Functions List

cfdHeaders.h

Included necessary header files for CFD project

cudaHelper.h

* cudaOpenLogFile (filename)
  + Create a file prepare for writing message
* cudaCheckErrors (msg)
  + Given an error message and check if device status is not cudaSuccess. Call this function will write those message to file which is created by ***cudaOpenLogFile***
* cudaCloseLogFile (void)
  + Close the file when no need to write anything.
* cudaFinished (void)
  + Must be placed at the tail of the function, it’s actually a ***GOTO*** mark pair with ***cudaCheckError.*** You can place for instance ‘release resource’ or ‘throw errors’ those operations following with ***cudaFinished***.
* cudaDevice (gridDim, blockDim)
  + This symbol is used for instead of device parameters of CUDA. With this approach, rather than the original. In order to avoid the logical and grammatical errors.
* cudaIndex2D (i, j, elements\_x)
  + High-dimensional array in memory is stored as a one-dimensional array. However, no explicit method for computing high dimensional array’s index, so we requiring such a macro definition to help us to easily calculate the index corresponding to 1-D array
* cudaIndex3D (i, j, k, elements\_x)
  + High-dimensional array in memory is stored as a one-dimensional array. However, no explicit method for computing high dimensional array’s index, so we requiring such a macro definition to help us to easily calculate the index corresponding to 1-D array
* cudaTrans2DTo3D (i, j, k, elements\_x)
  + CUDA currently only supports two-dimensional data, which requires us use 2-D to represent 3-D data. We need a macro definition in the kernel function, it will convert to a 3-D index from 2-D subscript.

macroDef.h

Contains definition of variables

visualFramework.h

* Structure of mouse events
* Structure of FPS
* Structure of view matrix
* Class of visual framework
  + Visual
  + ~Visual
  + OnCreate (void)
  + OnResize (GLuint width, GLuint height)
  + OnIdel (void)
  + OnKeyboard (SG\_KEYS keys, SG\_KEY\_STATUS status)
  + OnMouse (SG\_MOUSE mouse, GLuint x\_pox, GLuint y\_pos)
  + OnDestory (void)

launchMain.cpp

* Main
  + Create SGE framework, initialize the routine and register the callback functions.

visualFramework.cpp

* contains static variables
  + m\_mouse - recording mouse events
  + m\_fps - recording FPS
  + m\_view - recording view matrix info
  + m\_font - recording FreeType data info
  + m\_hAct - handler of MainActivity
  + m\_width, m\_height - recording client’s size
  + m\_density - Boolean value, density (true) or velocity (false)
  + m\_size - keep the dimension of grids
* Visual
  + Constructor function, initialize those variables when Visual first be created.
  + *ZeroResources (****called******in******Visual****)*
  + *AllocateList (****called******in******Visual****)*
  + *cudaInitList (****called******in******Visual****)*
* ~Visual
  + Destructor function, be called when Visual is be killed.
  + *OnDestroy (****called in ~Visual****)*
* OnCreate
  + This function will be called automatically when