```
    1: // Two-Dimensional Sierpinski Gasket
    2: // Generated using randomly selected vertices and bisection
    3:
    4: #include <iostream>
    5: #include "Angel.h"
    6:
    7: const int NumPoints = 5000;
    8:
    9: //----------------------------------------------------------------------------
   10:
   11: void
   12: init( void )
   13: {
   14:     vec2 points[NumPoints];
   15:
   16:     // Specifiy the vertices for a triangle
   17:     vec2 vertices[3] = {
   18:         vec2( -1.0, -1.0 ), vec2( 0.0, 1.0 ), vec2( 1.0, -1.0 )
   19:     };
   20:
   21:     // Select an arbitrary initial point inside of the triangle
   22:     points[0] = vec2( 0.25, 0.50 );
   23:
   24:     // compute and store N-1 new points
   25:     for ( int i = 1; i < NumPoints; ++i ) {
   26:         int j = rand() % 3;   // pick a vertex at random
   27:
   28:         // Compute the point halfway between the selected vertex
   29:         //   and the previous point
   30:         points[i] = ( points[i - 1] + vertices[j] ) / 2.0;
   31:     }
   32:
   33:     // Create a vertex array object
   34:     GLuint vao;
   35:     glGenVertexArrays( 1, &vao );
   36:     glBindVertexArray( vao );
   37:
   38:     // Create and initialize a buffer object
   39:     GLuint buffer;
   40:     glGenBuffers( 1, &buffer );
   41:     glBindBuffer( GL_ARRAY_BUFFER, buffer );
   42:     glBufferData( GL_ARRAY_BUFFER, sizeof(points), points, GL_STATIC_DRAW );
   43:
   44:     // Load shaders and use the resulting shader program
   45:     GLuint program = InitShader( "vshader_a2.glsl", "fshader_a2.glsl" );
   46:     glUseProgram( program );
   47:
   48:     // Initialize the vertex position attribute from the vertex shader
   49:     GLuint loc = glGetAttribLocation( program, "vPosition" );
   50:     glEnableVertexAttribArray( loc );
   51:     glVertexAttribPointer( loc, 2, GL_FLOAT, GL_FALSE, 0,
   52:                            BUFFER_OFFSET(0) );
   53:
   54:     glClearColor( 1.0, 1.0, 1.0, 1.0 ); // white background
   55: }
   56:
   57: //----------------------------------------------------------------------------
   58:
   59: void
   60: display( void )
   61: {
   62:     glClear( GL_COLOR_BUFFER_BIT );     // clear the window
   63:     glDrawArrays( GL_POINTS, 0, NumPoints );    // draw the points
```

```
    glFlush();
}

//----------------------------------------------------------------------------

void
keyboard( unsigned char key, int x, int y )
{
    switch ( key ) {
    case 033:
        exit( EXIT_SUCCESS );
        break;
    }
}

//----------------------------------------------------------------------------

int
main( int argc, char **argv )
{
    glutInit( &argc, argv );
    glutInitDisplayMode( GLUT_RGBA );
    glutInitWindowSize( 512, 512 );

    // If you are using freeglut, the next two lines will check if
    // the code is truly 3.2. Otherwise, comment them out

    glutInitContextVersion( 3, 2 );
    glutInitContextProfile( GLUT_CORE_PROFILE );

    glutCreateWindow( "Sierpinski Gasket" );

    glewExperimental = GL_TRUE;
    glewInit();

    init();

    glutDisplayFunc( display );
    glutKeyboardFunc( keyboard );

    std::cout << "OpenGL version: " << glGetString(GL_VERSION) << std::endl;

    glutMainLoop();
    return 0;
}
```