GPU Computing Using WebGL -Day 4

Friday, March 22, 2019 8:00 AM

Go to the following link
http://www.chaos.gatech.edu/ccis2019/sc1/

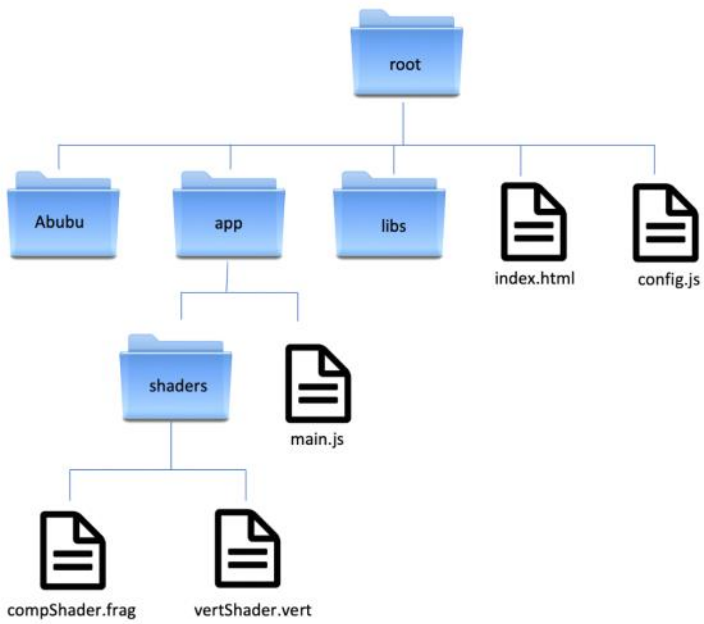
From Day 4, on the page, download the materials and the codes needed!

The objectives for today:

- Using lookup tables
- Extension to 3D simulations
- Writing data to disk
- Reduction operations
- Irregular geometries
- 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**.

Using lookup tables

Let's look at the Beeler-Reuter [1] model again:

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

$$\begin{aligned} dy/dt &= (y_\infty - y)/\tau_y, \\ \tau_y &= 1/(\alpha_y + \beta_y), \\ y_\infty &= \alpha_y/(\alpha_y + \beta_y). \end{aligned}$$

C, defining function and values for rate constants (α or β)

$$\alpha = (C_1 \exp[C_2(V_m + C_3)] + C_4(V_m + C_5))/(\exp[C_6(V_m + C_3)] + C_7). \quad (13)$$

Rate constant (msec⁻¹)	C_1 (msec⁻¹)	C_2 (mV⁻¹)	C_3 (mV)	C_4 ((mV.msec)⁻¹)	C_5 (mV)	C_6 (mV⁻¹)	C_7
α_{x_1}	0·0005	0·083	50	0	0	0·057	1
β_{x_1}	0·0013	−0·06	20	0	0	−0·04	1
α_m	0	0	47	−1	47	−0·1	−1
β_m	40	−0·056	72	0	0	0	0
α_h	0·126	−0·25	77	0	0	0	0
β_h	1·7	0	22·5	0	0	−0·082	1
α_j	0·055	−0·25	78	0	0	−0·2	1
β_j	0·3	0	32	0	0	−0·1	1
α_d	0·095	−0·01	−5	0	0	−0·072	1
β_d	0·07	−0·017	44	0	0	0·05	1
α_f	0·012	−0·008	28	0	0	0·15	1
β_f	0·0065	−0·02	30	0	0	−0·2	1

We can define textures on the CPU side (e.g. main.js) as lookup tables of pre-calculated values:

```
/* function to calculate the coeficients */
function abCoef(Vm,C){
    var al =
        (C[0]*Math.exp(C[1]*(Vm+C[2])) + C[3]*(Vm+C[4]))/
        (Math.exp(C[5]*(Vm+C[2])) + C[6]) ;
    return al ;
}

var nSamples = 512 ;                /* no. of samples          */

var Vlt = new Float32Array(nSamples) ;
var tab = new Float32Array(nSamples*4) ;

for(var i=0 ; i< nSamples; i++){
    /* Assume params.minVlt< Vlt < params.maxVlt */
    Vlt[i] = (i+0.5)*(params.maxVlt-params.minVlt)/nSamples
        + params.minVlt ;
}
/* mtht */
for(var i = 0; i<nSamples; i++){
    var indx = i*4 ;
    var V = Vlt[i] ;

    var a_m  = abCoef(V, ca_m) ;
    var b_m  = abCoef(V, cb_m) ;
    var a_h  = abCoef(V, ca_h ) ;
    var b_h  = abCoef(V, cb_h ) ;

    /* m_inf        */
    tab[ indx   ] = a_m/(a_m+b_m) ;

    /* tau_m        */
    tab[ indx+1 ] = 1.0/(a_m+b_m) ;


    /* h_inf        */
    tab[ indx+2 ] = a_h/(a_h + b_h) ;

    /* tau_h        */
    tab[ indx+3 ] = 1.0/(a_h + b_h) ;

}
var mtht = new Abubu.TableTexture(tab, nSamples ) ;
```

Using lookup tables - cont'd

Later, we can use the interpolating features of the textures in the shader to use the pre-calculated values of the error prone functions:

Let's look at the compShader.frag:

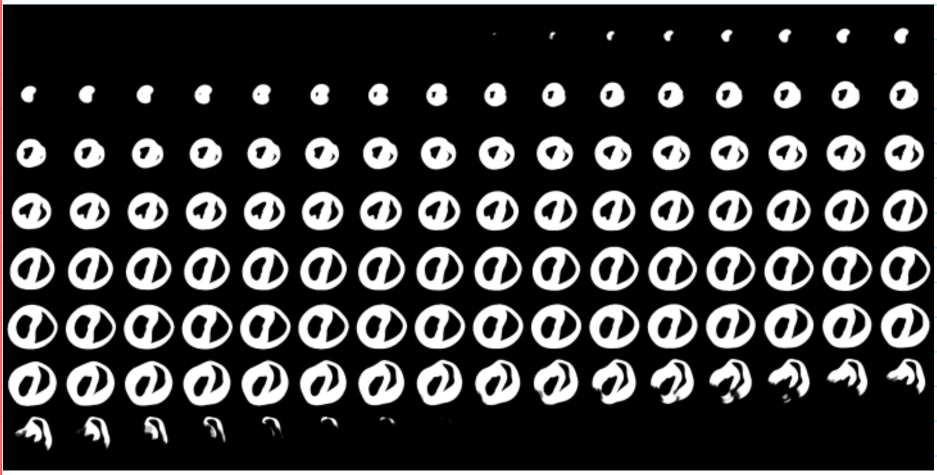
```
uniform sampler2D   mtht ;
uniform sampler2D   jtdt ;
uniform sampler2D   xtft ;
uniform sampler2D   ikix ;
```

```
    float   V       = vcxfVal.r ;
    vec2    v       = vec2((V - minVlt)/(maxVlt - minVlt),  0.5 ) ;

    vec4    mthtTab = texture( mtht, v ) ;
    vec4    jtdtTab = texture( jtdt, v ) ;
    vec4    xtftTab = texture( xtft, v ) ;
    vec4    ikixTab = texture( ikix, v ) ;
```

```
    float   m_inf   = mthtTab.r ;
    float   tau_m   = mthtTab.g*C_tau_m ;
    m = m_inf - (m_inf-m )*exp(-dt/tau_m) ;
```

Extension to 3D space



By stacking z-layers in a matrix structure we can start doing 3d simulations. We can also design the following function to substitute the texture function to carry out access to various locations of the matrix.

Let's have a look at the compShader.frag:

```
/*========================================================================
 * Texture3D
 *========================================================================
 */
vec4 Texture3D( sampler2D S, vec3 texCoord )
{
    vec3    texCrd = texCoord ;
    vec3    ii = vec3(1.)/domainResolution ;
    texCrd.x = zeroFlux(texCrd.x,ii.x) ;
    texCrd.y = zeroFlux(texCrd.y,ii.y) ;
    texCrd.z = zeroFlux(texCrd.z,ii.z) ;

    vec4    vColor1, vColor2 ;
    float   x, y ;
    float   wd = mx*my - 1.0 ;

    float zSliceNo  = floor( texCrd.z*mx*my) ;

    x = texCrd.x / mx ;
    y = texCrd.y / my ;

    x += (mod(zSliceNo,mx)/mx) ;
    y += floor((wd-zSliceNo)/ mx )/my ;

    return texture( S,  vec2(x,y) ) ;
}

```

```
/*------------------------------------------------------------------------
 * Laplacian
 *------------------------------------------------------------------------
 */
    vec3 cc = texture( crdtTxt, pixPos ).xyz ;
    vec3 ii = vec3(1.,0.,0.)/domainResolution ;
    vec3 jj = vec3(0.,1.,0.)/domainResolution ;
    vec3 kk = vec3(0.,0.,1.)/domainResolution ;

    float gamma = 1./3. ;
    vec4 laplacian = (      Texture3D( inTrgt, cc+ii)
                        -   2.*C
                        +   Texture3D( inTrgt, cc-ii)    )*cdd.x
                   + (      Texture3D( inTrgt, cc+jj)
                        -   2.*C
                        +   Texture3D( inTrgt, cc-jj)    )*cdd.y
                   + (      Texture3D( inTrgt, cc+kk)
                        -   2.*C
                        +   Texture3D( inTrgt, cc-kk)    )*cdd.z ;
    float dUlt2dt   = laplacian.r ;
    dUlt2dt *= diffCoef ;
```

Saving data to disk

Let’s at the main.js in the OVVR model:

```
/*------------------------------------------------------------------------
 * recorder.probe
 *------------------------------------------------------------------------
 */
    env.rec_probe = new Abubu.Probe( env.fvvxc, { channel : 'r',
            probePosition : [0.5,0.5] } ) ;
    env.rec_recorder = new Abubu.ProbeRecorder(env.rec_probe,
            {
                sampleRate : env.rec_interval,
                recording: env.rec_recording ,
                fileName : env.rec_fileName} ) ;
```

```
                env.rec_recorder.record(env.time) ;
```

```
            env.rec_recorder.save() ;
```

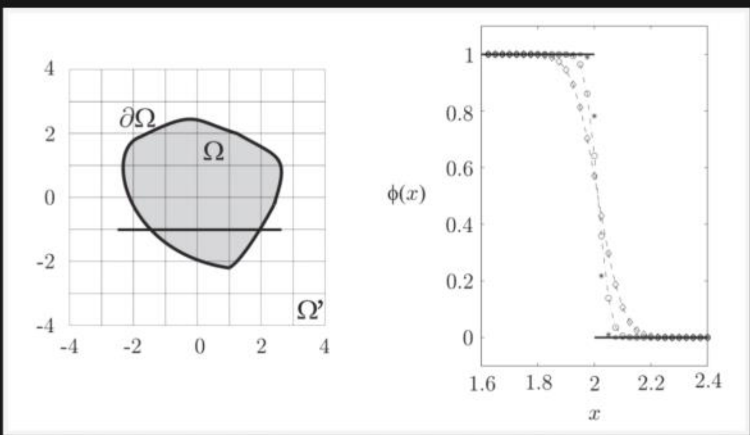
Reduction operators

Irregular geometries - Phase-Field Formulations

Fenton, Flavio H., et al. "Modeling wave propagation in realistic heart geometries using the phase-field method." *Chaos: An Interdisciplinary Journal of Nonlinear Science* 15.1 (2005): 013502.

From <https://scholar.google.com.sg/scholar?hl=en&as_sdt=0%2C11&q=fenton+phase-field&btnG=#d=gs_cit&u=%2Fscholar%3Fq%3Dinfo%3A9dqonl9agmYJ%3Ascholar.google.com%2F%26output%3Dcite%26scirp%3D0%26hl%3Den>

Fenton, Flavio H., et al. "Modeling wave propagation in realistic heart geometries using the phase-field method." Chaos: An Interdisciplinary Journal of Nonlinear Science 15.1 (2005): 013502.



$$\phi C_m \frac{\partial V}{\partial t} = \nabla.(\phi D \nabla V) - \phi I_{ion}$$

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).