Source code for Finding Faces

April 2010

We are interested in creating computer-based tools to help design engineers during the first stage of the design process, known as conceptual design. Hence, we develop Sketch-Based Modelling (SBM) systems. Our approach for SBM relies on finding cues or regularities, which are those sketch properties which reveal properties of the three-dimensional object depicted in the sketch. In this context, faces are cues. In particular, the *finding-faces* algorithm we consider here (FF) is aimed at finding circuits of lines that represent faces of the polyhedral shapes depicted by wireframe models of polyhedral objects.

Attached is a C++ implementation of the code of the new FF algorithm.

Although most of the cues are mutually related, we intend to detect every cue using minimal information. For FF to work, the input sketch must have been previously vectorised, so as to convert strokes of the sketch into lines of the line drawing. Hence, the input to the code is a 2D drawing, which includes a list of vertices and edges in the following format:

- Coordinates of every vertex are stored in an instance of the POINT2D class. The set of all vertices is stored in a standard vector (std::vector <POINT2D>Vertex)
  - VertexCount= number of vertex in the line drawing
  - VertexX[i]= X coordinate of the i-th vertex
  - VertexY[i]= Y coordinate of the i-th vertex
- Edges are defined by way of their head and tail vertices. The set of all edge heads is stored in a standard vector (std::vector <long> EdgeU) and the set of all edge tails is stored in another standard vector (std::vector <long> EdgeV):
  - EdgeCount= number of edges in the line drawing
  - EdgeU[i]= Head vertex defining the i-th edge
  - EdgeVl[i]= Tail vertex defining the i-th edge

Reader must note that the vectorisation of the sketch required by this approach includes previous merging of endpoints of all the edges that share nodes or vertices of the polyhedral shape (merging into a single vertex all endpoints that should be perceived by humans as a common junction).

The output is a set of lists of lines of the drawing, each one listing the lines that belong to a particular face of the shape depicted in the drawing. The set of lists of vertices that belong to every face is returned too.

- The information is stored in a vector of instances of a FACE class, (std::vector <FACE> Face), where every FACE instance includes:

  int C= Number of edges/vertices that define the face
  std::vector <int> E= List of edges that define the face
  std::vector <int> V= List of vertices that define the face

The approach is encapsulated into a main class CFacesVarley, which shares the same file with some auxiliary classes (CueFacesVarley.cpp). Besides, it includes three auxiliary files (CueFacesV.Geometry.cpp, CueFacesV_Nuts.cpp and CueFacesV_Ortholin.cpp).

This approach for finding faces is extensively described in:

Varley P.A.C. and Company P. (2010). A new algorithm for finding faces in wireframes. Computer-Aided Design, 42(4), pages 279-309.

To help understanding how the code works, and in order to offer a test environment too, the following main file is also provided:

- Main.cpp, and its header Main.h.

Finally, we must highlight that the code was written to make it readable. Efficiency never was a goal.

***

The FF code is free software. But if you find it useful for your own research, please cite our paper:

Varley P.A.C. and Company P. (2010). A new algorithm for finding faces in wireframes. Computer-Aided Design, 42(4), pages 279-309.