Simulating Persisting Water Droplets on Water Surface

Yahya Alaa Mohamed Massoud

[a]*May 2018*
[b]*Computer Graphics Course*
[c]*Arab Academy for Science, Technology and Maritime Transport*
[d]*Presented to Professor Taher El Sonny*

Abstract

Computer simulation reproduces the behavior of a system using a mathematical model. It also have become a useful tool for the mathematical modeling of many natural systems in physics, chemistry and biology, and engineering. Simulation of a system is represented as the running of the system's mathematical model. It can be used to explore and gain new insights into new technology and to estimate the performance of systems too complex for analytic solutions.

Computer simulations are computer programs that can be either small, running almost instantly on small devices, or large-scale programs that run for hours or days on network-based groups of computers. The scale of events being simulated by computer simulations has far exceeded anything possible (or perhaps even imaginable) using traditional paper-and-pencil mathematical modeling.

Keywords: Computer simulation, Mathematical modeling, Physics, Computer programs.

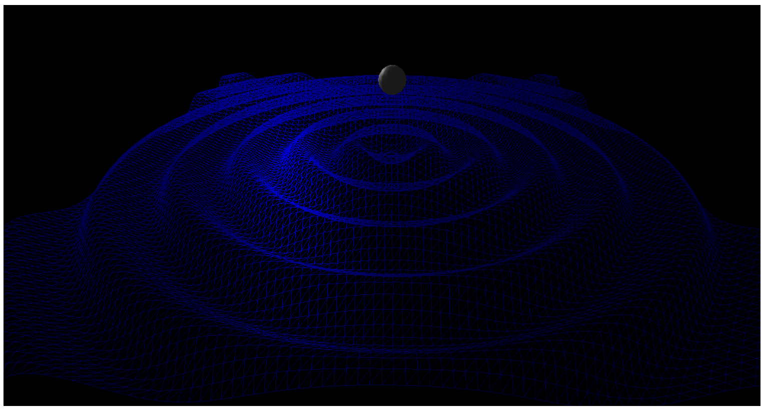


Figure 1: Screen shot of simulation program using OpenGL

*

1. Introduction

This report will discuss the simulation process for the effect of a falling droplet of water on a water surface, and explains the behavior of the water waves according to the falling speed of the water droplet on the water surface.

1.1. Water surface definition

In this simulation, I 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 represents the position of the X coordinate of the point, j represents the position of the Y coordinate of the point, and k represents the value of the Z coordinate of the point, which is in our case, the height of the vertex affected by the fall of the droplet of water on the surface composed of N * N vertices.

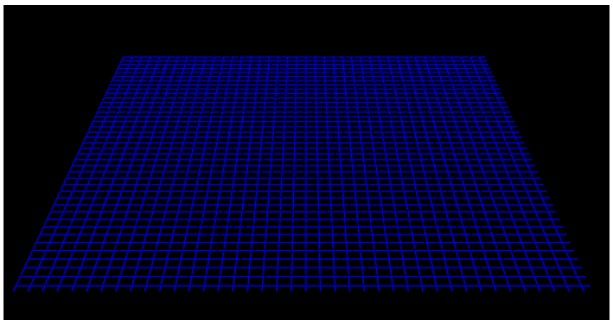


Figure 2: Water Surface

1.2. Vertex definition

For simulation purposes, a vertex's X and Y coordinates won't changes through out the whole process, because the water surface's position won't change during the simulation, but the variable parameter in this case will be the height of every vertex in the 3D space as an effect of water waves and height propagation in a function of time, which will result in a graph like the following.

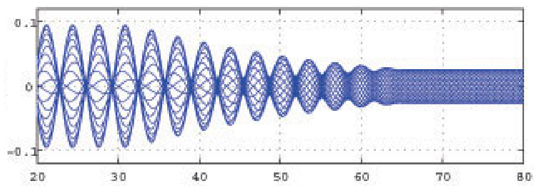


Figure 3: Graph for wave simulation

This graph simulates the variation in the vertex's height from the state when a water droplet is fallen on the surface, to the state where the droplet's effect has decayed to a constant value.

This is more like a complex definition for the simple Damped Sine Wave function which has the formula of:

$$y(t) = e^{-t}.\cos(2\pi t)$$

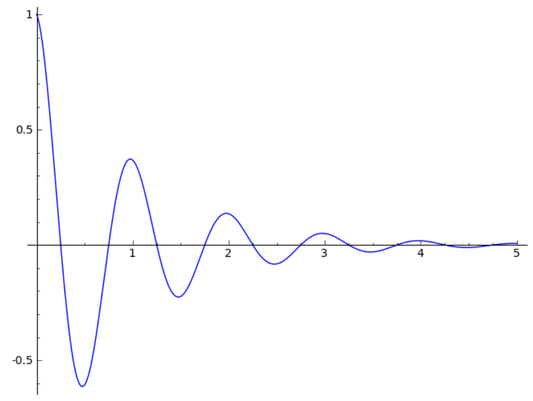


Figure 4: Damped Sine Wave

2. Mathematical Modeling

In this section I will discuss about how I used our predefined environment elements: Water Surface and Droplet Vertex, to perform a simulation for water waves caused by a falling water droplet.

2.1. Three dimensional geometric transformations

When a water droplet falls in the center of the water surface, the height of this vertex will change as an effect to the droplet's weight, and as a consequence, all the neighbours vertices' heights will change also in a function of the Damped Sine Wave function.

In order to change a vertex's Z-axis value in the 3D space, we have to perform three-dimensional geometric translation which is defined as follows:

$$P(x, y, x) \Rightarrow P'(x', y', z')$$

$$P'(x', y', z') = T(tx, ty, tz).P(x, y, z)$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

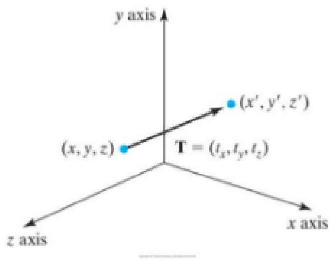


Figure 5: 3D geometrical translation

2.2. Connecting water surface vertices

In order to let the water surface feel more realistic and visual, I had to connect all the vertices with each other, to reach a grid like shape on a 2D plane in the 3D space. And that let's us see the water surface is always connected even if some vertices' Z-axis value varies in effect of a droplet fall on the surface. This is done during the simulation using a drawing line algorithm.

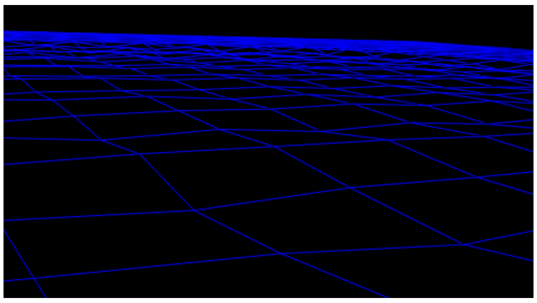


Figure 6: Connecting vertices to enhance visualization

2.3. Wave simulation algorithm

In order to compute every vertex's height in a given state during the wave simulation process, I had to develop an algorithm to do this. I have used Dynamic Programming to compute a vertex's height according to the neighbouring vertices, where the Z-axis value of a vertex in a given state is defined by:

$$D(x,y) = \frac{(D(x-1,y) + D(x+1,y) + D(x,y-1) + D(x,y+1))}{2} - D(x,y)$$

Where

$$D \Rightarrow \text{a droplet on the water surface (vertex)}$$

$$x \Rightarrow \text{X-coordinate of the point}$$

$$y \Rightarrow \text{Y-coordinate of the point}$$

After reaching this conclusion, I have added a Damping variable, which states by how much the water droplet will make echo on the water surface, so that the final formula is:

$$D(x,y) = \frac{(D(x-1,y) + D(x+1,y) + D(x,y-1) + D(x,y+1))}{2} - D(x,y)$$

$$D(x,y) = D(x,y) - \frac{D(x,y)}{DAMP}$$

Where the base state for all vertices is:

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

and for the position where the droplet has fallen is:

$$D(x,y) = -c \text{ where c is a constant}$$

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.

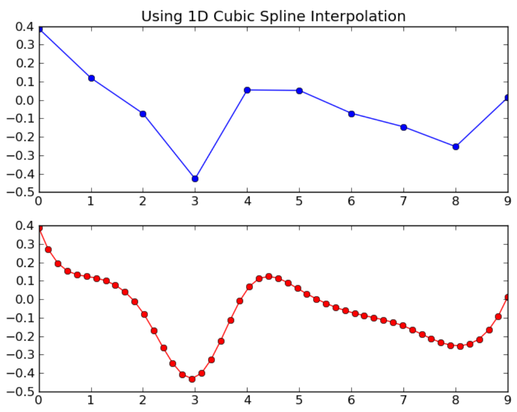
2.4. Cubic Splines

I faced a problem that connecting the vertices with straight lines while every vertex changes its height periodically will result in a non-smooth connections, which will not allow the program to simulate the scene realistically. The wave simulation should as smooth as possible.

After facing this problem, I realized that the best solution is to used splines to make the connections between vertices more smooth and curvy, which is more like the real water droplet fall effect.

In mathematics, a spline is a special function defined piece wise by polynomials. In interpolating problems, spline interpolation is often preferred to polynomial interpolation because it yields similar results, even when using low degree polynomials.

A spline aims to give a straight line smoother look by using interpolation techniques.



3. Platform Specs (Hardware / Software)

In this section, I will explain the hardware specs of my PC, as well as the software that I will use to develop the simulation program.

3.1. Hardware Specs.

- Processor: Intel Core i7 2.2GHz
- RAM: 8GB
- OS: Windows 10 64-bit
- Graphics Card: AMD Radeon

3.2. Software Specs.

- Language: C/C++
- Framework: GLUT/OpenGL v3.2
- IDE: CodeBlocks

4. Implementation Details

In this section, I will discuss some of implementation details.

4.1. User Interactions

To let the user have a close look for the simulation process, I have added a user interaction feature, which allows the user to zoom in and out from the scene and rotate around the center of the screen to be able to observe the simulation scene from all different states.

User interactions must be done using three-dimensional geometric transformations such as Scaling and Translation.

Three dimensional scaling is defined as:

$$P(x, y, z) \Rightarrow P'(x', y', z')$$

$$P'(x', y', z') = S(s_x, s_y, s_z).P(x, y, z)$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Three dimensional rotation is defined as:

$$P(x, y, z) \Rightarrow P'(x', y', z')$$

$$P'(x', y', z') = R(\theta).P(x, y, z)$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

4.2. OpenGL geometric transformation functions

OpenGL framework offer a set of helpful function to be used in the code for geometric transformations.

- void glMatrixMode(GLenum mode); - specify which matrix is the current matrix. mode - Specifies which matrix stack is the target for subsequent matrix operations.
- void glPushMatrix(void); - Make copy of current matrix stack.
- void glPopMatrix(void); - Pops matrix from top of stack.
- void glTranslatef(GLfloat x, GLfloat y, GLfloat z); - produces a translation by x y z.
- void glScalef(GLfloat x, GLfloat y, GLfloat z); - produces a scaled version of x y z.

5. Experimentation

This section will include a set of my experimentation and trials to make the simulation algorithm more realistic.

5.1. Trial One

In my first trial, I tried to draw the water surface as a set of unconnected vertices in the 3D space, but the simulation was not realistic, it just showed some points move in a 3D space.

5.2. Trial Two

In order to make a more realistic simulation, I had to introduce connections between vertices, the edges were straight lines, I connected every set of horizontal N vertices with each other, then connected every set of vertical N vertices with each other, which resulted in a grid view in the 3D space.

This trial was not very successful as there was no smoothness in the simulation scene, the connections between vertices was very sharp, which did not simulate the water waves in the right way.

5.3. Trial Three

My last trial was the last, I solved the problem of edges sharpness using splines connecting between all the surface vertices.

This resulted in the best simulation scene could be reached, but still not realistic. The problem was that all vertices were connected using splines, but all the spline lines pass through the vertices, which introduced smoothness but still not realistic as using B-splines, where the splines does not mandatory pass through every vertex in the surface plane.