

Assignment #3

- Group 3 - presentation
- Group 1 - opponents

Water simulation

- We will implement a simple water simulation.
 - A 2d grid based simulation with direct mapping from to the simulation grid points
 - Each grid encodes the height of the water in that grid point.
 - We will iteratively do the computational kernel to create animated water

1. Initialize the PBuffer with water heights
2. Apply water kernel
3. Visualize the water height as a texture – create a fp to clamp the values.

$$\frac{\partial^2 y}{\partial t^2} = c^2 \left(\frac{\partial^2 y}{\partial x^2} + \frac{\partial^2 y}{\partial z^2} \right)$$

$$\left(\frac{\partial^2 y}{\partial x^2} + \frac{\partial^2 y}{\partial z^2} \right) \approx \left[\sum_{4\text{-neighbors}} y_{i,j} \right] - 4 * y_{0,0}$$

Water kernel

Acceleration a

$$a_{i,j}(n) = h_{i-1,j}(n-1) + h_{i+1,j}(n-1) + h_{i,j-1}(n-1) + h_{i,j+1}(n-1) - 4h_{i,j}(n-1)$$

Velocity v

$$v_{i,j}(n) = v_{i,j}(n-1) + \Delta t * a_{i,j}(n)$$

Height h

$$v_{i,j}(n) = (1.0 - \text{damping}) * v_{i,j}(n)$$

$$h_{i,j}(n) = h_{i,j}(n-1) + \Delta t * v_{i,j}(n)$$

- You need:
 - Previous heights
 - Previous velocities

(this is actually Euler integration)

Optional

- More elaborate visualizations
 - Create a model of vertices that animates based on the water calculation
 - Do read-back of the Pbuffer
 - Use render-to-vertex functionality to render
 - Potentially time-consuming programming !
 - Visualize the water animation as bump-maps
 - Use the bump mapping from assignment #1
 - Use parallax mapping to visualize !

