# Physically-based simulation of deformable surfaces in gaming and animation applications. Water surface animation

**Alexander Markhichev**
**Alexei Soupikov**

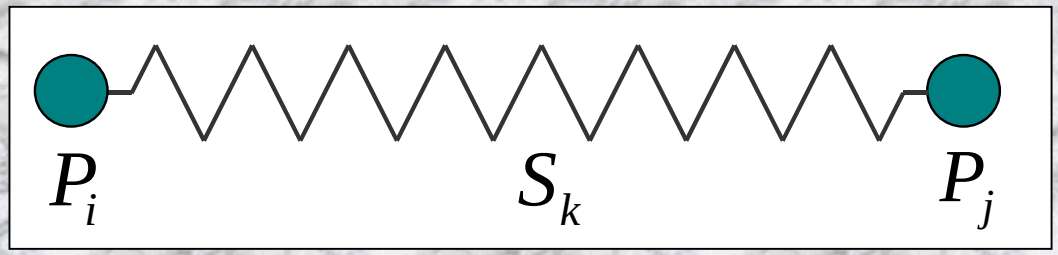
International Conference Graphicon 1999,
Moscow, Russia, http://www.graphicon.ru/

# Water surface animation

- Deformable surface model
  - Deformable surface model element
  - Deformable surface model structure
  - Model motion equation
  - Numerical integration of equation
  - Default parameters
- Polygonal surface simplification
- Increasing view realizm



# Deformable surface model element




**Each node $P_i$ has following parameters:**

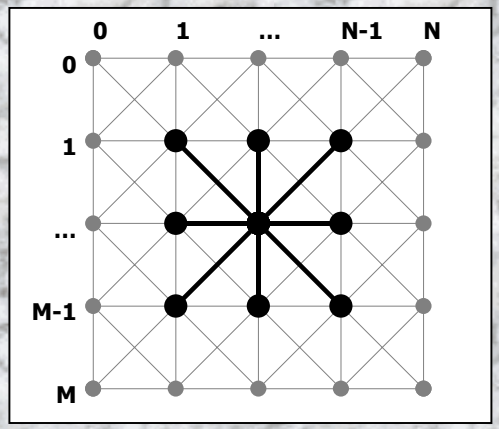
- nodal mass;
- position and velocity;
- sum of applied forces;

**Spring $S_k$ connecting nodes $P_i$ and $P_j$ has following parameters:**

- rest length;
- current length;
- spring coefficient;

# Deformable surface model structure

**Modeled surface represents M*N regular grid**




- Elemental nodes are placed in grid nodes;
- The springs connecting elemental nodes are placed on grid square edges and diagonals;
- In non-boundary case node connected with springs to 8 incident nodes;

# Model motion equation

The surface motion equation is a system of second order ODE. Each ODE in the system is motion equation for separate node $\mathbf{P_{ij}}$ :

$$m_{ij}\frac{d^2\bar{x}_{ij}}{dt^2}+\gamma_{ij}\frac{d\bar{x}_{ij}}{dt}+\bar{f}_{ij}=\bar{f}_{ij}^{ext},$$

where

$m_{ij}$ - nodal mass;

$\bar{f}_{ij}$ - sum of forces induced by attached springs;

$\bar{f}_{ij}^{ext}$ - sum of external forces;

$\gamma_{ij}$ - viscosity coefficient;



# Numerical integration of equation

Each equation in the system is integrated using explicit Euler method

$$\bar{a}_{ij}(t) = \frac{1}{m_{ij}}\left(\bar{f}_{ij}^{ext}(t) - \gamma_{ij}\bar{v}_{ij}(t) - \bar{f}_{ij}(t)\right)$$

$$\bar{v}_{ij}(t+\Delta t) = \bar{v}_{ij}(t) + \bar{a}_{ij}(t)\Delta t$$

$$\bar{x}_{ij}(t+\Delta t) = \bar{x}_{ij}(t) + \bar{v}_{ij}(t+\Delta t)\Delta t$$

Benefits of using this integration method are simplicity and high performance though numerical instability is possible when integration timestep becomes large.



# Default parameter

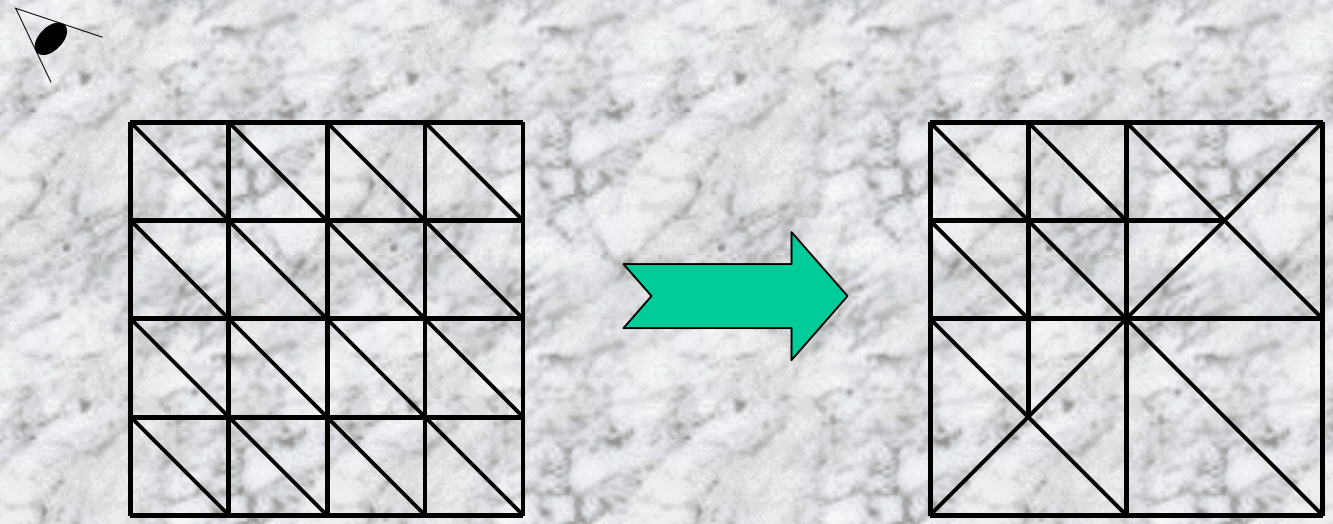
The system parameters we tuned to acquire visual correspondance to water surface motion and critical integration timestep

| | |
|---|---|
| Grid step, $h$ | 1.0 |
| Nodal mass, $m_{ij}$ | 4.0 |
| Spring coefficient, $c_k$ | 7.0 |
| Viscosity coefficient, $\gamma_{ij}$ | 0.05 |
| Integration timestep, $\Delta t$ | 0.4 |



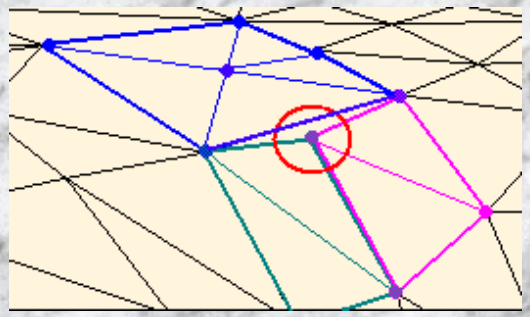
# Polygonal surface simplification

To increase rendering performance we use view-dependent algorithm of polygonal surface generation from nodal points. The LOD number is introduced as function of distance to viewpoint.

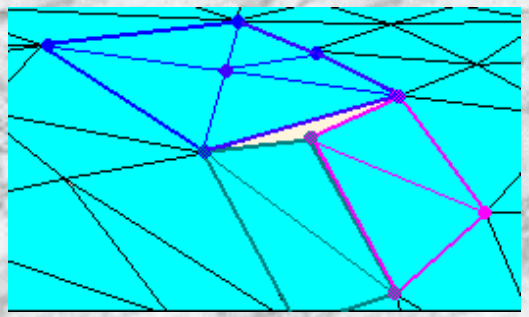


# Polygonal surface simplification(2)

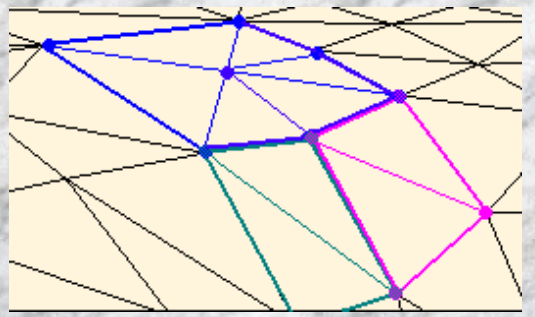
The main problem is to keep resulting mesh connected on LOD boundaries



Connectivity violation

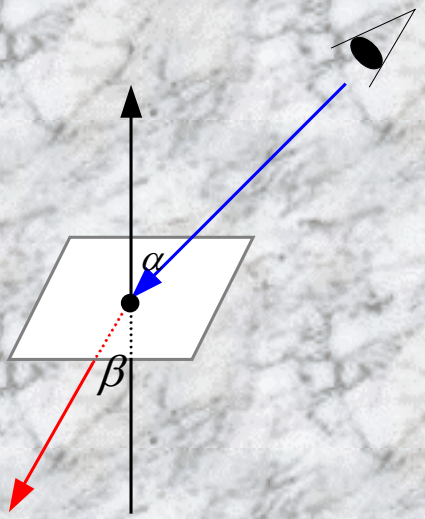


Mesh with disconnectivity when rendered



Adjust required

# Bottom view refraction




$$\frac{\sin(\alpha)}{\sin(\beta)} = const$$

**To acquire refracted bottom view:**

- Refracted ray direction is calculated from view direction and normal direction at the node
- Position of point where ray hits the bottom is calculated from node position and refracted ray direction
- Having surface node mapping to point on bottom one might map surface node to any bottom point parameters. For instance map surface node to bottom texture coordinates and render the surface with texture of bottom.