 Points) For simplicity, let's consider one-way coupling from blocks to water first. Based on block positions, you can figure out the columns in contact with the blocks, labeled out as `mask`, and the desired heights of these columns, stored in an array `low_h`. Use the conjugate gradient (CG) method to solve the Poisson equation $\mathbf{Av} = \mathbf{b}$, in which $\mathbf{A}$ is the masked Laplacian matrix and $\mathbf{b} = (\texttt{new\_h} - \texttt{low\_h})/rate$. The example package includes a CG solver so you can just call that function. Remember to set the solvable region as well to optimize the performance. Do this for both blocks.

The result $\mathbf{v}$ can be intuitively understood as virtual pressure pushing water out of a column. If you use it directly, it can cause very large and unrealistic waves, fundamentally due to explicit integration used for ripple update. To solve this problem, simply diminish it as $\mathbf{v} \leftarrow \gamma\mathbf{v}$ and finally renew $h_{i,j}^{new}$:

$$h_{i,j}^{new} \leftarrow h_{i,j}^{new} + (v_{i-1,j} + v_{i+1,j} + v_{i,j-1} + v_{i,j+1} - 4v_{i,j}) * rate. \quad (2)$$

Once you implement the above, you should now see the ripples caused by block movement.

Bonus Credit (4 Points) Since you have already learned rigid body dynamics, can you attach rigid body dynamics with blocks as well, by using the computed $\mathbf{v}$ as the source of the force (and the torque)? Doing this will allow you to fully implement two-way coupling. (Note: Once the blocks are rotated, you cannot simply use block positions to calculate low_h. You can fix this by Unity's bounding box and raycasting function, but how?)

2 Submission Guideline

Save all of your files, including scene and script files, and export them into a package. Submit your package by the SmartChair system.