USC Viterbi
School of Engineering

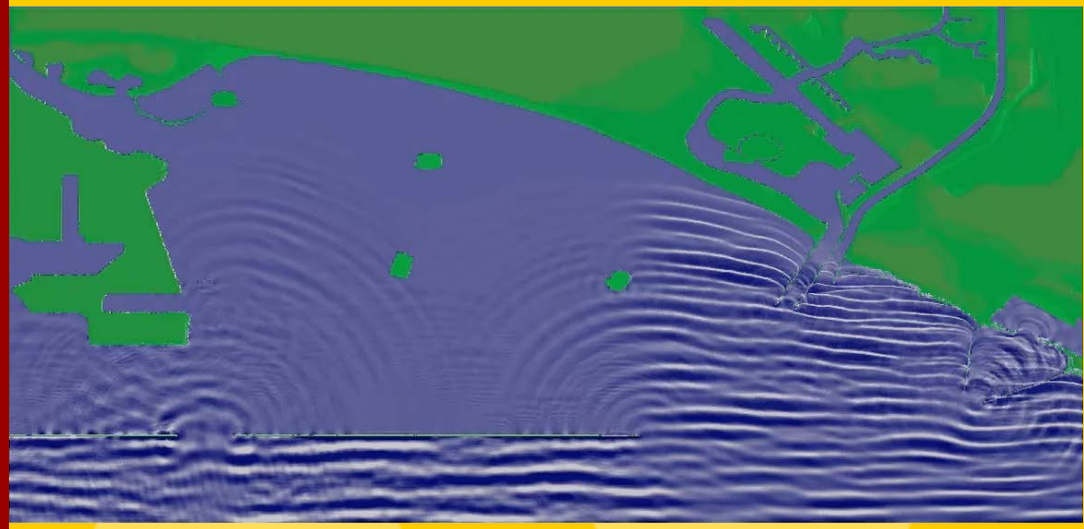
USC

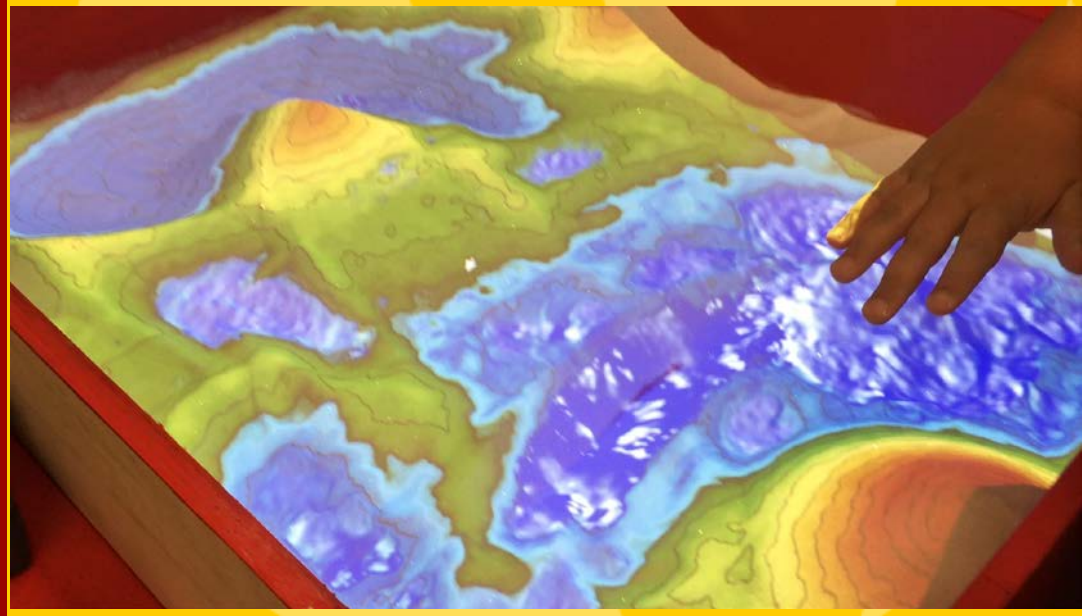
# Opportunities for Interactive, Physics-Driven Wave Simulation

**Dr. Patrick J. Lynett**

**Sasan Tavakkol**

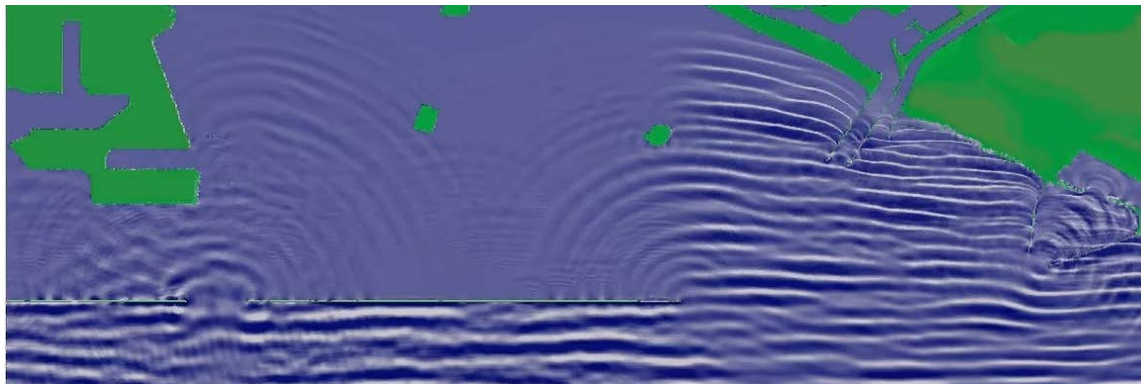
*University of Southern California*





# Interactive, Physics-Driven Wave Simulation

- **Focus here on phase-resolving wave models (shallow water, *Boussinesq-type*)**
- **Widespread usage in scientific applications, more limited in engineering and operations usage**
  - Why?
    - Time → computational cost can be high, often hours - days for large domains ---- can the project support it?
    - Input Uncertainty → deterministic modeling with lots of imprecision / uncertainty in bathymetry, incident wave spectrum, etc. --- can the user tell the result is “better” than a result from a “simpler” model?
      - But… this is related to the cost of the model…
- **What if we could run these models faster than real time, with ordinary hardware? If real-time, can we “interact” with the simulation?**



USC

# Interactive, Physics-Driven Wave Simulation

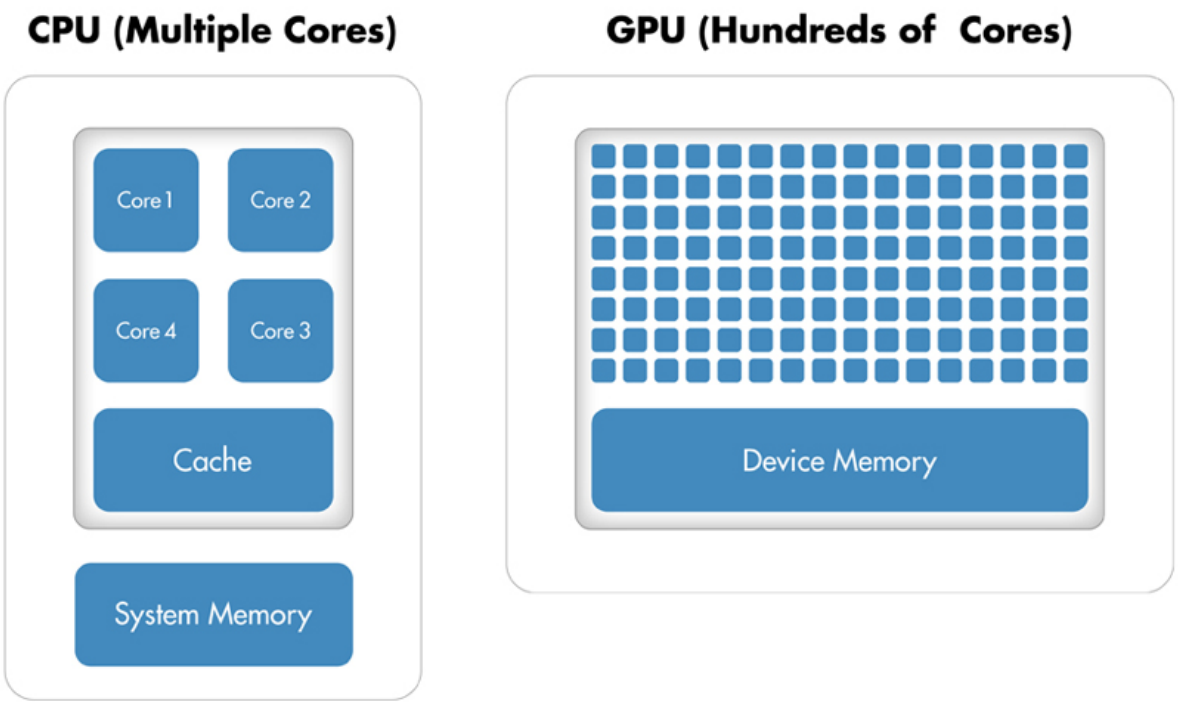
- **Our approach**
  - Implement a Boussinesq-type solver on GPU
  - Design for visualization and interactivity

- **Our equations**
  - Extended Boussinesq equations of Madsen et al (1991, 1992)

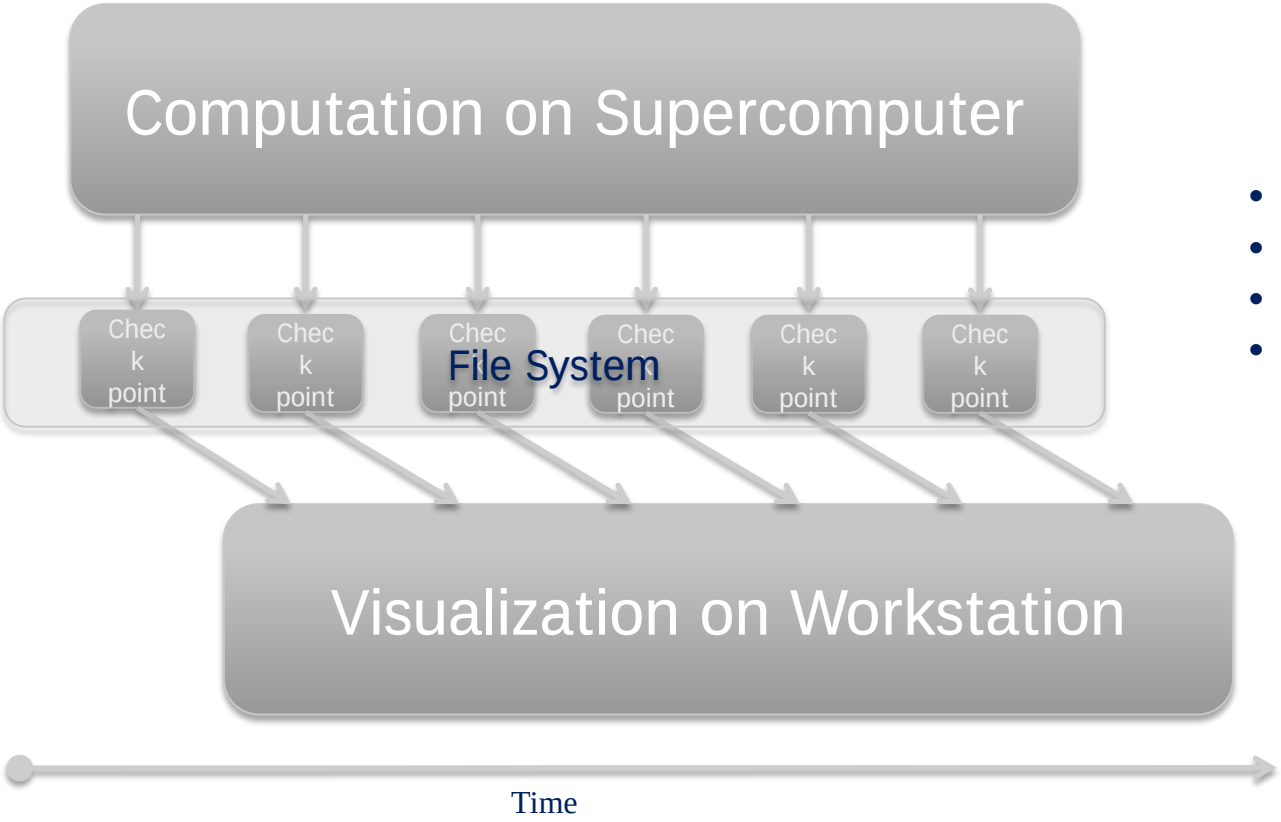
- **Numerical Solution**
  - Finite volume approach in space
  - Predictor-corrector in time
  - Breaking approximated via numerics, use of slope and flux limiters

USC

# CPU vs. GPU


CPU (Multiple Cores)
Core 1
Core 2
Core 4
Core 3
Cache
System Memory
GPU (Hundreds of Cores)
Device Memory

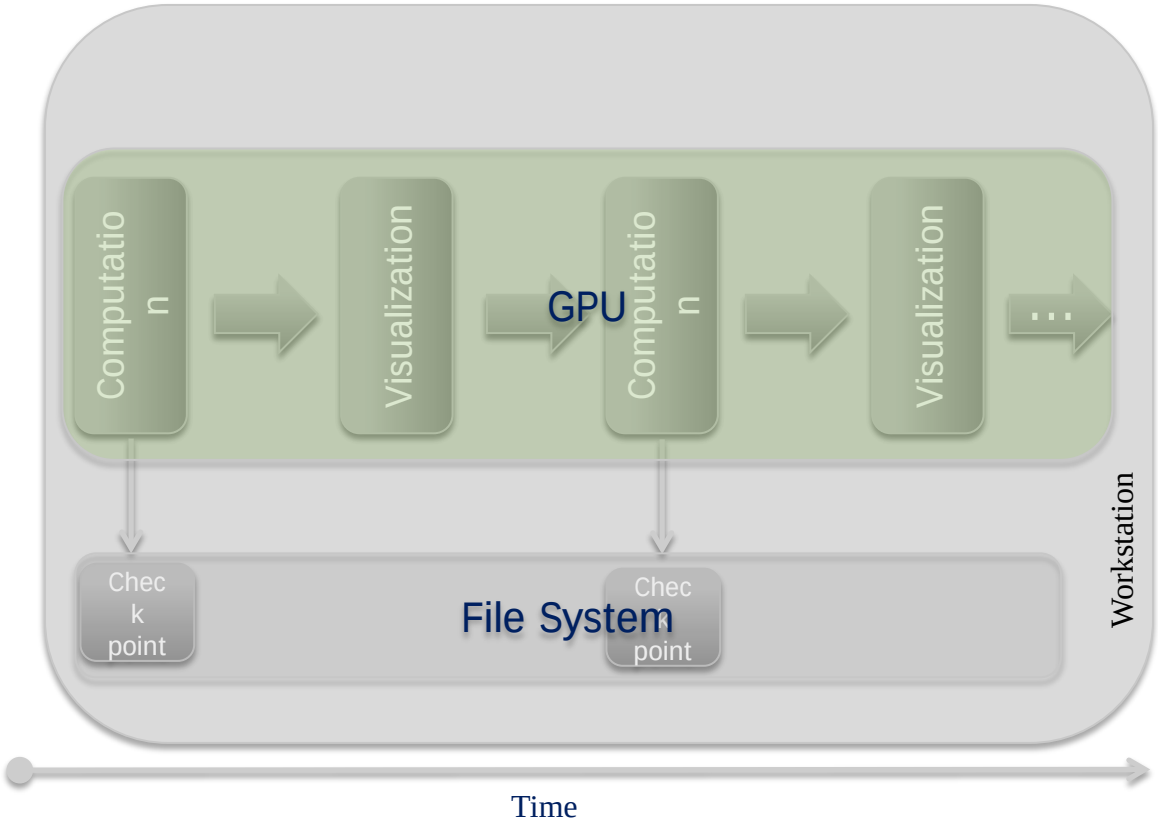
# Traditional Visualization on Supercomputers




- High cost
- I/o is the bottleneck
- Visualization is delayed
- Lack of interactivity

USC

# Simulation and Visualization on GPU




- Low cost
- I/o is optional
- Visualization is Real-time
- Interactivity is possible

# Shader Programming and CUDA

## Shader Programming : (on OpenGL, DirectX)

- Reformulating problem in terms of graphics primitives (textures, pixels, etc.).
- Longer learning curve
- Visualization is done via direct access to graphical libraries
- Runs on many devices
- Easy to generate stand-alone software

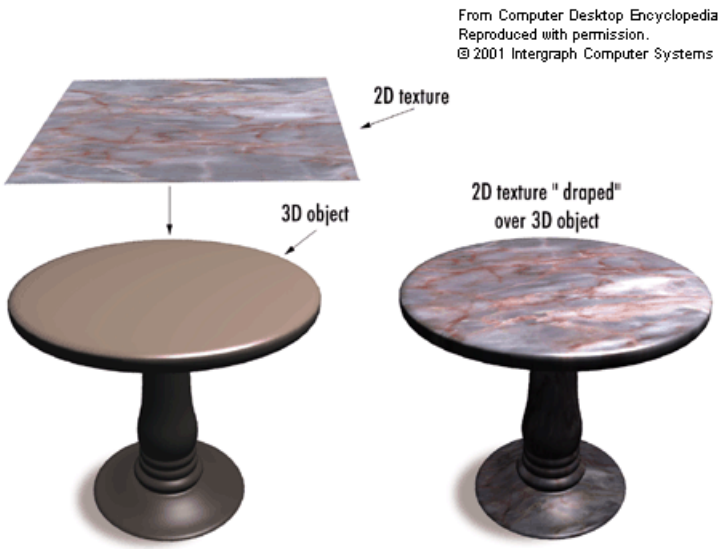
## CUDA:

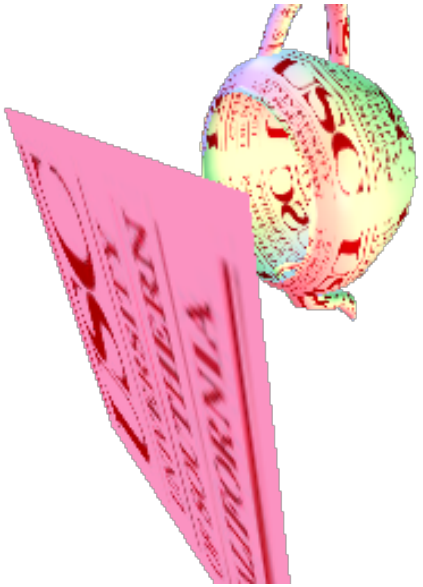
- Ignoring the underlying graphical concepts.
- Shorter learning curve
- Visualization is done through a graphical API such as OpenGL
- Runs only on CUDA enabled GPU's from NVIDIA
- Depended on CUDA library.

USC

# Textures in Graphic Rendering

- Texture mapping is wrapping a bitmap image around a 3D model in order to add details and surface texture to the model.
- We use textures as a data structure to store flow parameters and to perform the computations.



# Reformulating problem in terms of graphics primitives

- Data must be stored within textures. Each texture is a matrix of texels*.
- Each texel, generally, contains 4 scalar variables: R, G, B, A.
- In a real texture image, RGB values represent the intensity of red, green, and blue color of the texel, and A (alpha) represent its opacity.
- We use these variables to store flow parameters.

Examples:

```
Texture2D<float4> txState : register( t0 );
// .r =  h
// .g = hu
// .b = hv
// .a = (unused)
```

```
Texture2D<float4> txH : register( t0 );
// .r = North
// .g = East
// .b = South
// .a = West
```

```
Texture2D<float4> txXFlux : register( t2 );
// .r = h-flux (x direction)
// .g = hu-flux (x direction)
// .b = hv-flux (x direction)
// .a = (unused)
```

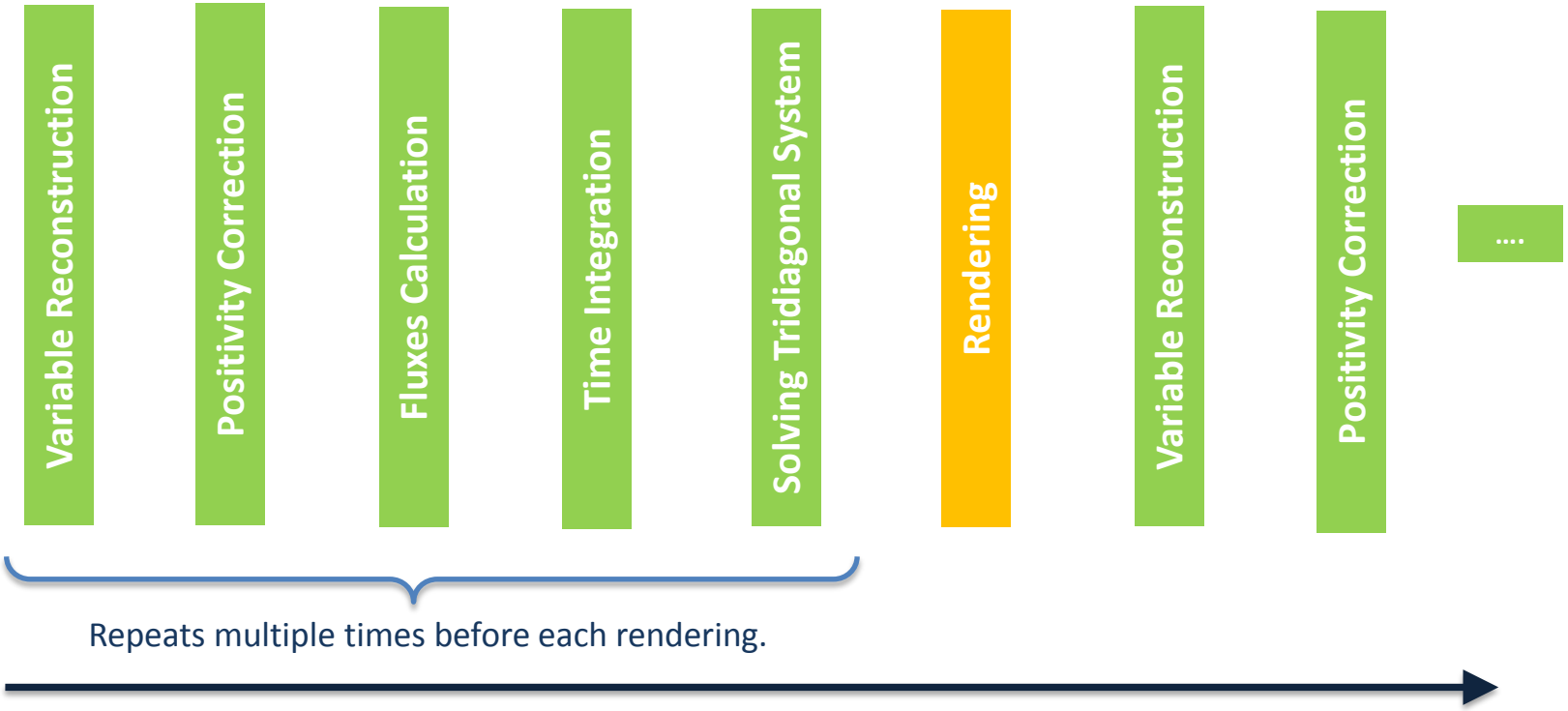
- Computations are done for each texel in parallel.

```
CalcUV(txState.Load(idx).r, txState.Load(idx).g, txState.Load(idx).b, u,v);
// CalcUV(          h          ,          hu          ,          hv          , u,v);
```

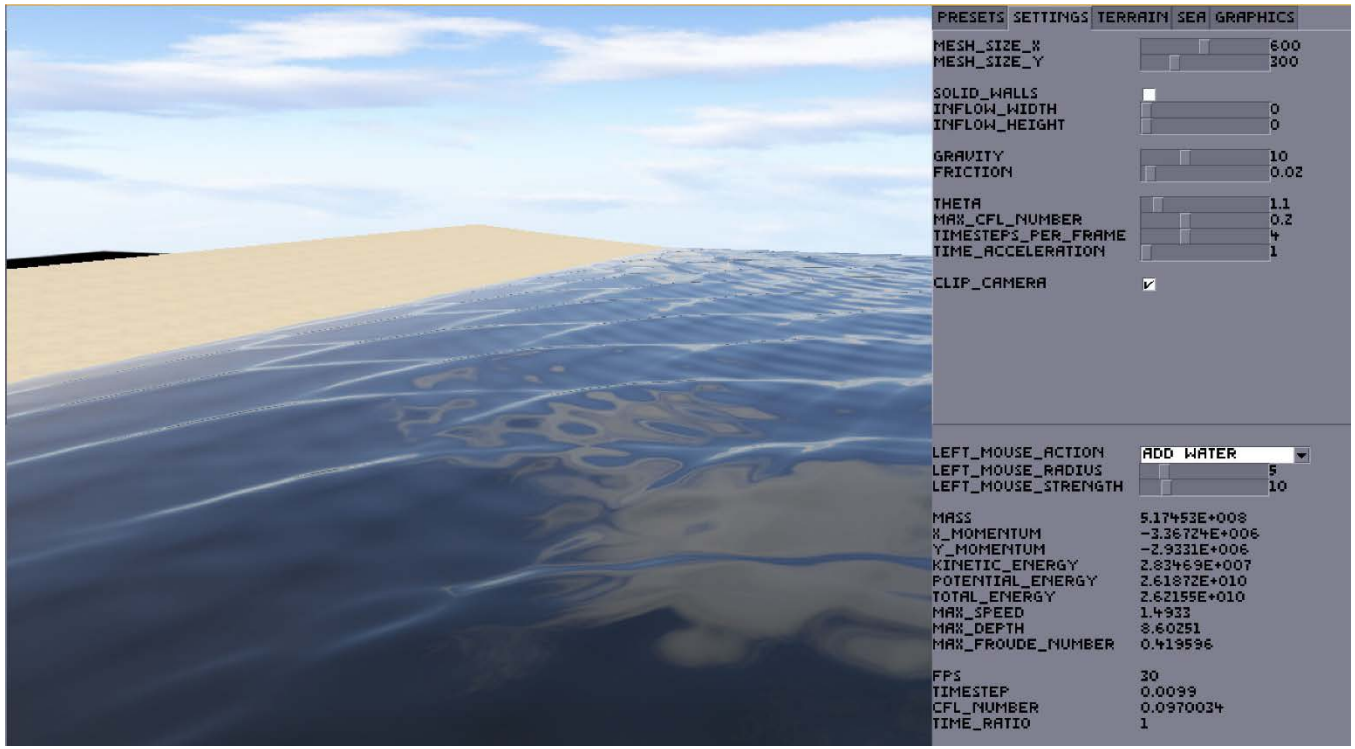
* Texels are texture elements as pixels are image elements. The renderer maps texels to appropriate pixels in the output picture.

# Simulation flow

Each step is done through passing a shader and appropriate textures to the GPU



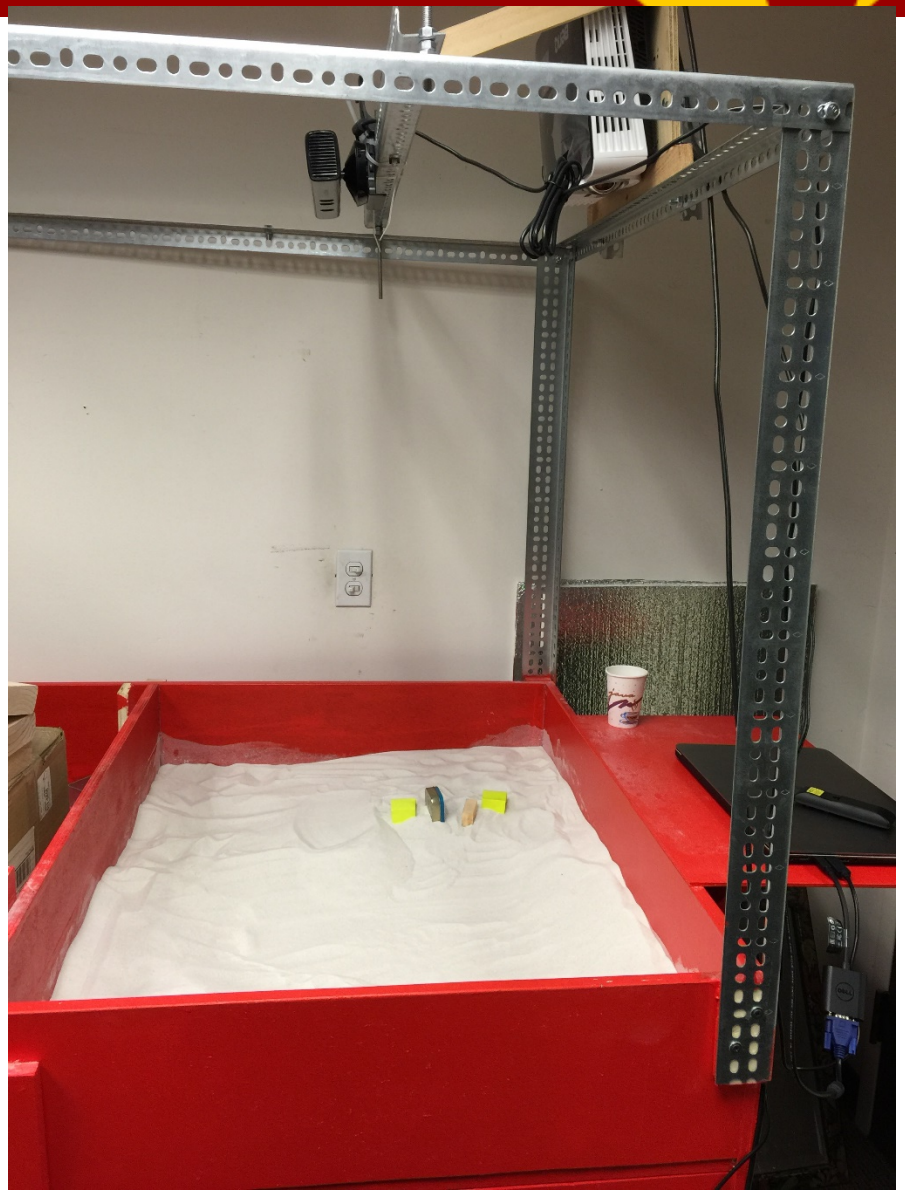
# Boussinesq Modeling Software




Shader Programming using DirectX10 and HLSL
Runs on any Directx10 compatible device.

# Interactive, Physics-Driven Wave Simulation - Opportunities

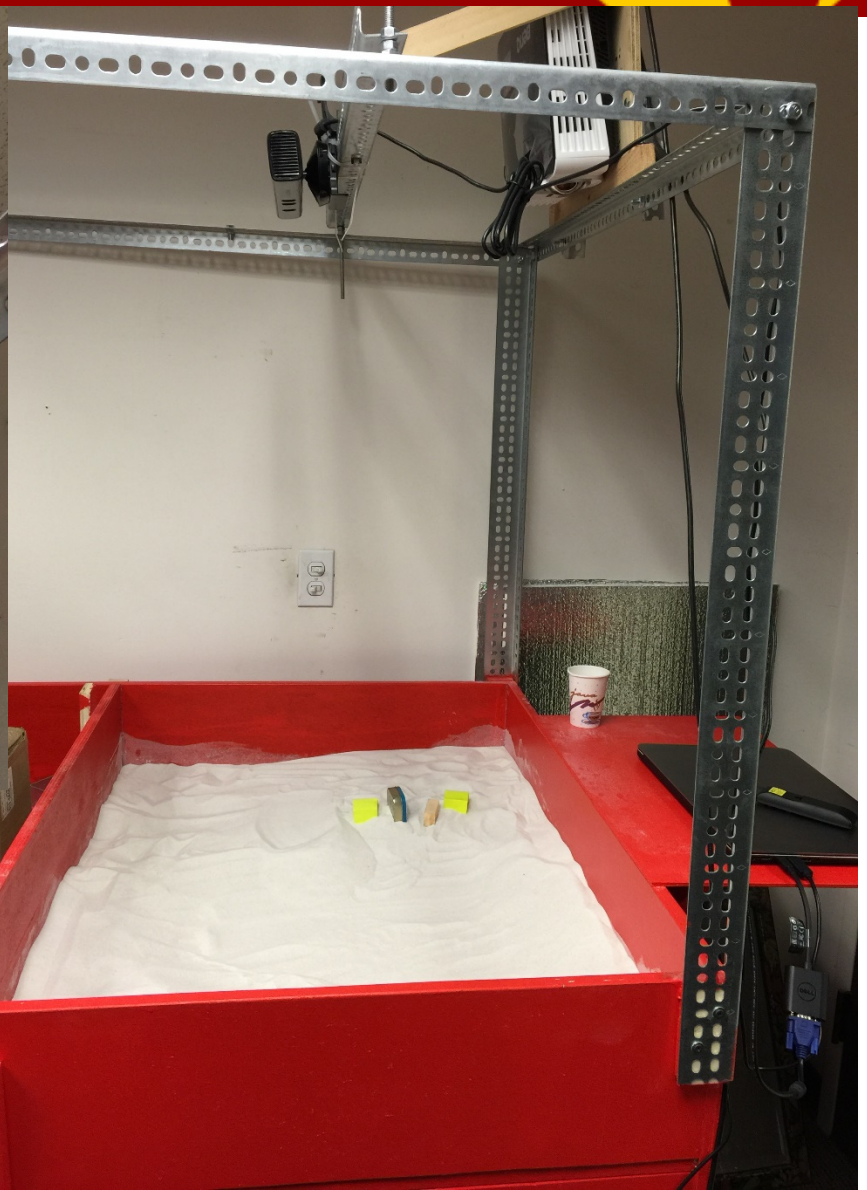
- Faster than real-time simulation of Boussinesq-type models:
  - Interactive, tablet based simulation
  - Wave-by-wave resolving nowcasts
  - Collaborative operations planning using “real” physics engine
  - Outreach & Education



Augmented Reality Sandbox - Oliver Kreylos
http://idav.ucdavis.edu/~okreylos/ResDev/SARndbox/

USC

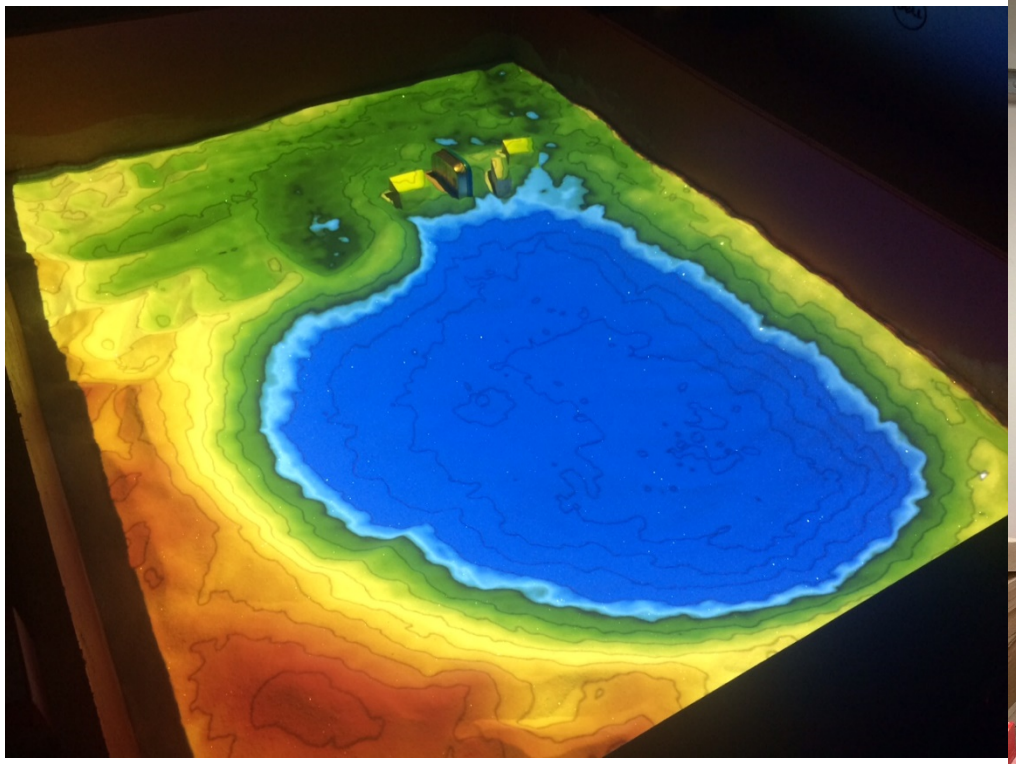
# Interactive, Physics-Driven Wave Simulation - Opportunities

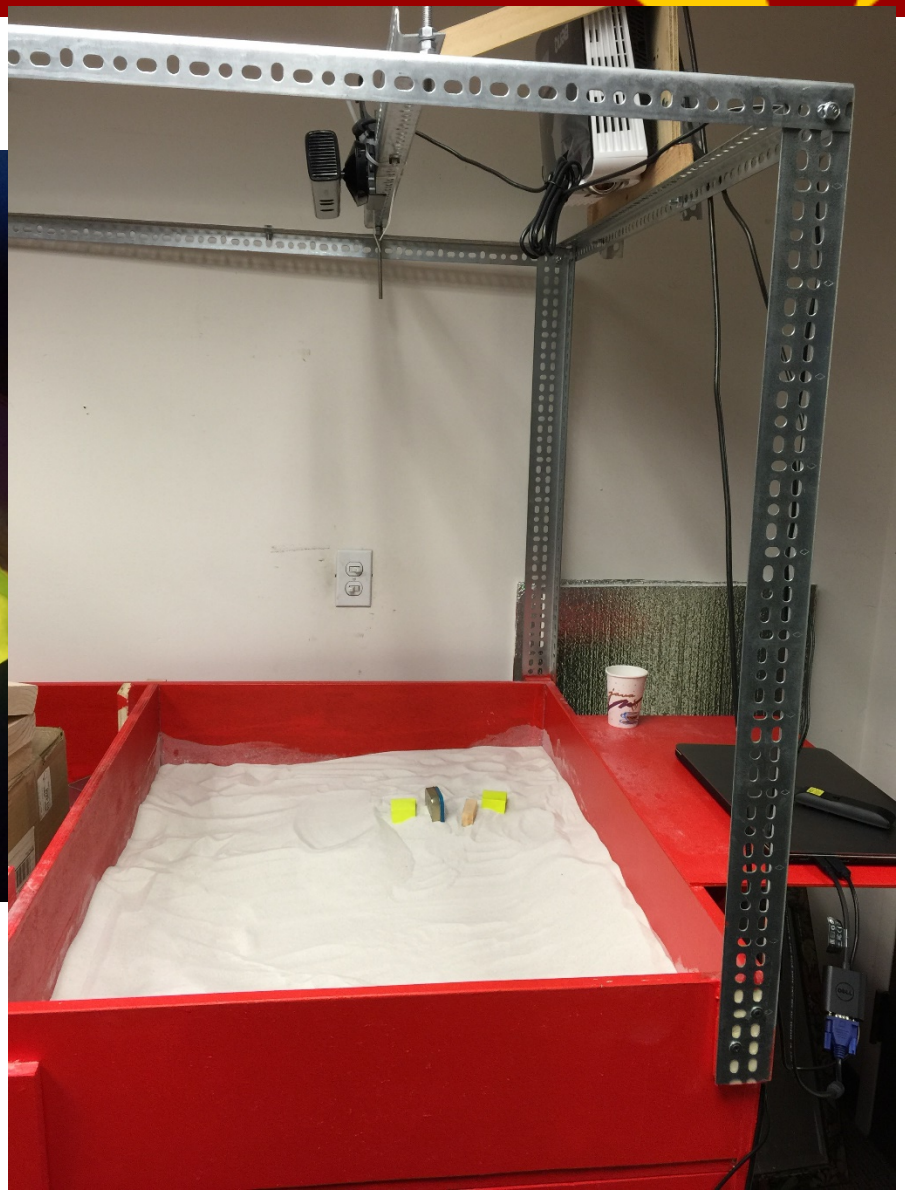


USC

Augmented Reality Sandbox - Oliver Kreylos
http://idav.ucdavis.edu/~okreylos/ResDev/SARndbox/

# Interactive, Physics-Driven Wave Simulation - Opportunities

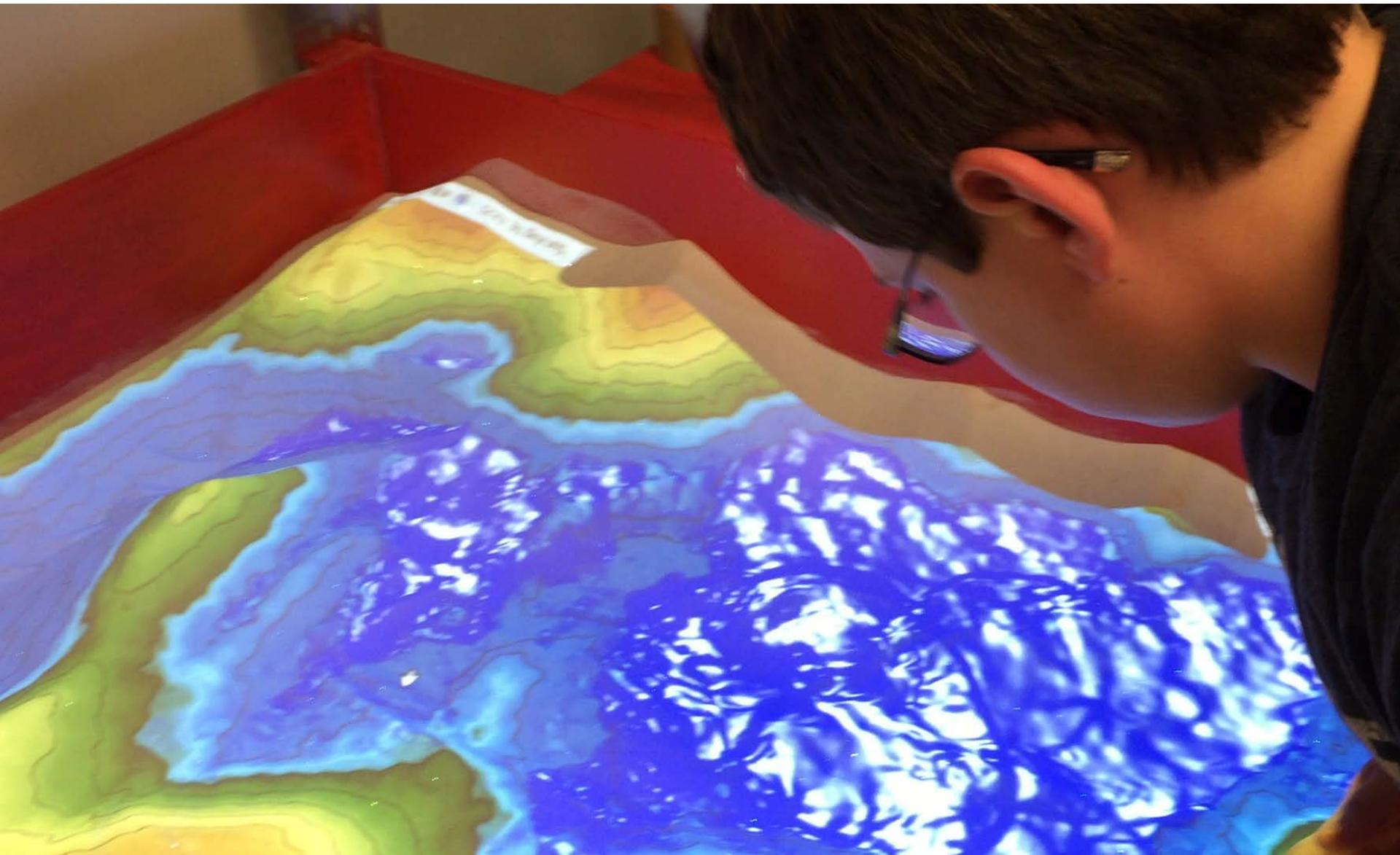




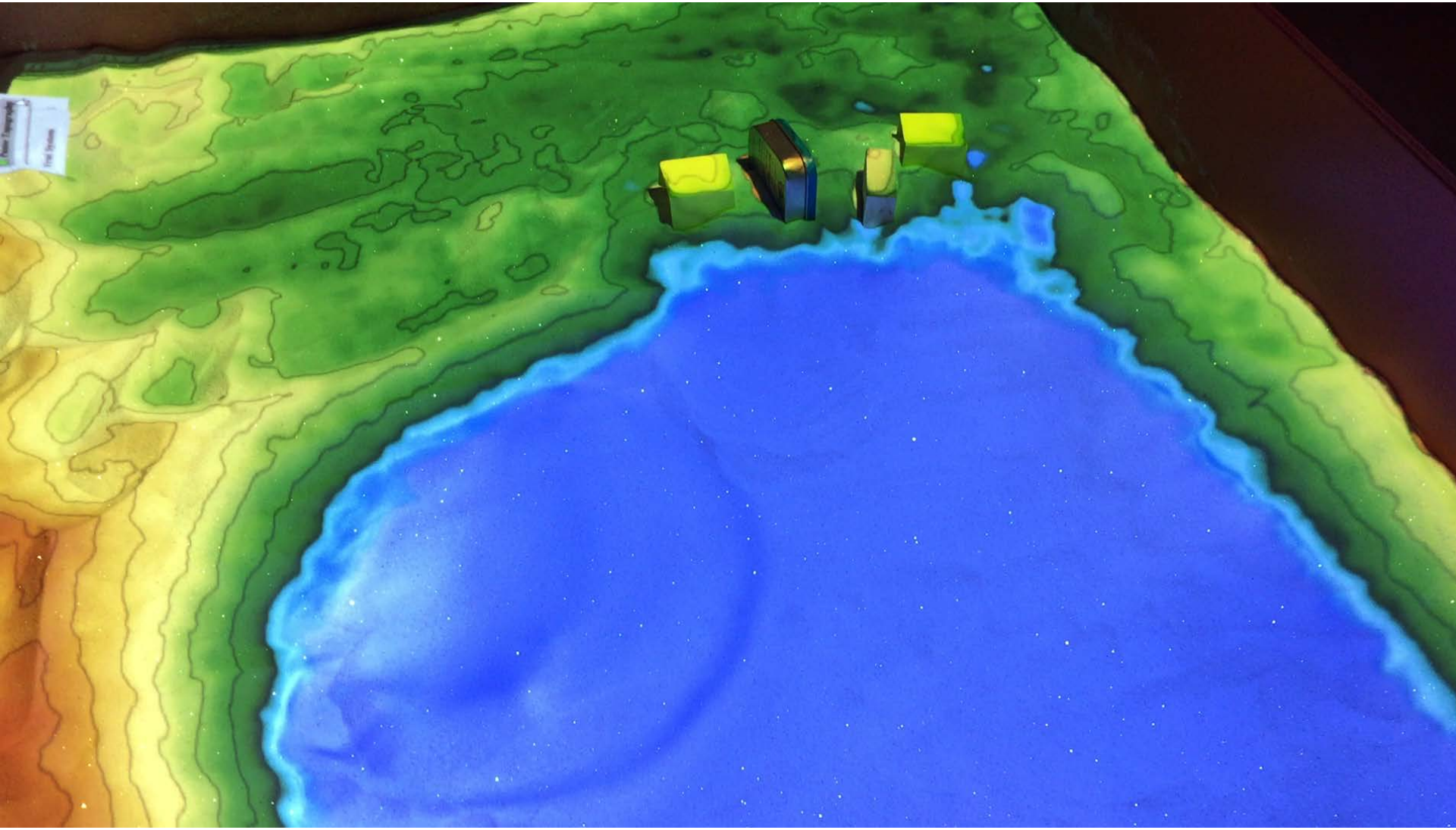
USC

Augmented Reality Sandbox - Oliver Kreylos
http://idav.ucdavis.edu/~okreylos/ResDev/SARndbox/

# Interactive, Physics-Driven Wave Simulation - Opportunities

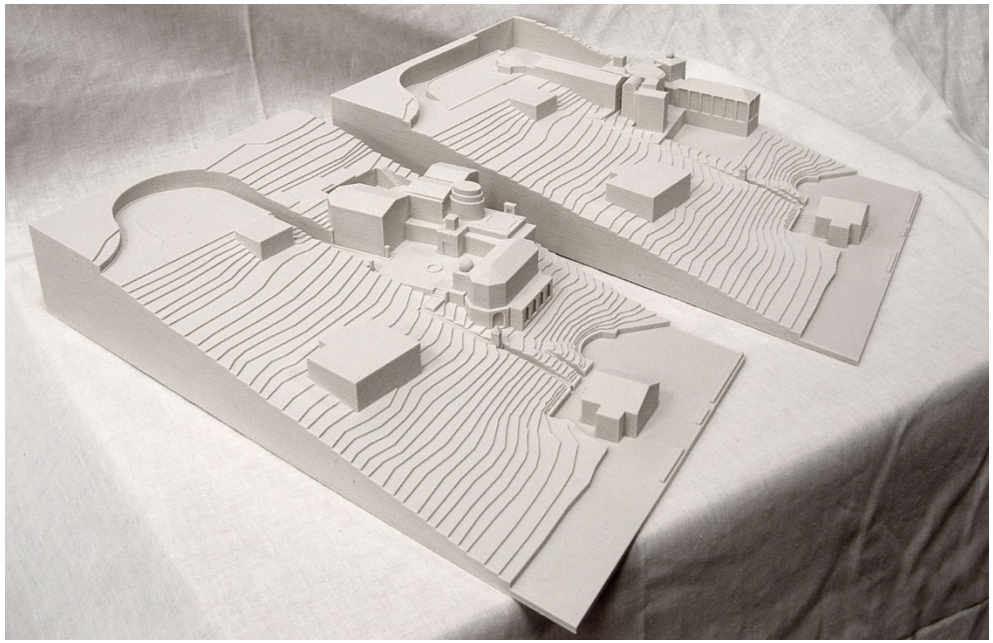


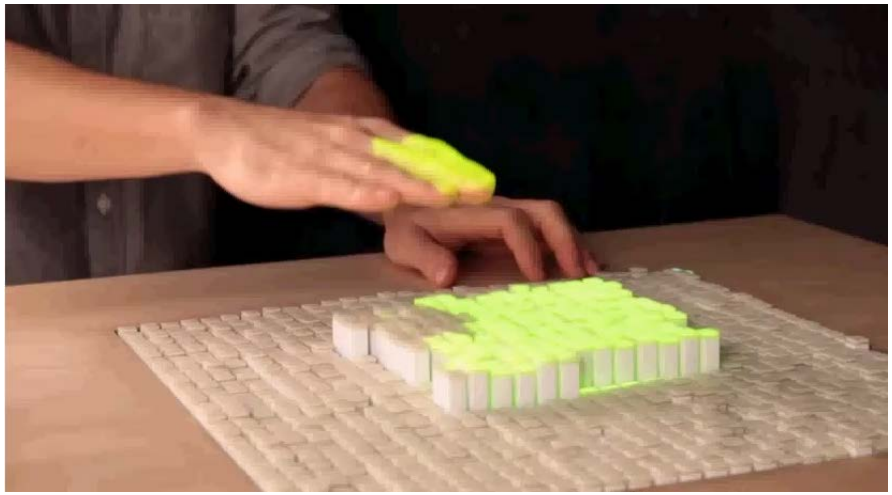
# Interactive, Physics-Driven Wave Simulation - Opportunities



# Interactive, Physics-Driven Wave Simulation - Opportunities

- Faster than real-time simulation of Boussinesq-type models:
  - Interactive, tablet based simulation
  - Wave-by-wave resolving nowcasts
  - Collaborative operations planning using “real” physics engine
  - Outreach & Education





IN-Form @ MIT
http://tangible.media.mit.edu/project/inform/