Simulating and rendering waves on water surfaces

Field of study: Media informatics
Semester: 4

Date: Juni 22, 2018

Authors:
Fabian Niehaus
Tuyet Nguyen

Contents

List of Figures

1 Abstract

In this project we will build a C++ application to sumulating and rendering waves on water surfaces. It will also be able toâĂę. We will use QT Creator for programming, QT for window management and OpenGL for 3D rendering.

2 The function will be used

3 Realisation

3.1 Creating an interface with QT

First off, we create a new QT widget application. This allows us to use QT Creators design feature to set up our application's interface. A new QOpenGLWidget is placed and will be used as a placeholder for a new custom class inheriting QOpenGLWidgets functionality. This class, called OGLWidget, needs to implement the following methods: initializeGL (for setting up OpenGL), paintGL (for doing the actual rendering), resizeGL (for handling resizes of the display window). Additionally, the functions stepAnimation SetMaterialColor and InitLightingAndProjection[1] are used.

3.2 Creating the data structure

The data structure is separated in different classes.
The basic class "Waves" contains the contains the information of the waves like sine waves, Height-Field, coordinate, direction vector, phase velocity, the frequency and the wave number.
The class "Wavesurface" contains the wavesfunction . Logic and data regarding the computation of quad meshes is stored in a seperate class, as are Bezier surfaces and rotational sweep surfaces.
In order to allow for easier use of a two dimensional matrix of vertices, a wrapper class containing a two dimensional vector of vertices is introduces.

3.2.1 Quad mesh

After creating the required data structure, a method to make a mesh for the waves.

3.3 Rendering as a wireframe

Depending on the desired way of rendering the object, different draw methods are implemented. These methods are then being called from the paintGL() function.

3.3.1 Rendering as a solid

After drawing the object as a wireframe we want to draw it as a solid cube with lighting. This is being achieved in the method drawQuads() which once again iterates over the list of quads. This time using GL_Quads, the four vertices of a quad are connected

[1] Taken from Prof. Dr. Martin Hering-Bertrams OpenGL_Example

and the area inbetween is filled. The normal vector for this is calculated using the cross product of the two diagonals vectors.

3.4 Calculation the vertex valence

3.5 Determining adjacent quads

3.6 Printing the data

3.7 Catmull-Clark-Subdivision

3.7.1 Calculating the Face-Mask

3.7.2 Calculating the Edge-Masks

3.7.3 Calculating the Vertex-Masks

3.7.4 Reconnecting the mesh

3.8 Bezier surface calculation