```
//------------------------------------------------------------------------------------
//  Filename: main.cpp
//------------------------------------------------------------------------------------
//------------------------------------------------------------------------------------
//  Notes:
//
//  + An internal path is created from an ADFPath by the CPU. The ADFPath
//    is transformed from font units to floating point image coordinates
//    and curves are tesselated into line segments. The resulting internal
//    path (which consists of line cells and corner cells) is then
//    processed by the GPU.
//
//  + We exploit the depth buffer of the GPU to process line cells and
//    corner cells as follows.  A fragment shader computes the distance
//    from a sample point to the line segment of a line cell or to the
//    corner point of a corner cell, maps the distance to a density value,
//    and stores the density value as the output color of the fragment. The
//    shader can also store the distance value (scaled to [0,1]) as the
//    output depth of the fragment. Scaling is easy: just divide the
//    distance by the filter radius (we actually precompute 1/filterRad and
//    pass the inverse to the shader as a constant). The end result is that
//    the fragment will have a Z value that is in the range [0,1] and this
//    Z value is a scaled version of the real, true distance value.
//    Consequently, we can just turn on GPU depth testing with the standard
//    LESS (i.e., <) compare function, and the GPU will simply keep only the
//    fragments with the smallest Z values, which correspond exactly to the
//    sample points with the minimum distance values. In summary, the GPU
//    depth buffer acts as a pseudo-distance buffer, and the GPU color buffer
//    ends up holding the density values associated with the sample points
//    with the minimum distances.
//
//  + To determine interior pixels via the GPU, we use the stencil buffer
//    to perform a fence fill algorithm on the internal path.
//------------------------------------------------------------------------------------
```