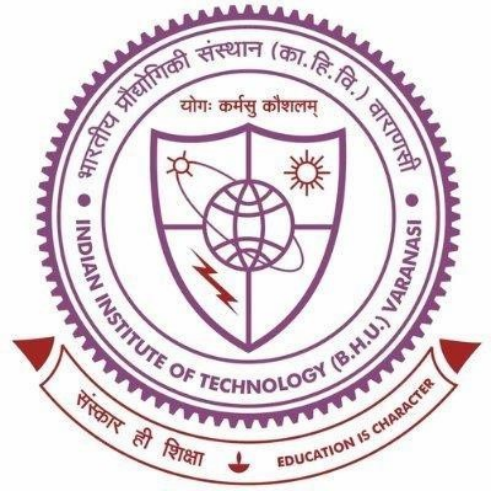
COMPUTER GRAPHICS CSO 351

SESSION 2022-23



MINI PROJECT

WATER RIPPLE SIMULATION

Ramesh Poorna Theja	**20075072**
Katukojwala Ritesh	**20075044**
Koushik K	**20075045**

INTRODUCTION

Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. OpenGL (open graphics library) is a standard specification that defines a crosslanguage, cross-platform API for creating 2D and 3D computer graphics applications. The interface is made up of over 250 distinct function calls that may be used to create elaborate 3D scenes from simple primitives. OpenGL serves two main purpose:

- To hide the complexities of interfacing with different 3D accelerators, by presenting programmer with a single, uniform API
- To hide the differing capabilities of hardware platforms, by requiring that all Implementations support the full OpenGL, feature set.

This report will discuss the simulation process for the effect water ripples.

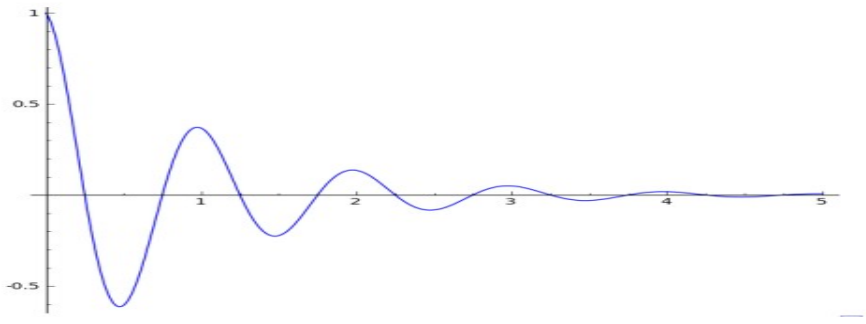
OVERVIEW

Water Surface definition

We have defined the water surface as plane in the 3D space, composed of a set of N * N connected vertices, where the point P (i, j, k) has 3 identifiers, i(X-coordinate), j (Y-coordinate), and k (Z-coordinate).

Vertex definition

For simulation purposes, a vertex's X and Y coordinates wont change throughout the whole process, but the variable parameter in this case will be the height of every vertex in the 3D space. Resulting in a curve something like Damped Sine Wave graph as the following:



When a water ripple is created on the water surface, the height of this vertex will change and as a consequence, all the neighbours' vertices heights will change according to Damped Sine Wave function.

Connecting water surface vertices

We have used Dynamic Programming to compute a vertex's height according to the neighbouring vertices, where the Z-axis value of a vertex in each state is defined by:

$$D(x,y) = \frac{\sum of\ neighbouring\ droplets}{2} - \frac{D(x,y)}{DAMP}$$

where,

D ⇒ a droplet on the water surface (vertex) .
x ⇒ X-coordinate of the point .
y ⇒ Y-coordinate of the point.
DAMP ⇒ Damping coefficient of water wave.

By running the previous recursive formula over all the vertices of the surface, every frame during the simulation process, we will get a simple yet beautiful wave simulation.

Implementation details

To let the user have a close look for the simulation process, We have added a user interaction feature, which allows the user to rotate the screen around the center of the screen. This is done using three-dimensional geometric transformations such as Rotating and Translation. The opengl functions used for this are **glTranslatef**() and **glRotatef**().

MODULES AND FUNCTIONS

- **glutInitDisplayMode(unsigned int mode)** requests a display with the properties in mode. The value of mode is determined by the logical OR of options including the colour model and buffering.

- **glutInitWindowSize (int width, int height)** specifies the initial position of the topleft corner of the window in pixels.

- **glutCreateWindow (char *title)** creates a window on the display. The string title can be used to label the window. The return value provides references to the window that can be used when there are multiple windows.

- **glutMouseFunc(void *f(int button, int state, int x, int y)** registers the mouse callback function f. The callback function returns the button, the state of button after the event and the position of the mouse relative to the top-left corner of the window.

- **glutKeyboardFunc(void(*func) (void))** function is called every time when you press enter key to resume the game or when you press ‘b’ or ‘B’ key to go back to the initial screen or when you press esc key to exit from the application.

- **glutMotionFunc(void(*func) (int x, int y))** set the motion and passive motion callback respectively for the current window.The motion callback for a window is called when the mouse moves within the window while one or more mouse buttons are pressed.

- **glutMainLoop()** causes the program to enter an event-processing loop. It should be the last statement in main function.

- **glutIdleFunc(void(*func) (void)** sets the global idle callback to be `func` so a GLUT program can perform background processing tasks or continuous animation when window system events are not being received.

- **glutInitWindowPosition(int x, int y)** intents the initial window position and size values to be provided as a suggestion to the window system for a window's initial size and position.

- **glPushMatrix()** pushes the current matrix stack down by one, duplicating the current matrix. That is, after a glPushMatrix() call, the matrix on top of the stack is identical to the one below it.

- **glPopMatrix()** pops the current matrix stack, replacing the current matrix with the one below it on the stack.

- **glTranslatef()** function multiplies the current matrix by a translation matrix.

- **glRotatef()** function multiplies the current matrix by a rotation matrix.

- **glutSwapBuffers()** performs a buffer swap on the layer in use for the current window.

- **glMatrixMode()** specifies which matrix stack is the target for subsequent matrix operations.

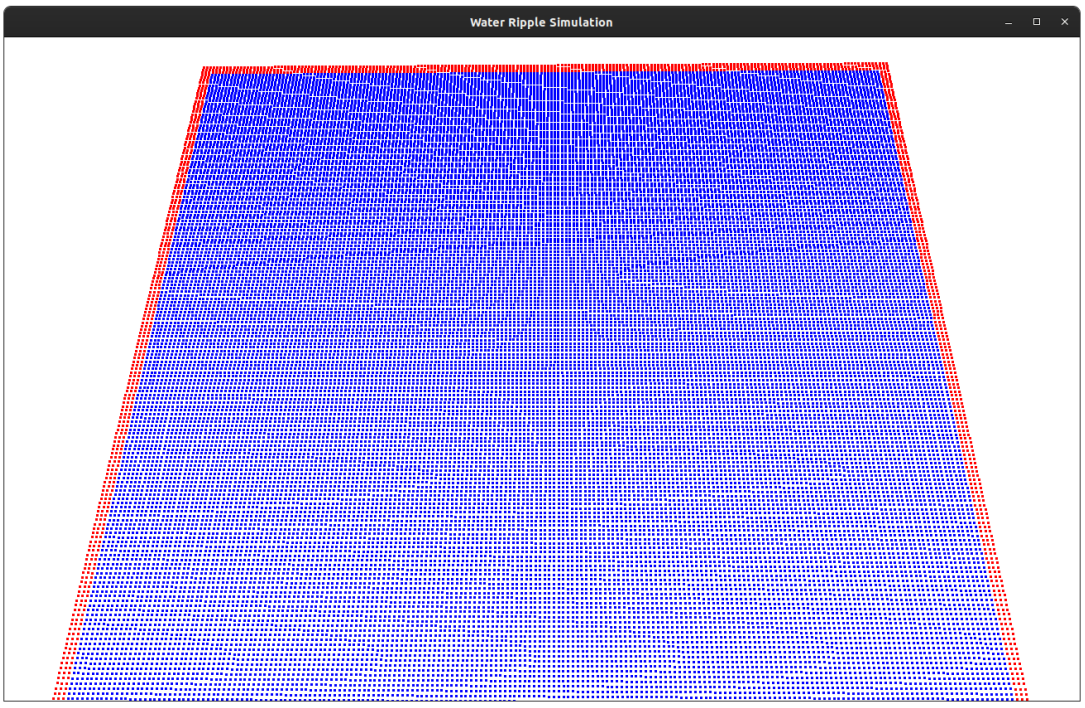
- **gluPerspective()** specifies a viewing frustum into the world coordinate system.

- **glutPostRedisplay()** marks the normal plane of current window as needing to be redisplayed.

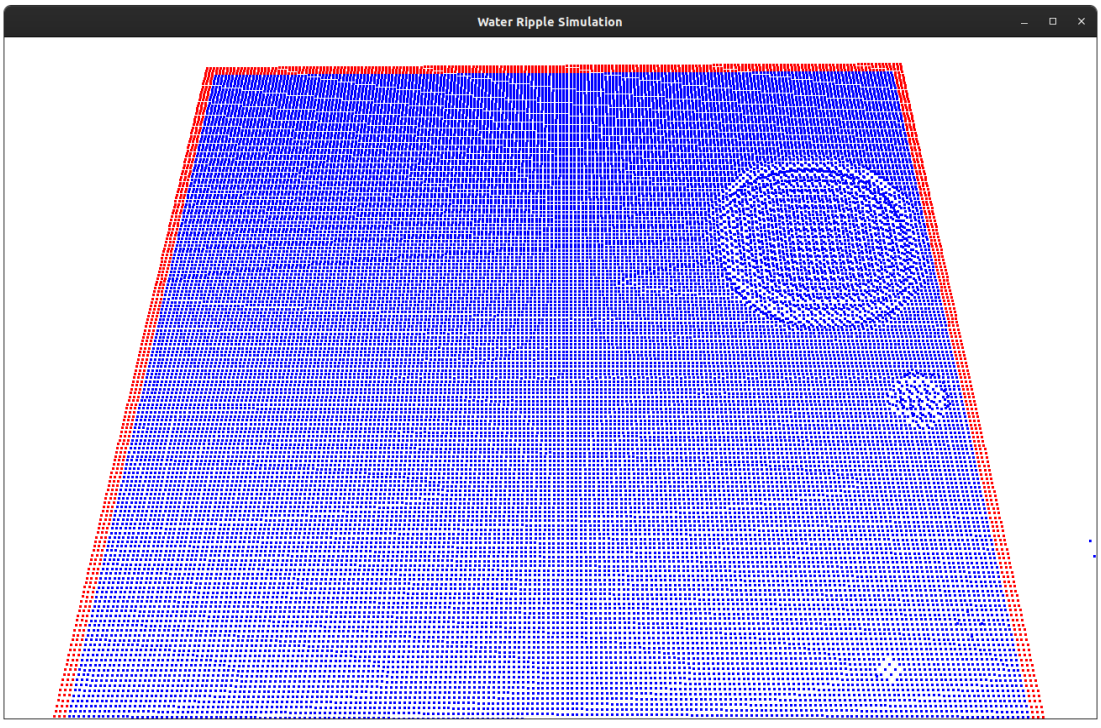
- **gluUnProject()** maps the specified window coordinates into object coordinates using model, proj, and view. The result is stored in objx, objy and objz. We used this to get the exact 3D coordinate from the mouse click.

RESULTS

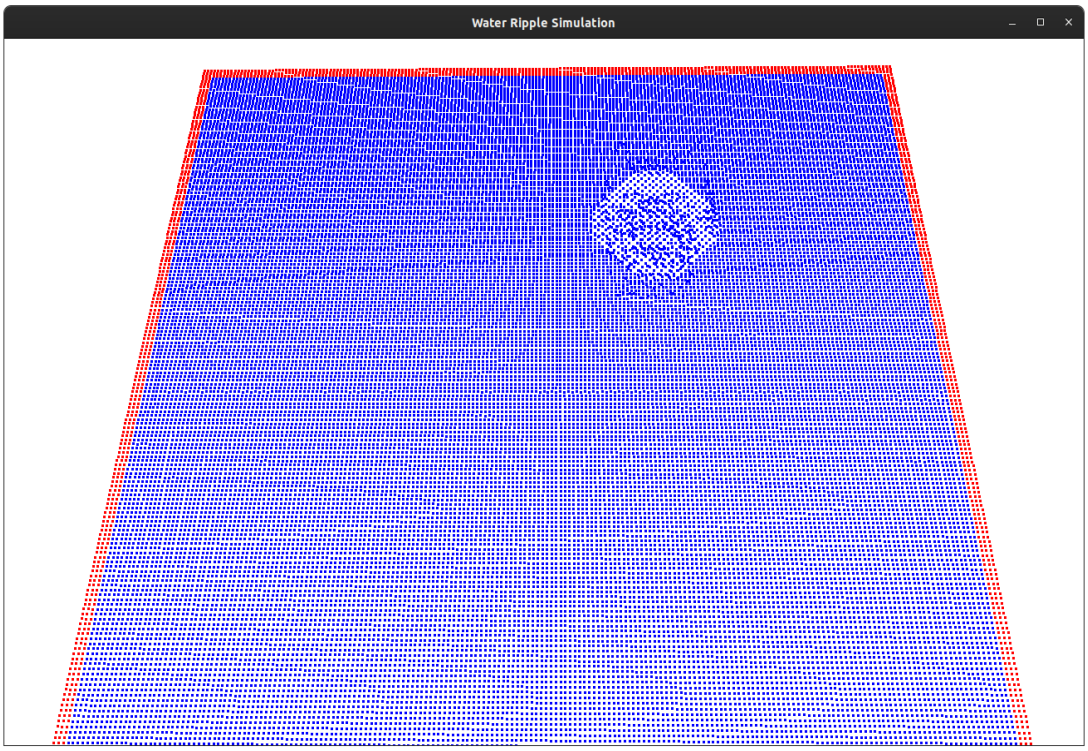
- **Ideal Water Plane**

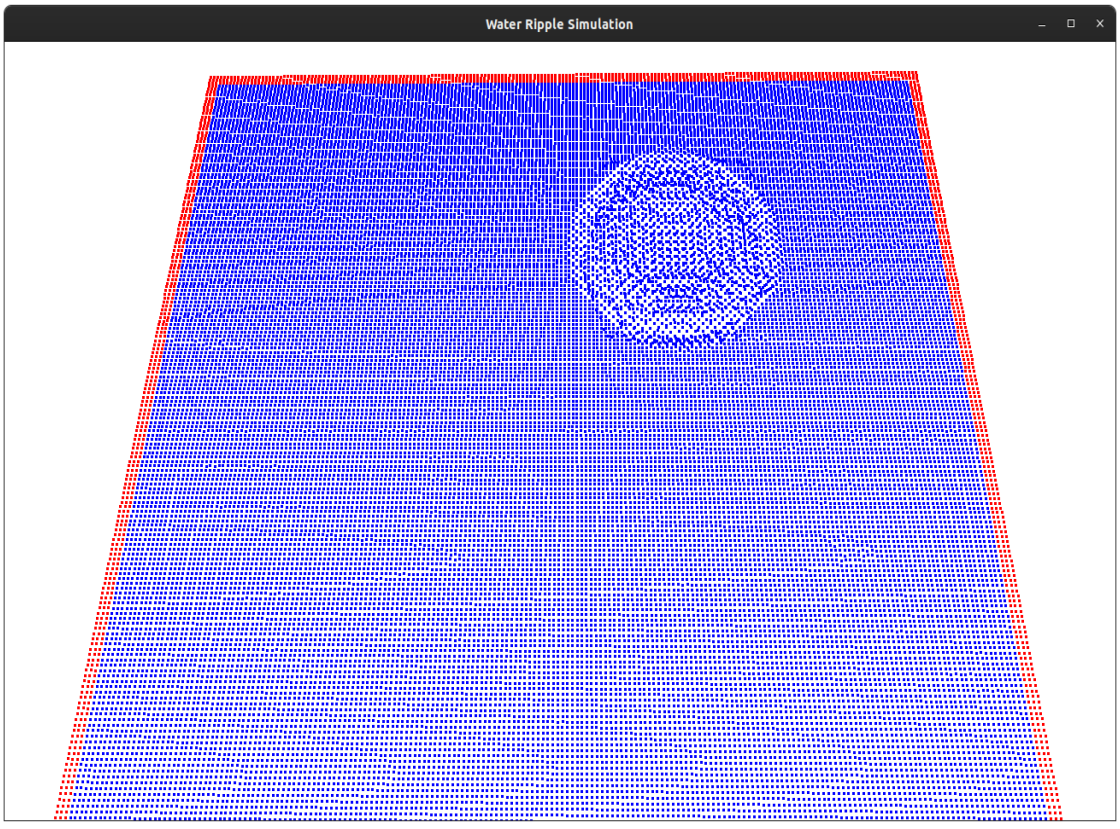


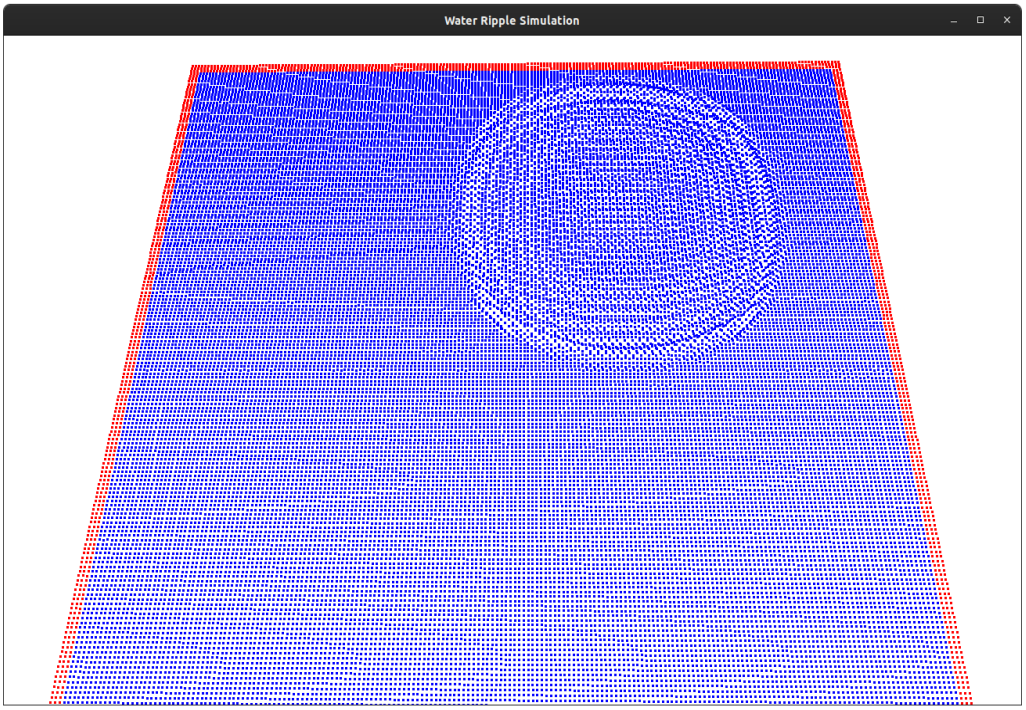
- **Random Ripples**



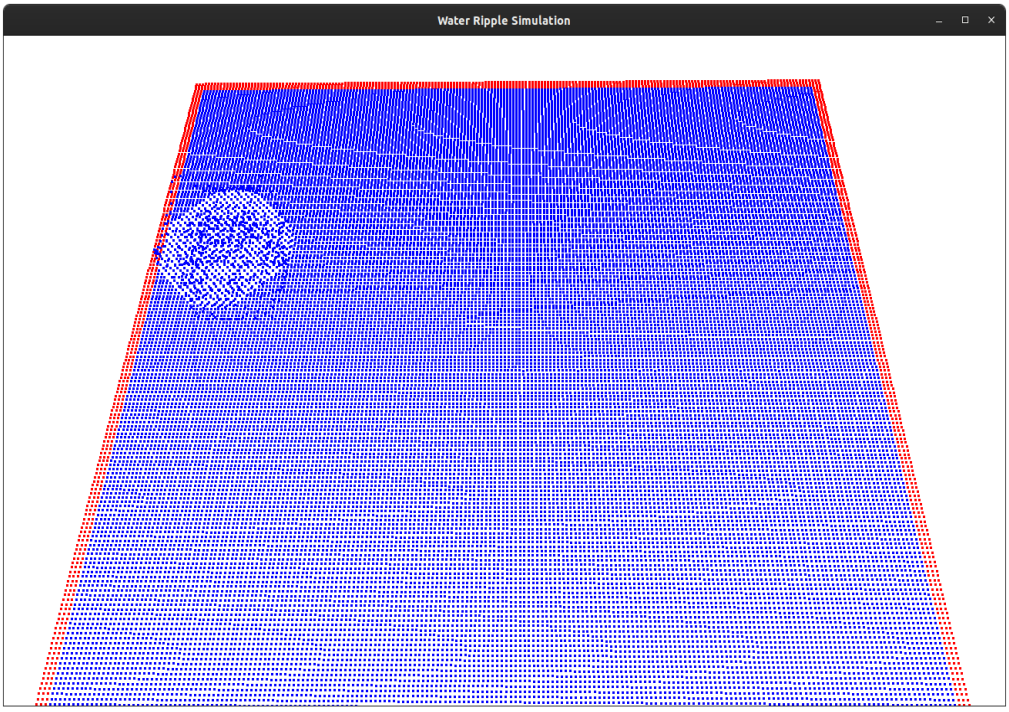
- **Mouse Click**

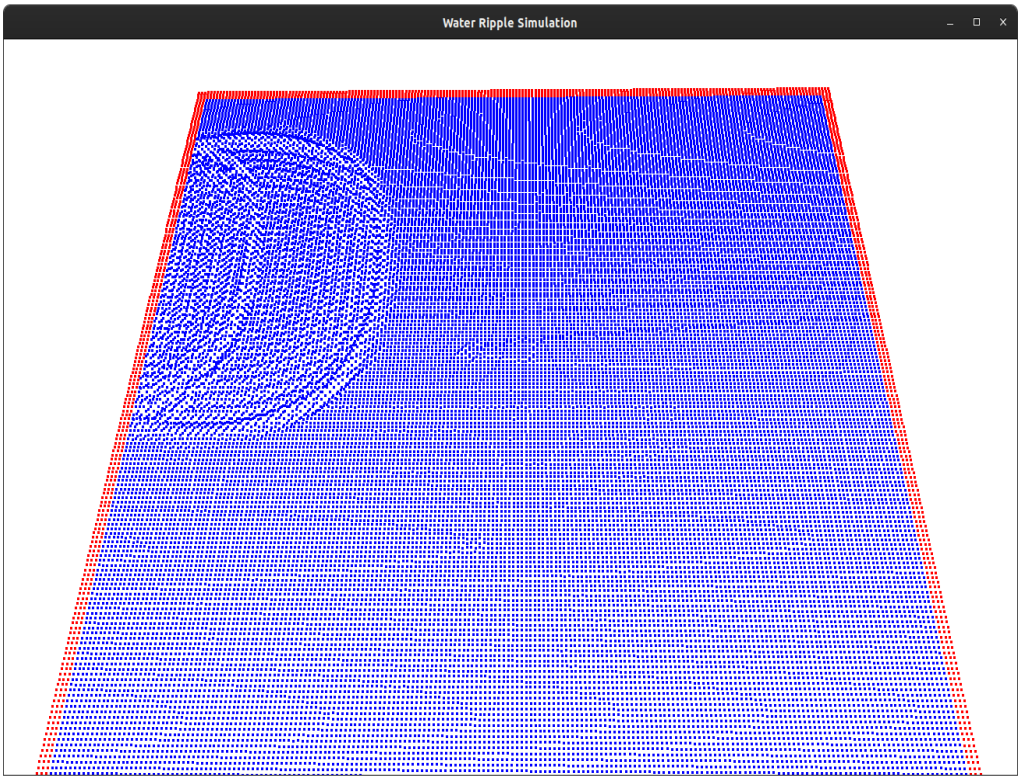




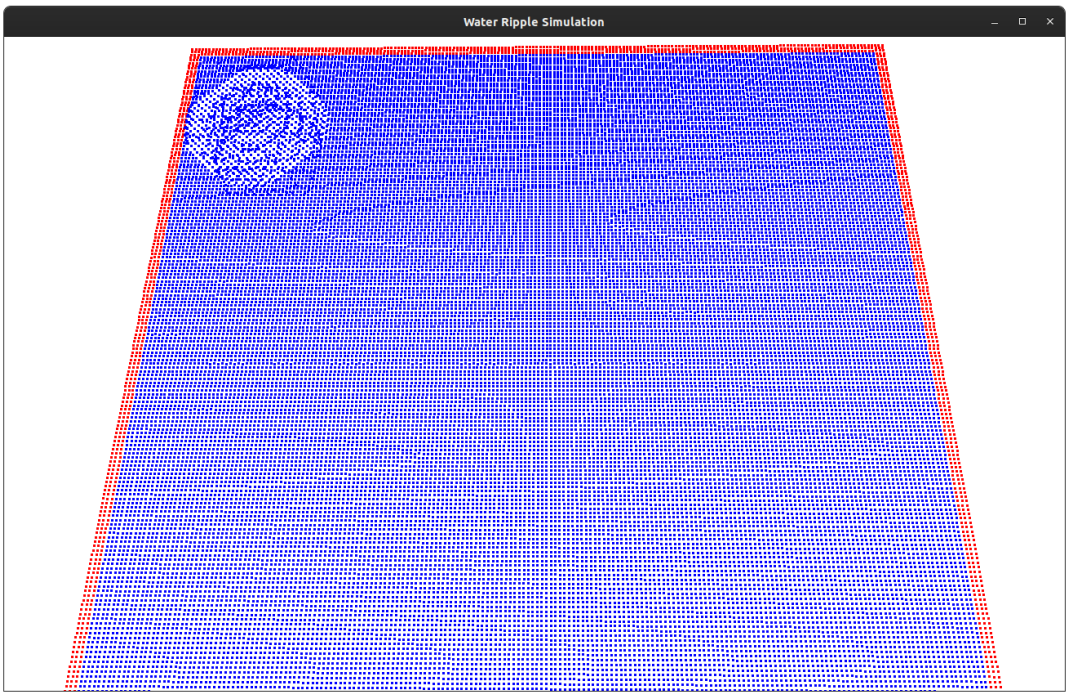


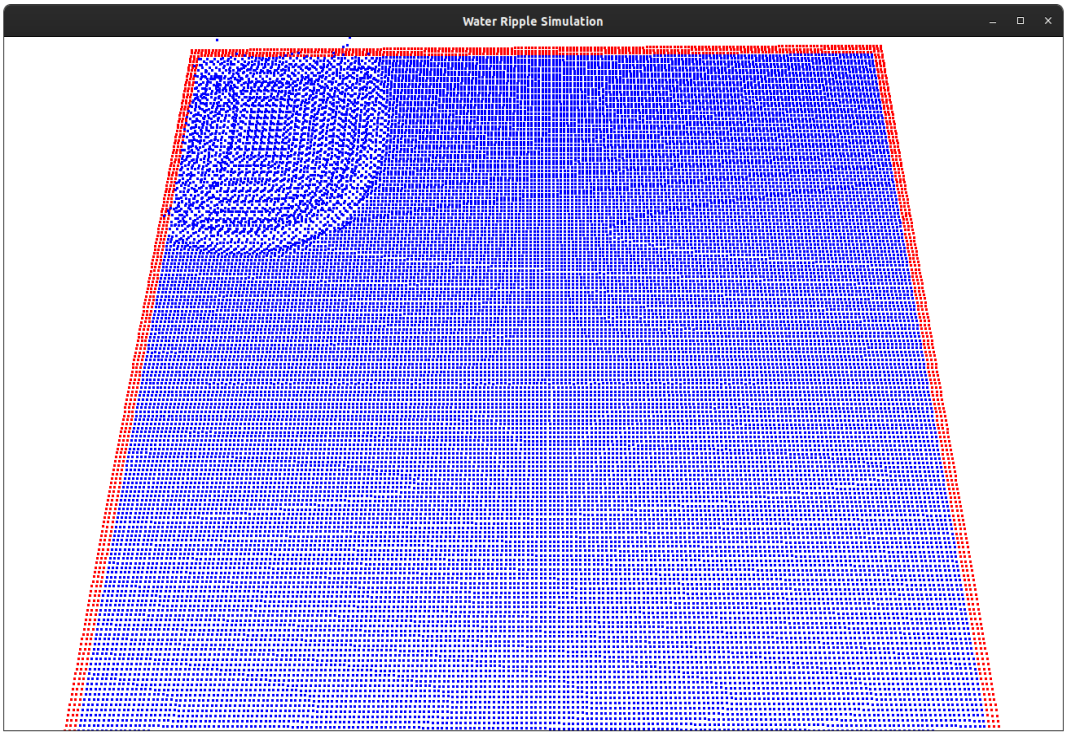
- **Ripples reflected from a wall**

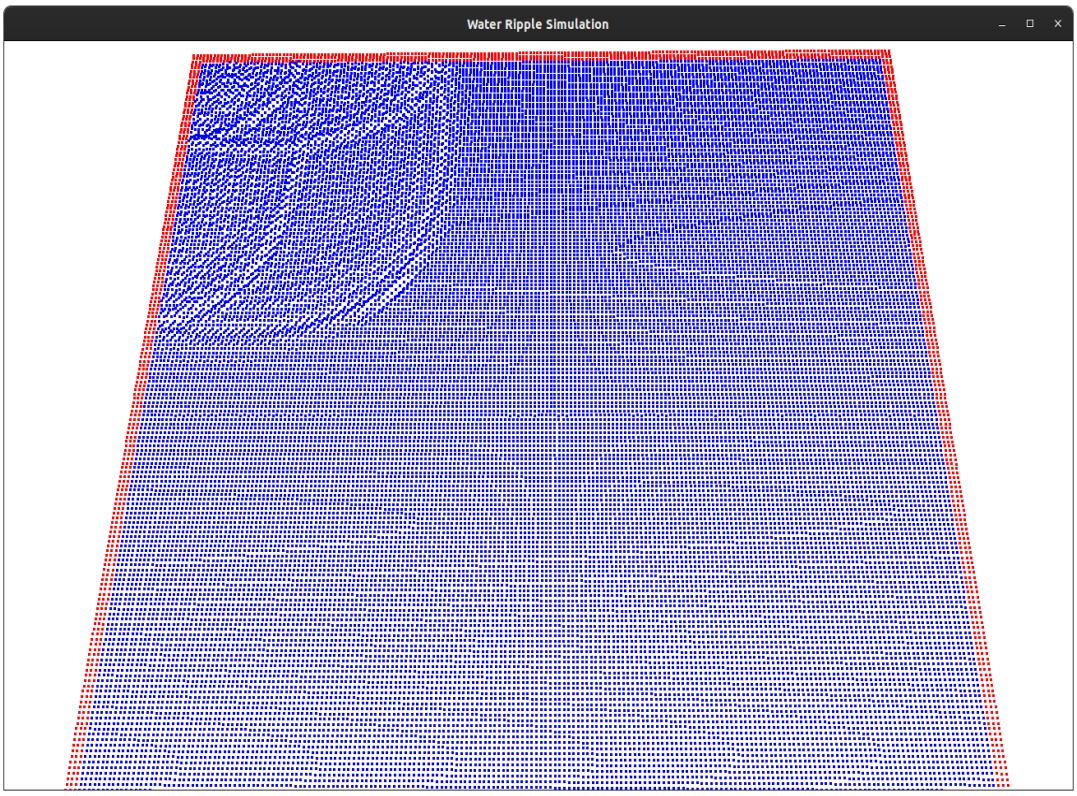




- **Ripples reflected from the corner**




Water Ripple Simulation


Water Ripple Simulation