View3D version 3.5

User Manual

by George N. Walton (retired)
Building Environment Division
National Institute of Standards and Technology
Gaithersburg, MD 20899-8633
gwalton@nist.gov

Updated by John Pye 23 Oct 2008.

This document is incomplete and contains errors in some places. Please consider it only as a rather out-of-date guide at this stage. We are working on gradually bringing it up to date. There are a number of example models provided with the source code tarball for View3D which should be useful in combination with this document. -- John Pye 23 Oct 2008.

This document only really covered the 'view3d' program. Another program 'view2d' is also provided with the source code for View3D, but it is not currently documented here.

New 2D and 3D viewer programs have been added to View3D by John Pye. These are 'viewer' and 'viewer2d'. To compile them you need to have SoQt and Cairo development packages installed on your machine.

According to George Walton, the View2D program still requires further testing before it should be considered mature.

Comments by John Pye are given in red, do distinguish them from the more-or-less original text by George.

Certain trade names or company products are mentioned in the text to specify adequately the software and equipment used. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that they are the best available for the purpose.

1. BACKGROUND

This file provides a brief description of use of the **View3D** program for computing view

factors 3-dimensional geometries described in terms of simple planar polygons – triangles and convex quadrilaterals. It is an extension of previous work. The algorithms use improved integration and geometric methods with special emphasis on the processing of view obstructions.

The earlier VLITE program [2] attempted to predict the subdivisions necessary for the view factor numerical integrations, but this process would occasionally leave relatively large errors in some of the computed values. The new program uses adaptive integration to control the calculation to approximately the same level of accuracy for all view factors. Once the view factors are computed several post-processing operations are executed to combine or separate surfaces, include the effect of surface emissivity (Hottel's "script-F" view factors or "total interchange areas" [3]), and slightly adjust the view factors to guarantee conservation of energy when they are used for an enclosure.

History:
(See CHANGELOG.txt)

Full details of the View3D program and the algorithm that it implements are available from:

Walton, G. N.: "Calculation of obstructed view factors by adaptive integration", Technical Report NISTIR–6925, National Institute of Standards and Technology, Gaithersburg, MD, 2002.

http://www.bfrl.nist.gov/IAQanalysis/docs/NISTIR-6925.pdf

Abstract from the above report:

This report describes the use of adaptive integration for the calculation of view factors between simple convex polygons with obstructions. The accuracy of the view factor calculation is controlled by a convergence factor. The adaptive integration method is compared with two other common methods implemented in a modern computer program and found to have significant advantages in accuracy and even advantages in computational speed in some cases.

2. INSTALLATION

2.1 System Requirements

~~The View3D program was compiled with Microsoft Visual C++™ to run in MS-DOS™ mode~~

~~under the MS Windows™ operating system on PC™ -compatible computers because of common availability. The program has been implemented as a simple "console application" without a graphic user interface because its only purpose is to test the computational algorithms. It is intended that the algorithms eventually become part of more general heat transfer programs. The source code is in ANSI C, which should allow recompilation for almost any operating system. It should also be relatively easy to modify the program to make it a subroutine called from other programs. The source code will be available upon request after the program is officially released.~~

As of View3D version 3.4, only a source code tarball is provided. This code can be easily compiled on the Linux platform using the Python-based build tool 'SCons'. SCons is available with most Linux distributions and also runs on Windows. In the next View3D release, we will be working on restoring support for Windows, but this time using the MinGW GCC compiler.

As of version 3.4, 2D and 3D viewer code is required. This code requires the GTK+ and Cairo libraries (for the 2D viewer) and the SoQt and Coin3D libraries (for the 3D viewer). The 3D viewer is written in C++. The 2D viewer is written in plain C.

2.2 Installation

Currently no installation is required. Compile the programs using SCons, then they will appear in the 'build' subdirectory. In the next version we will work on providing an installation process that will allow you to install all necessary files permanently on your system.

3. USING VIEW3D

All input and output is through simple ASCII text files. For a reminder of the sequence of file names on the program command line just enter the program name followed by a space and then a question mark.

3.1. Executing VIEW3D

View3D computes view factors for a three-dimensional geometry. First you need to create an input file

Usage: VIEW3D <input.vs3> <output.txt>

where:

- <input.vs3> is the name of the 3-D vertex and surface data file (we will choose to use

a '.vs3' suffix for these files, although they contain just plain text) (see 3.1) and

- <output.txt> is the name of the view factor output file (see 3.2). We don't have a recommended file suffix for these files at this stage.

You can also use view2d in a similar way, except that it uses a 2D data file format. We plan to merge these two programs into a single commandline program capable of dealing with both file formats.

Finally a program called 'cavity' (in the examples subdirectory) is provided, which demonstrates how you can automate the generation of larger 2D vertex/surface data files using the C++ library Coin3D to perform some simple 2D vector algebra.

3.1. Input File Format

Each line of a data file for the **View3D** program contains specific information in a specific order. Each line begins with a single character to specify the kind of data on the rest of the line. Remaining data elements are separated by one or more blanks.

(**!** /) A '!' or '/' begins a comment which reports information to the user but is not read by the program. Use comments to document special features described by the input file. A comment may also begin after the data on a line.

(**T t**) A 'T' or 't' indicates a title. The remaining information on this line is transferred to all intermediate files and is echoed in all view factor reports. This is usually the first line of the input file. If there are multiple title lines, only the contents of the last will be echoed.

(*** E e**) A '*' or 'E' or 'e' indicates end-of-information which terminates reading the input file. Any information on the file after this line will not be read. This allows you to store additional information, such as different geometric descriptions, at the end of the input file.

(I i) Identifier number (0 to 2^31) @@ ???

Note that the syntax for 'C' lines may differ from that documented here. Note all parameters appear to still be present in the code either.

(**C c**) The control line includes the following parameters (in order): name = value

eps = 1.0e-4: integration convergence criterion – for both adaptive integration and view obstruction. This is not an exact measure of the accuracy of the computed view factors, but smaller values will usually lead to more precise values. The convergence criteria should not be less than about 1.0e-6 because many of the intermediate calculations are accurate only to single (32-bit) precision.

maxU = 8: maximum recursions used in computing the unobstructed view factors.

maxO = 8: maximum recursions used in computing the obstructed view factors. Limiting the maximum number of recursions limits the total execution time of the program but may prevent reaching the specified convergence.

minO = 0: minimum recursions: used in computing the obstructed view factors. This can help in cases where an obstruction occurs very near the view between the edges of two surfaces. The normal adaptive integration may miss the obstruction. Increasing this value from its normal value of 0 to 1 or 2 may catch the obstruction. This is probably not necessary except when very accurate view factors are desired. It can add considerably to execution time.

row = 0: selected row for computing view factors (0 = all rows)

col = 0: selected column for computing view factors (0 = all columns)

encl = 0: 1 indicates that the surfaces form an enclosure; 0 indicates that they do not. This data is used to adjust the view factors of an enclosure to guarantee conservation of energy.

emit = 0: 1 indicates that diffuse reflectance effects will be included in the computed view factors; 0 indicates they will not, i.e., surfaces will be considered 'black'. If emit=1, your computed view factors will be modified to become Hottel's 'script F' view factors. If emit=0, you will get a bona fide geometric view factors, which is probably what you want.

out = 0: view factor output file format – 1 = simple text file, 0 = binary.file.

list = 0: computational summary written to the VIEW3D.LOG file; 0 gives minimal information; 1 gives slightly more; 2 prints all the view factors; 3 causes dumping of some intermediate values.

(**F f**) Most of the input file is devoted to describing the geometry of the surfaces plus some related surface data. There are several surface types and two ways of describing surface geometry. The geometry line indicates the type of geometry being described by the rest of the input file. It consists of a 'F' or 'f' followed by a single parameter: 3a or 3b (2 is reserved for a 2-D geometry processed by another program, and 1 is reserved for a "1-D" geometry using only surface areas for "very approximate" view factors.)

Note: in 2D problems, the parser currently expects you to have the text 'G 2' instead of 'F 2'. We propose to unify the 2D and 3D vertex/surface data file syntax, but this work is still ongoing.

Conventional surface radiating to other surfaces - **S**

Restrictions: A surface has an active side and an inactive side. (See geometry below.)

Subsurface - **S** + base

A subsurface, such as a window in a wall, is indicated by setting the base pointer to the wall surface number. A subsurface interacts radiatively with other surfaces except the base surface. View factors between the base and other surfaces are reduced by the subsurface during post-processing.

Restrictions:

The subsurface must lie in the plane of its base surface.
It must face in the same direction as its base.
It must be entirely within its base.
Its base may also be a subsurface.
It must be numbered immediately after its base.

Masking surface - **M** + base

A masking surface is much like a subsurface except that it faces toward the base thereby blocking the view from the masked portion of the base to other surfaces. The view factor from the mask to the base is 1.

Restrictions:
The masking surface must lie in the plane of its base surface.
It must face in the opposite direction as its base.
It must lie entirely within its base.
Its base may be any type S surface including a subsurface.

Null surface - **N** + base

A null surface is a special type of masking surface that nullifies the covered portion of the base surface. The null surface is removed from the matrix of radiating surfaces, and the area of the base surface is reduced by the area of the null surface during post-processing.

Restrictions are identical to those for a masking surface.

Non-radiating, view obstructing surface - **O**

An obstruction surface may block the view between radiating surfaces but there is no view factor computed between it and the radiating surfaces.

Subsurface of an obstruction - **S** + base

The subsurface participates as a radiating surface with the other type S surfaces. The base surface number must refer to a type O surface. The purpose of this arrangement it to simplify the view obstruction calculation when several surfaces can be grouped as a single view obstructing surface.

The following is required for the type 3a geometry:

(**V v**) The vertex lines give the coordinates of each vertex. Each line consists of a 'V' or 'v', the vertex number, and the X, Y, and Z coordinates of the vertex. The vertices must be numbered in order starting at 1.

(**S s**) (**M m**) (**N n**) (**O o**) Each line consists of the surface type character (S, M, N, or O), the surface number, four vertex pointers, a possible base surface number, a possible surface combination number, the emissivity of the surface, and the surface name. The surfaces must be numbered in order beginning at 1.

```
!   #  v1  v2  v3  v4  base cmb emit  name
S   1   1   4   3   2    0   0  0.90  X=0
```

The vertex pointers are the numbers of the *previously defined* vertices listed in counter-clockwise sequence as viewed from the front of the surface. If the surface is a triangle, the fourth vertex number should be zero. Quadrilateral surfaces must be convex, i.e. no interior angle greater than 180 deg.

The meaning of the base surface number depends on the surface type, explained below (above?).

Surface combination allows several surfaces to be combined into a single surface in the final view factors. This is necessary for describing a non-convex surface; until post-processing it must be treated as two or more convex surfaces. The wall around the window could have been described as four rectangles which will be combined during post-processing. Combination reduces the number of surfaces and causes a renumbering of the surfaces. The surface(s) to be combined may appear anywhere after the surface it is (they are) to be combined. A non-zero value indicates the number of the surface with which this surface is combined.

Review the WINDOWS.VS3 data file to see surface inclusion and combination both used to represent a door and a window located on a wall.

The surface emissivity allows for the modeling of "gray" reflection from non-black surfaces. The values may range between 0.01 (very reflective) and 0.99 (near black).

The surface name helps to identify surfaces in the reports. A name may have no imbedded blanks; use '_', '-', or selective capitalization instead of blanks.

Example of a G=3 input file:

```
T  Pinney & Bean test 4: L-shaped room (page A2.14; Table A2.11) BRE
! encl list  epsu  epso  maxu maxo mino emit
C  1    2   1.e-4  1.e-4   8    8    0    1       ! run control values
G  3
!  #   x    y    z      coordinates of vertices
V  1   0.   3.   3.
V  2   1.   3.   3.
V  3   0.   3.   0.
V  4   1.   3.   0.
V  5   0.   1.   0.
V  6   1.   1.   0.
V  7   3.   1.   0.
V  8   0.   0.   0.
V  9   1.   0.   0.
V 10   3.   0.   0.
V 11   0.   0.   3.
```

```
V 12   1.   0.   3.
V 13   3.   0.   3.
V 14   0.   1.   3.
V 15   1.   1.   3.
V 16   3.   1.   3.
!  #   v1  v2  v3  v4 base cmb  emit  name      surface data
S  1   11  13  10   8   0   0   0.90  srf-1
S  2   10  13  16   7   0   0   0.90  srf-2
S  3    6   7  16  15   0   0   0.90  srf-3
S  4    4   6  15   2   0   0   0.90  srf-4
S  5    3   4   2   1   0   0   0.90  srf-5
S  6    8   3   1  11   0   0   0.90  srf-6
S  7   11  14  15  12   0   0   0.90  srf-7
S  8    8   9   6   5   0   0   0.90  srf-8
S  9    1   2  15  14   0   7   0.90  srf-7b    ! surfaces that combine
S 10   15  16  13  12   0   7   0.90  srf-7c
S 11    5   6   4   3   0   8   0.90  srf-8b
S 12    9  10   7   6   0   8   0.90  srf-8c
End of data
```

The following is required for the type 3b format @ geometry:

(**S s**) (**M m**) (**N n**) (**O o**) Each surface is then described by two or three lines of data which fully describe the surface and its vertices. The first line is similar to the surface lines described above except that instead of four vertex pointers there is a shape indicator: **R** for a rectangle, **Q** for a general quadrilateral, and **T** for a triangle. The data sequence is: surface type (S, M, N, or O), surface shape (R, Q or T), a possible base surface number, a possible surface combination number, the emissivity of the surface, and the surface name. The surfaces must be numbered in order beginning at 1.

The second line will include the 3-D coordinates (X, Y, Z) of the origin a plane containing the surface, the azimuth angle (clockwise from the Y axis - north) of that plane, and its tilt angle (vertical = 90, facing flat upward = 0, flat downward = 180). The data for the vertices depends on the shape. They may be placed on a subsequent line.

A rectangle (shape = R) is described by its width and height. The origin coordinates refer to the lower left corner of the rectangle as viewed from the 'front' or active side. The vertices of a quadrilateral (or triangle) is described by four (or 3) sets of (X, Y) coordinates relative to the origin of the plane.

Example of a G=4 input file:

```
T Pinney & Bean test 4: L-shaped room (page A2.14; Table A2.11) BRE
! encl list  epsu  epso  maxu maxo mino emit
C  1    2   1.e-5  1.e-4   8    8    0    0
G  4
!  #  s base cmb emit  name
S  1  R   0   0  0.90  srf-1
  3. 0. 0.    0  90    3. 3.
```

```
S  2  R   0   0  0.90  srf-2
  3. 1. 0.  270  90    1. 3.
S  3  r   0   0  0.90  srf-3
  1. 1. 0.  180  90    2. 3.
S  4  R   0   0  0.90  srf-4
  1. 3. 0.  270  90    2. 3.
S  5  R   0   0  0.90  srf-5
  0. 3. 0.  180  90    1. 3.
S  6  R   0   0  0.90  srf-6
  0. 0. 0.   90  90    3. 3.
S  7  Q   0   0  0.90  srf-7
  3. 0. 3.    0  180
  2. 0.   3. 0.   3. 1.   2. 1.
S  8  Q   0   0  0.90  srf-8
  0. 0. 0.  180   0
  0. 0.   1. 0.   1. 1.   0. 1.
S  9  Q   0   7  0.90  srf-7b
  3. 0. 3.    0  180
  2. 1.   3. 1.   3. 3.   2. 3.
S 10  Q   0   7  0.90  srf-7c
  3. 0. 3.    0  180
  0. 0.   2. 0.   2. 1.   0. 1.
S 11  Q   0   8  0.90  srf-8b
  0. 0. 0.  180.  0.
  0. 1.   1. 1.   1. 3.   0. 3.
S 12  q   0   8  0.90  srf-8c
  0. 0. 0.  180.  0.
  1. 0.   3. 0.   3. 1.   1. 1.
End of data
```

3.2. View Factor Output File

The first line echoes the title from the input file (a). The second line has four integer values: the number of surfaces, the enclosure flag (1 if surfaces form an enclosure), the emittances flag (1 if emittances have been included in view factors), and the output control flag (b). For each surface there is a header line giving the surface (row) number, surface area, emittance, and name (c). That is followed by N ($A_n F_{n\to m}$) values (d), where N is the surface number. In other words, only the values in the lower triangle of a square N by N symmetric matrix are printed. These "view factors" will include the effect of surface emissivity if that calculation has been specified in the run control data. The following example view factor file was generated using the example input file in section 3.1.

Example of a (triangular) view factor file:

```
T Pinney & Bean test 4: L-shaped room (page A2.14; Table A2.11) BRE      (a)
C 8 1 0 1                                                                (b)
```

```
!  #     area          emit    name                                      header
   1  9.000000e+000   0.900   srf-1                                        (c)
 2.711825e-001                                                           (d)
   2  3.000000e+000   0.900   srf-2                                        (c)
 8.931738e-001  2.881335e-002                                            (d)
   3  6.000000e+000   0.900   srf-3                                        (c)
 2.828227e+000  8.196874e-001  1.512733e-001                             (d)
   4  6.000000e+000   0.900   srf-4
 2.932999e-001  2.230434e-002  3.695820e-002  1.512733e-001
   5  3.000000e+000   0.900   srf-5
 2.768928e-001  1.114731e-002  2.230434e-002  8.196874e-001  2.881335e-002
   6  9.000000e+000   0.900   srf-6
 1.431202e+000  2.768927e-001  2.932999e-001  2.828227e+000  8.931738e-001
 2.711825e-001
   7  5.000000e+000   0.900   srf-7
 1.053011e+000  3.239905e-001  6.241250e-001  6.241250e-001  3.239905e-001
 1.053011e+000  5.466339e-002
   8  5.000000e+000   0.900   srf-8
 1.053011e+000  3.239905e-001  6.241250e-001  6.241250e-001  3.239905e-001
 1.053011e+000  4.430834e-001  5.466339e-002
```

3.3. View3D Log File

A log file (VIEW3D.LOG) is produced whenever View3D is run. It provides information on the calculation of the view factors. The contents of the following example log file are noted with the comments (.) below it.

Some changes to current View3D SF.net code has disabled certain of the logfile output, because we are moving to a shared library implementation in which use of global variables needs to be eliminated. Currently, logfile output is likely to vary significantly from that given below.

Example of a View3D log file:

```
Program: C:\BC\VIEWS2K\VIEW3D\VIEW3D.EXE
Time:  Wed Mar  1 11:04:43 2000                                          (a)
Data:  test4.vs3
Title: Pinney & Bean test 4: L-shaped room (page A2.14; Table A2.11) BRE
Control values for 3-D view factor calculations:                         (b)
     enclosure designator:   1
 output control parameter:   2
 unobstructed convergence: 1e-004
   obstructed convergence: 1e-004
  unobstructed recursions:   8
    obstructed recursions:   8
       process emittances:   1
```

```
 total number of surfaces:  12
   heat transfer surfaces:  12
      volume of enclosure: 15.000                                          (c)
    possible obstructions:   2
Possible view obstructing surfaces:     4     3                            (d)

Surface pairs where F(i,j) must equal zero:    13                         (e)
Surface pairs without obstructing surfaces:    39                         (f)
   nd     2AI     1AI     2LI     1LI     ALI
    2       0       0       0       0      34 direct
    3       5       0       0       0       0 fixes                       (g)
    4       0       0       0       0
  fix       0       0       0       0
Adaptive line integral evaluations used:     1424
Surface pairs with obstructing surfaces:       14                         (h)
Average number of obstructions per view:     1.50
Obstruction adaptive evaluations used:        944
Obstruction adaptive evaluations lost:        906

   0.16 seconds to compute view factors.                                   (i)
 New,   Old surface numbers                                                (j)
   1:   1
   2:   2
   3:   3
   4:   4
   5:   5
   6:   6
   7:   7   9  10
   8:   8  11  12
Number of surfaces reduced to 8.

Pinney & Bean test 4: L-shaped room (page A2.14; Table A2.11) BRE
          #        name   SUMj Fij (encl err)
 Row:     1        srf-1  1.000003 (0.000003)                              (k)
.000000 .113154 .378093 .027473 .032890 .182357 .133018 .133018
 Row:     2        srf-2  0.999923 (0.000077)
.339463 .000000 .318996 .000000 .000000 .098671 .121396 .121396
 Row:     3        srf-3  1.000000 (0.000000)
.567140 .159498 .000000 .000000 .000000 .041210 .116076 .116076
 Row:     4        srf-4  1.000000 (0.000000)
.041210 .000000 .000000 .000000 .159498 .567140 .116076 .116076
 Row:     5        srf-5  0.999923 (0.000077)
.098671 .000000 .000000 .318996 .000000 .339463 .121396 .121396
 Row:     6        srf-6  1.000003 (0.000003)
.182357 .032890 .027473 .378093 .113154 .000000 .133018 .133018
 Row:     7        srf-7  0.999947 (0.000053)
.239432 .072838 .139291 .139291 .072838 .239432 .000000 .096826
 Row:     8        srf-8  0.999947 (0.000053)
.239432 .072838 .139291 .139291 .072838 .239432 .096826 .000000
Summary:
Max enclosure error:  7.75e-005                                            (l)
```

```
RMS enclosure error:  4.69e-005
Largest errors [row, error]:
       5  0.000077
       2  0.000077
       8  0.000053
       7  0.000053
       1  0.000003
       6  0.000003

NormAF: 1  maxError: 7.86e-005                                          (m)
NormAF: 2  maxError: 1.68e-005
NormAF: 3  maxError: 1.49e-006
NormAF: 4  maxError: 8.31e-007
NormAF: 5  maxError: 2.66e-007
NormAF: 6  maxError: 6.06e-008
6 normalization iterations.
   0.06 seconds to adjust view factors.
   0.00 seconds to include emissivities.                                (n)
NormAF: 1  maxError: 1.81e-007
NormAF: 2  maxError: 7.33e-008
2 normalization iterations.

Final view factors:
          #        name   SUMj Fij (encl err)
 Row:     1        srf-1  0.900000 (0.000000)                           (o)
.030131 .099242 .314247 .032589 .030766 .159022 .117001 .117001
 Row:     2        srf-2  0.900000 (0.000000)
.297725 .009604 .273229 .007435 .003716 .092298 .107997 .107997
 Row:     3        srf-3  0.900000 (0.000000)
.471371 .136615 .025212 .006160 .003717 .048883 .104021 .104021
 Row:     4        srf-4  0.900000 (0.000000)
.048883 .003717 .006160 .025212 .136615 .471371 .104021 .104021
 Row:     5        srf-5  0.900000 (0.000000)
.092298 .003716 .007435 .273229 .009604 .297725 .107997 .107997
 Row:     6        srf-6  0.900000 (0.000000)
.159022 .030766 .032589 .314247 .099242 .030131 .117001 .117001
 Row:     7        srf-7  0.900000 (0.000000)
.210602 .064798 .124825 .124825 .064798 .210602 .010933 .088617
 Row:     8        srf-8  0.900000 (0.000000)
.210602 .064798 .124825 .124825 .064798 .210602 .088617 .010933
Summary:
Max enclosure error:  5.62e-009
RMS enclosure error:  3.46e-009

   0.27 seconds for all calculations.
```

(a) The log file begins with a review of some program data.
(b) The title and run control data are echoed from the input file.
(c) If the surfaces form an enclosure, its volume is computed.
(d) The program identifies surfaces which might obstruct views between other surfaces before an view factors are computed.

Several items are reported after the view factors have been computed:
(e) Some surfaces cannot view each other because of their position and orientation; these view factors are set to zero.
(f) The view between some surface pairs is not interrupted by any other surfaces. These view factors are computed in one of several ways:
2AI = double area integration, 1AI = single area integration, 2LI = double line integration, 1LI = single line integration, and ALI = adaptive line integration. In the example 5 view factors were computed by double area integration with 4 edge divisions, and 34 view factors were computed by adaptive line integration.
(g) When a nonadaptive integration requires more than 4 edge divisions, adaptive integration is used to complete ("fix") the calculation.
(h) The views between other surface pairs may be (partially) obstructed. This is handled by adaptive single area integration. Note that the number of surface pairs (e) + (f) + (h) should equal $\frac{N(N-1)}{2}$, or 66 in this case.
(i) The time to compute the view factors is accurate to about 1/20 second.
(j) During post-processing 6 rectangular surfaces are combined into 2 L-shaped surfaces reducing the total number of surfaces to 8.
(k) If the surfaces form an enclosure, the sum of the view factors for each row should equal one. The difference between the actual sum and 1.0 for each row gives an indication of the error in the view factor calculations. This listing of view factors can be eliminated with a lower output control flag.
(l) Maximum and root-mean-square row error values are given. Large error values probably indicate an error in the description of the geometry.
(m) Normalization refers to a process which adjusts the computed view factors for each row to give the proper sum while maintaining reciprocity: $A_n \cdot F_{n,m} = A_m \cdot F_{m,n}$.
(n) The emissivities are included in the view factors using Hottel's method [3] and the resulting view factors are again adjusted to guarantee conservation of energy (this fixes any round-off errors from factoring the matrix).
(o) After the emissivities have been included, the sum of the revised view factors for each row should equal the surface emissivity (if the surfaces form an enclosure).

AUXILLIARY PROGRAMS

ViewGR

ViewGR provides a visual display of the surfaces defined by the input files. View factor calculations are all about geometry. The ASCII data files provide a portable means for transferring data, but they are awkward for entering or visualizing geometry data. Manual entry of geometry data is prone to error. Use VIEWGR to visually find errors in the definitions of vertex coordinates or surface vertices.

Usage: VIEWGR <input>
This is an interactive program allowing the user to look at the 3-D data file from different view points. Instructions for the interactive use of the program appear at the bottom of the screen. Since this program accesses the screen graphically it must include non-ANSI C functions.

Currently the ViewGR program is not included in the View3D SF.net distribution, because it was written for the Windows API and would not run on Linux. George Walton does still have the source code for this program, however, so it might be possible to get it running. In the meantime, it is suggested to use the 'viewer' and 'viewer3d' programs listed below. -- John Pye, 23 Oct 2008.

ViewHT

ViewHT takes the view factor file and computes the surface radiant fluxes for a given set of temperatures in the temperatures file. The fluxes are reported in the VIEWHT.LOG file. This program was developed to generate data for comparison to reported test cases.

Usage: `viewht <vf> <tk> <output>`

See section 4.3 for the format of the Tk file (<tk>) and 5.5 for the output file (<output>)

ViewHT was incorporated into to the View3D SF.net distribution as of version 3.5, when George Walton sent me the source code and it was merged with the latest code by John Pye. Currently, the logfile output is largely disabled, but the output file is generated corrected, and matches the sample output file received from George -- John Pye, 23 Oct 2008.

ViewSq

ViewSq (by George Walton) converts triangular view factor file (see 5.1), to a file VFs describing a square matrix (see 5.2).

Usage: VIEWSQ VFt VFs

ViewSq is not currently included with the View3D SF.net distribution. The code can probably be obtained from George Walton if you have a need for it.

viewer

viewer (by John Pye) displays a 3D rendering of the surfaces described by a View3D input data file (see section 3.1). It requires the SoQt 3D graphics library, available in most standard linux distributions as well as in binary installer form for Windows[1].

Usage: viewer <3Dscenefile>

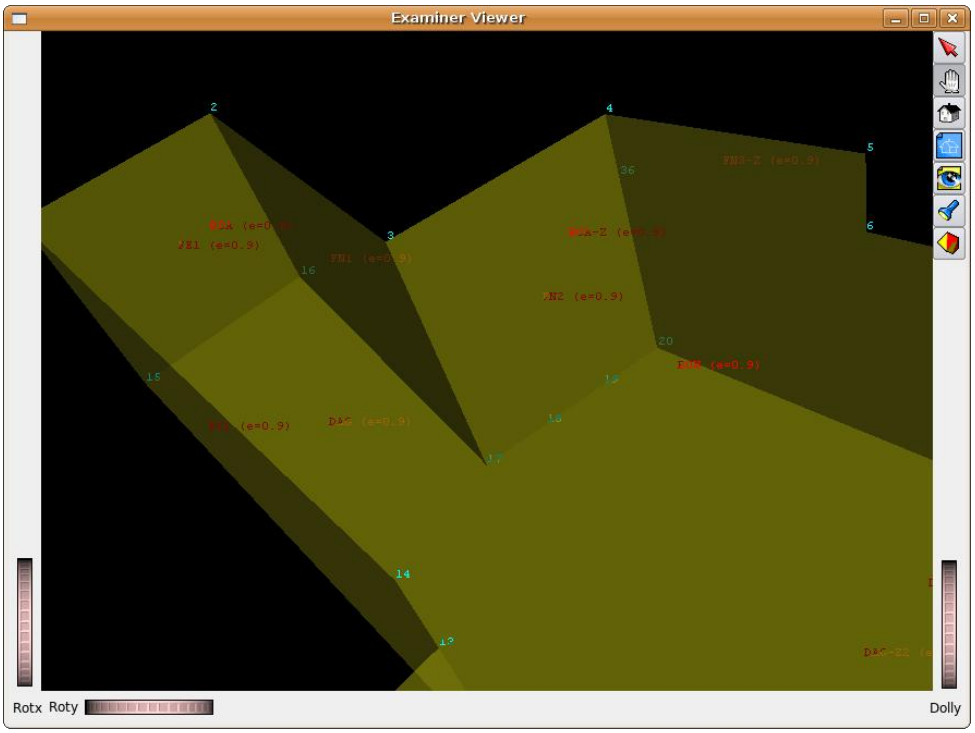

Figure: screenshot of the 'viewer' program for 3D rendering of 3D input data files for View3D.

viewer2d

viewer2d (by John Pye) displays a 2D rendering of the surfaces described by a 2D input data file (see section 3.1). The input data file for view2d is slightly different from that for view3d at present, but we plan to unify these two input data formats into a single format that can serve both types of problems. Currently they're still separate though, but there is no specific documentation of the view2d format. You might find the code in examples/cavity.cpp to be helpful though, however, in attempting to understand the format.

[1]The binary installer for SoQt can be obtained from the ASCEND wiki, at: http://ascendwiki.cheme.cmu.edu/Binary_installers_for_Coin3d_and_SoQt_on_MinGW

Usage: viewer <2Dscenefile>

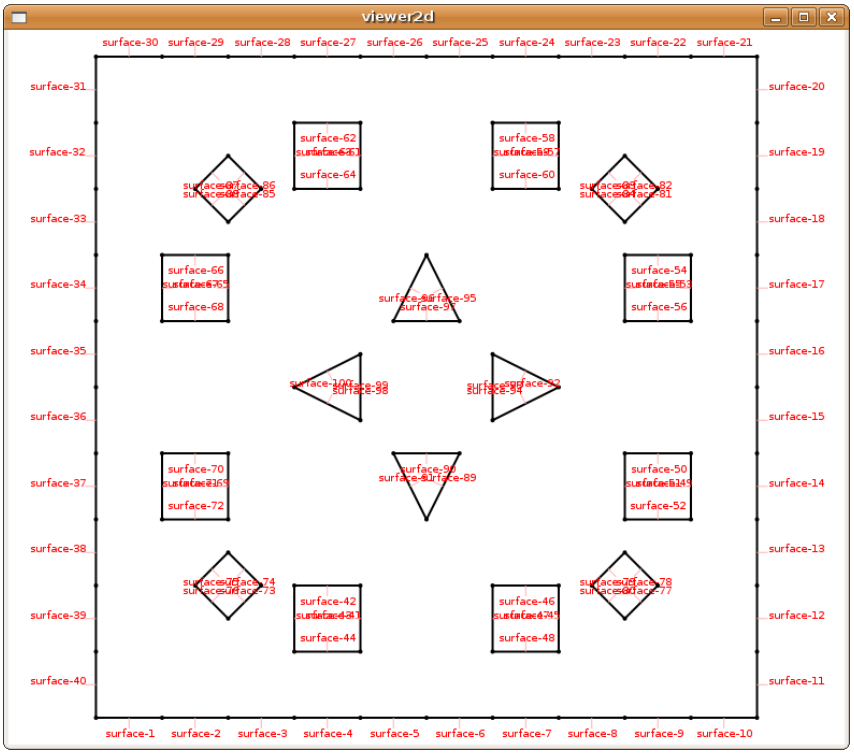

Figure: screenshot of the 'viewer2d' program, which displays the contents of 2D input data files as used by the 'view2d' program for calculating 2D view factors.

vf2asc

Another program called '**vf2asc**' is provided (by John Pye). This program reads the output from either view3d or view2d and generated a model file suitable for reading using the mathematical modelling program ASCEND (http://ascend.cheme.cmu.edu). You can use this to quickly solve the radiosity problem for smallish models (up to ~100 surfaces). This program was used by John Pye in his PhD for calculation of cavity losses from the thermal energy receiver of the CLFR solar energy system. the output of vf2asc is an ASCEND model code that can be 'refined' with the addition of boundary conditions, as desired.

4. INPUT FILE FORMATS

4.1 File of 3D geometry specification (vs3)

This file format is detailed in section 3.1.

Note that we plan to merge this file format with 'vs2'.

4.2 File of 2D geometry specification (vs2)

This format is fairly similar to the 3D data format but contains a few obvious differences. At this stage, the differences are not documented, so try the test problems (test/2d) as well as the example code (example/cavity.cpp).

Note that we plan to merge this file format with 'vs3'.

4.3 File of Temperatures (Tk)

These files are used by the ViewHT program.

Each line of the Tk file consists of a first surface number, a last surface number, and an absolute temperature. The temperature is applied to all surface between the first and the last, inclusive. The last line of the input file is indicated by zeroes for surface numbers.

Example of a Tk file:

```
 1   8   293.15    ! set all 8 surface to 293.15K (20C)
 3   3   283.15    ! reset surface 3 to 283.15K (10C)
 0   0             ! zeroes indicate end of data
```

4.4 File of Emittances (Em)

Usage of these files is not yet documented, and may possibly be irrelevant.

The surface emissivity allows for the processing of different surface emissivites than those included on the VS3d or VS2d files. Each line of the Em file consists of a first surface number, a last surface number, and an emissivity - values between 0.01 and 0.99 are permitted. The emissivity is applied to all surface between the first and the last, inclusive. The last line of the input file is indicated by zeroes for surface numbers.

5. OUTPUT FILE FORMATS

5.1 View Factor File (Lower triangular)

See section 3.2 for information on this file format.

5.2 View Factor File (Square)

This formatting of the view factors is created by ViewSQ. It may be the most convenient basis for creating a data file of view factors to be read by another program after appropriate editing.

Example of a (square) view factor file:

```
! Pinney & Bean test 4: L-shaped room (page A2.14; Table
A2.11) BRE                                                    (a)
 8 1 0                                                        (b)
!  #        area       emit   name                        header
   1       9.000000   0.900   srf-1                          (c)
.0301314 .0992415 .3142475 .0325889 .0307659 .1590225 .1170012
.1170012                                                      (d)
   2       3.000000   0.900   srf-2
.2977246 .0096044 .2732292 .0074348 .0037158 .0922976 .1079968
.1079968
   3       6.000000   0.900   srf-3
.4713712 .1366146 .0252122 .0061597 .0037174 .0488833 .1040208
.1040208
   4       6.000000   0.900   srf-4
.0488833 .0037174 .0061597 .0252122 .1366146 .4713712 .1040208
.1040208
   5       3.000000   0.900   srf-5
.0922976 .0037158 .0074348 .2732292 .0096044 .2977246 .1079968
.1079968
   6       9.000000   0.900   srf-6
.1590225 .0307659 .0325889 .3142475 .0992415 .0301314 .1170012
.1170012
   7       5.000000   0.900   srf-7
.2106022 .0647981 .1248250 .1248250 .0647981 .2106022 .0109327
.0886167
   8       5.000000   0.900   srf-8                          (c)
.2106022 .0647981 .1248250 .1248250 .0647981 .2106022 .0886167
.0109327                                                      (d)
```

5.5 ViewHt Output File

The **ViewHT** output file echoes the view factor file title, notes whether or not emittances were included in the view factors, and reports the heat flux and total heat gain for each surface. The example output below agrees very well with the reported values [4].

Example of ViewHt output:

```
Pinney & Bean test 4: L-shaped room (page A2.14; Table A2.11) BRE
  emittance effects included in the view factors

Srf #        name      area    emit    T [K]    q [W/m^2]      Q [W]
    1        srf-1    9.000   0.900   293.15  -1.706e+001  -1.5350e+002
    2        srf-2    3.000   0.900   293.15  -1.483e+001  -4.4489e+001
    3        srf-3    6.000   0.900   283.15   4.748e+001   2.8488e+002
    4        srf-4    6.000   0.900   293.15  -3.343e-001  -2.0059e+000
    5        srf-5    3.000   0.900   293.15  -4.035e-001  -1.2106e+000
    6        srf-6    9.000   0.900   293.15  -1.769e+000  -1.5919e+001
```

```
7         srf-7     5.000    0.900    293.15   -6.775e+000   -3.3875e+001
8         srf-8     5.000    0.900    293.15   -6.775e+000   -3.3875e+001
                                                               =========
                                                  balance:    -1.421e-014
```

6. REFERENCES

[1] Walton, G.N., "Algorithms for Calculating Radiation View Factors Between Plane Convex Polygons with Obstructions", National Bureau of Standards NBSIR 86-3463, Gaithersburg MD (1986)

[2] Walton, G.N., "Computer Programs for Simulation of Lighting/HVAC Interactions", National Institute of Standards and Technology NISTIR 5322, Gaithersburg MD (1993)

[3] Hottel, H.C. and A.F. Sarofim, *Radiative Transfer*, McGraw-Hill, New York NY (1967)

[4] Pinney, A.A. & M.W. Bean, "A Set of Analytic Tests for Internal Longwave Radiation and View Factor Calculations," Vol II, *An investigation into Analytical and Empirical Validation Techniques for Dynamic Thermal Models of Buildings*, BRE, Garston, Watford, WD2 7JR, UK (1988)

[5] Shapiro, A.B., "FACET -- A Radiation View Factor Computer Code for Axisymmetric, 2D Planar, and 3D Geometries with Shadowing," University of California, Lawrence Livermore National Laboratory, UCID-19887 (1983)