```
/**************************
* Notice: This code was 'ripped' from several different places and
* should contain all the necessary 'setup' for running the GATOR (Gpu
* Accelerated Tetrahedral Renderer) code. You will also need the vertex
* programs (for both constant cells and linear cells). This code will
* not work as is and is only intended to demonstrate the setup.
* Please send all error/questions/comments to bnwylie@sandia.gov.
**************************/

/**********************
*     GL SETUP CODE    *
**********************/

void Unstruct_Vol::glSetup() {

  string    strToken = "bad";

  /* Set up material and lighting */
  GLfloat light_ambient[] = { .2, .2, .2, 1.0 };
  GLfloat light_diffuse[] = { .7, .7, .7, 1.0 };
  GLfloat light_specular[] = { 1, 1, 1, 1.0 };
  GLfloat spec[] = {1, 1, 1, 1};
  GLfloat color[] = {1, 1, 1, 1};
  GLfloat light0[] = {1, 1, 1, 0};
  GLfloat shine[] = {128.0};

  glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE, color);
  glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, spec);
  glMaterialfv(GL_FRONT_AND_BACK, GL_SHININESS, shine);
  glLightfv(GL_LIGHT0, GL_AMBIENT,   light_ambient);
  glLightfv(GL_LIGHT0, GL_DIFFUSE,   light_diffuse);
  glLightfv(GL_LIGHT0, GL_SPECULAR,  light_specular);
  glLightfv(GL_LIGHT0, GL_POSITION,  light0);
  glEnable(GL_LIGHT0);

  GLERROR2();


  // Set up openGL parameters
  glShadeModel(GL_SMOOTH);
  glEnable(GL_BLEND);
  glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
  glEnable(GL_DEPTH_TEST);
  glPolygonMode(GL_FRONT, GL_FILL);
  glPolygonMode(GL_BACK, GL_LINE);
  glEnable(GL_CULL_FACE);

  GLERROR2();

  // Texture
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
  glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);

  // Exponential texture
  for (i=0;i<4096;i++)
    for (j=0;j<4096;j++)
      expo_tex[i][j] = 1.0 - exp((-(float)i/256.0)*((float)j/256.0));

  glTexImage2D(GL_TEXTURE_2D, 0, GL_ALPHA, 4096,4096, 0, GL_ALPHA,
                  GL_FLOAT, expo_tex);

  GLERROR2();

  // Setup required to run vertex program on nVIDIA card

#ifdef __linux__

   streamBuffer << glGetString( GL_EXTENSIONS );
   while( strToken != "GL_NV_vertex_program" && !streamBuffer.eof() ) {
      streamBuffer >> strToken;
   }
   if( strToken != "GL_NV_vertex_program" ) {
      NVvertexPrograms = 0;
      printf("No nv vertex program capability.\n");
   }
   else{
      NVvertexPrograms = 1;
      printf("We have nv vertex program capability.\n");
   }
#else
   if (!glh_init_extension("GL_NV_vertex_program")){
       NVvertexPrograms = 0;
   }
   else{
       NVvertexPrograms = 1;
   }

#endif

    if (NVvertexPrograms){

      const GLubyte *program=NULL;
      int plen, i;


      /**********************************************
      ** Load the vertex program.
      **********************************************/
      program = getProgram(externalProg, &plen);

      if (!program){
  fprintf(stderr,"Can't read in vertex program %s\n", externalProg);
          NVvertexPrograms = 0;
          goto NoGoVprog;

} else{
            printf("vertex program %s, length %d, read in\n",
                        externalProg, plen);
        }
        glGenProgramsNV(1, &progID); GLERROR();
        glBindProgramNV(GL_VERTEX_PROGRAM_NV, progID); GLERROR();

        glLoadProgramNV(GL_VERTEX_PROGRAM_NV, progID, plen, program);

        if ((glerr=glGetError()) != GL_NO_ERROR){
            if (glerr== GL_INVALID_OPERATION){
                /*
                ** display the error in the program
                */
                programError(plen, (char *)program, externalProg);
                NVvertexPrograms = 0;
                goto NoGoVprog;
            }
            else{
                fprintf(stderr,"tntvol server: %s: %d (%s)\n",
                     __FILE__, __LINE__, gluErrorString(glerr));
                NVvertexPrograms = 0;
                goto NoGoVprog;
            }
        }

        /*******************************************************
        ** Write parameters to the vertex unit parameter
        ** registers, track the necessary matrices there also.
        *******************************************************/

        // Modelview-projection goes into c[0] to c[3]
        glTrackMatrixNV(GL_VERTEX_PROGRAM_NV, 0, GL_MODELVIEW_PROJECTION_NV,
                    GL_IDENTITY_NV);
        GLERROR();

        // Other program parameters
        for (i=0; i < N_VPARAMS; i++){

            glProgramParameter4fNV(GL_VERTEX_PROGRAM_NV,
               (GLuint)vparams[i][0],
               vparams[i][1], vparams[i][2], vparams[i][3], vparams[i][4]);

            GLERROR();
        }


NoGoVprog:
            if (program) free((void *)program);
        }
}

/*********************
*     END GL SETUP   *
*********************/


/*************************
*     VERTEX FEED CODE    *
*************************/

Here’s how we feed the vertices to the vertex program

// the 4 vertices geometric positions
glVertexAttrib3fvNV(1, nodes[0]-getXYZ());
glVertexAttrib3fvNV(2, nodes[1]-getXYZ());
glVertexAttrib3fvNV(3, nodes[2]-getXYZ());
glVertexAttrib3fvNV(4, nodes[3]-getXYZ());

// color for the vertices

// Constant cell
glVertexAttrib4fvNV(6, "address of color (RGBA) of cell");

     OR

// Linear cell
glVertexAttrib4fvNV(6, "address of color (RGBA) of node");
glVertexAttrib4fvNV(7, "address of color (RGBA) of node");
glVertexAttrib4fvNV(8, "address of color (RGBA) of node");
glVertexAttrib4fvNV(9, "address of color (RGBA) of node");

// This is the reciprocal of an optical distance constant
// (usually modified by the application based on the average
// cell size of the model). We use the reciprocal so that we
// don’t have to do a divide in the vertex program.

// For example: Average cell size is .05 (in model space)
// so in order to completely extinquish light the
// optical_distance will be half of the average
// cell size .025. The repirocal of that is 40.

glVertexAttrib1fNV(5, reciprocal_of_optical_distance);


// Which run is this?  (the last is identical to the second)
// Writing to v[0] here invokes the vertex program.
// There is nothing here that the user should change unless
// the vertex program is being modified/hacked/improved/etc.
glBegin(GL_TRIANGLE_FAN);
  glVertexAttrib3sNV(0, 0, 1, 0);  /* run, run==0, run != 0 */
  glVertexAttrib3sNV(0, 1, 0, 1);
  glVertexAttrib3sNV(0, 2, 0, 1);
  glVertexAttrib3sNV(0, 3, 0, 1);
  glVertexAttrib3sNV(0, 4, 0, 1);
  glVertexAttrib3sNV(0, 1, 0, 1);
glEnd();

/*************************
*     END VERTEX FEED    *
*************************/
```