GPU Computing Using WebGL -Day 3

Thursday, March 21, 2019 8:00 AM

GPU Computing Using WebGL -Day 2

Wednesday, March 20, 2019
8:00 AM

Go to the following link

http://www.chaos.gatech.edu/ccis2019/sc1/

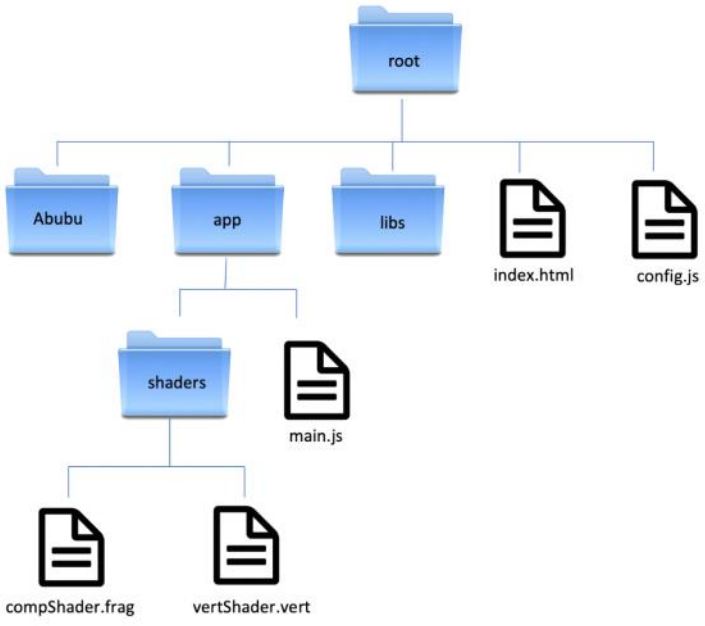
From Day 3, on the page, download the materials and the codes needed!

The objectives for today:

- Learn fundamentals of time stepping in WebGL
- Modeling systems of up-to 4 state variables
- Modeling systems with more than 4 state variables
    - Modeling Systems with upto 32 state variables
    - Modeling systems with more than 32 state variables
- Applying boundary conditions on regular grids
- Using lookup tables
- Using random numbers

Quick reminder:

Our sample programs all have the following directory structure.



The source codes that we will be editing are mostly **main.js**, **vertShader.vert** and **compShader.frag**.

Time marching problems

Systems with up-to 4 state varibles

Let's look at the FitzHugh-Nagumo model:

$$\frac{\partial u}{\partial t} = D\nabla^2 u + u(1-u)(u-a) - v$$
$$\frac{\partial v}{\partial t} = \epsilon(bu - v + \delta)$$

```
#version 300 es
precision highp float ;
precision highp int ;

uniform sampler2D   inTrgt ;
uniform float       dt ;
#define lx          8.0
#define diffCoef    0.001

uniform float   aa, bb, epsilon, delta ;
uniform float   pacePeriod ;

out vec4 outTrgt ; // output color of the shader

in vec2 pixPos ;
void main() {
    vec2 size = vec2(textureSize( inTrgt,0 )) ;
    float ddx = size.x/lx ;
    ddx *= ddx ;

/*-------------------------------------------------------------------------
 * Calculating laplacian
 *-------------------------------------------------------------------------
*/
    vec2 cc = pixPos ;
    vec2 ii = vec2(1.,0.)/size ;
    vec2 jj = vec2(0.,1.)/size ;

    vec4 laplacian = (  texture(inTrgt,cc+ii)+
                        texture(inTrgt,cc-ii)+
                        texture(inTrgt,cc+jj)+
                        texture(inTrgt,cc-jj)
                        -4.*texture(inTrgt,cc) ) ;
```

```
/*-------------------------------------------------------------------------
 *  calculate time derivatives & do an Euler update
 *-------------------------------------------------------------------------
 */
    vec4  C = texture( inTrgt, cc ) ;
    float u = C.r ;
    float v = C.g ;
    float time = C.b ;

    float du2dt = laplacian.r*ddx*diffCoef + u*(1.-u)*(u-aa)-v ;
    float dv2dt = epsilon*(bb*u-v+delta) ;


    u += du2dt*dt ;
    v += dv2dt*dt ;
    time += dt ;

    if ( (int(time)%int(pacePeriod) < 1) && length(pixPos)<0.1 )(u = 1. ;)

    outTrgt = vec4(u,v,time,1.) ;
    return ;
}
```

Boundary conditions and texture wrapping

Systems with up-to 32 state variables

Example: the Beeler-Reuter 8 variable model

Beeler, Go W., and H. Reuter. "Reconstruction of the action potential of ventricular myocardial fibres." The Journal of physiology 268.1 (1977): 177.

In the compShader.frag

```
uniform sampler2D   inMhjd ;
uniform sampler2D   inUcxf ;

layout (location = 0 )  out vec4 outMhjd ;
layout (location = 1 )  out vec4 outUcxf ;
```

In the definition of the solvers (in main.js):

The following two uniforms need to be added:

```
inMhjd     = { type : 's',     value : _inMhjd         } ;
inUcxf     = { type : 't',     value : _inUcxf         } ;
```

And the render targets need to be defined as:

```
outMhjd     = { location : 0   , target :  _outMhjd     } ;
outUcxf     = { location : 1   , target :  _outUcxf     } ;
```

Systems with more than 32 state variables

Let's look at the OVVR cardiac model:
O'Hara, Thomas, et al. "Simulation of the undiseased human cardiac ventricular action potential: model formulation and experimental validation." PLoS Comput Biol 7.5 (2011): e1002061.

In this situation, we have to break our solution in groups of 32 state variables and solve them separately.

OVVR has 41 state variables, and to solve those we have two different solvers per time step that are carried out using different shaders.

Random numbers need distributed states over the domain

An example model that needs random numbers at each point:

Kang, Qinjun, et al. "Lattice Boltzmann model for crystal growth from supersaturated solution." *Geophysical Research Letters* 31.21 (2004).