UVMAP Developer Integration Notes

UVMAP can be integrated within other systems relatively easily using C or C++ depending on how the system stores tria-face connectivity and coordinate data. Integration requires access to and installation of the *UVMAP* developer library package file UVMAP_LIB,*.tar.gz or source package UVMAP_SRC,*.tar.gz. Installation of package files is described in the links shown on the SimSys Software page and the developer installation is described specifically in SimSys Developer_Install and Setup Instructions (pdf). Please contact David L. Marcum for assistance. Simple Makefiles to build the library and demonstration program are also included for Linux, MacOS, and Windows. Usage of the SimSys Developer script files, such as simsys_compile, is not required. *UVMAP* also allows for integration with external memory allocation routines, malloc, realloc, and free, and call back functions for registering those routines are provided.

The following describes routines to generate a UV coordinate map for a given tria-face triangulation, find the location of given UV coordinates, and free the required data structure. The routines using EGADS style data include EG_uvmap_gen, EG_uvmap_find_uv, and EG_uvmap_def_free. Alternative routines are also available using AFLR style data, and include uvmap_gen, uvmap_find_uv, and uvmap_def_free.

Generate a UV coordinate map for a given tria-face triangulation (EGADS style data).

int ***EG_uvmapGen*** (int idef, int ntria, int nvert, int set_struct, int verbosity, int *local_idef,
int *tria, double *xyz, double **uv, void **ptr);

INPUT ARGUMENTS

idef	Surface ID label. Not used if set_struct=0.
ntria	Number of tria-faces.
nvert	Number of nodes/vertices.
set_struct	UV mapping data structure flag. If set_struct=1, then create UV mapping data structure. Required to use EG_uvmap_find_uv. If set_struct=0, then do not create UV mapping data structure.
verbosity	Message flag. If verbosity=0, then do not output progress information. If verbosity=1, then output progress information to stdout. If verbosity=2, then output progress and additional CPU usage information to stdout.
local_idef	Local surface ID label for each tria-face of surface idef (ntria in length). If the surface idef is a virtual/composite surface, then it is composed of one or more local surfaces with differing surface ID labels. If the surface idef is a standard surface, then the local surface ID label need not be set as it is the same as that for surface idef. Not used if set_struct=0.
tria	Tria-face connectivity (3*ntria in length).
xyz	XYZ coordinates (3*nvert in length). The XYZ coordinates are used only to determine discontinuous locations on the outer and inner (if any) boundary edges. If the XYZ coordinates are NULL on input, then discontinuity on the outer and inner (if any) boundary edges is not considered.
ptr	UV mapping data structure.

Set ptr to NULL if this is the first call to this routine and use previously set ptr on subsequent calls.
Not used if set_struct=0.

RETURN VALUE

EGADS_SUCCESS	Normal completion without errors.
EGADS_MALLOC	Unable to allocate required memory.
EGADS_UVMAP	An error occurred during UV mapping generation.

OUTPUT ARGUMENTS

uv	Generated UV coordinates (2*nvert in length).
ptr	UV mapping data structure. UV mapping data structure with entry added for surface idef. If the structure ptr is NULL on input, then the structure is allocated and surface idef is added. If the structure ptr is not NULL on input (from a previous call to this routine), then the structure is reallocated and surface idef is added. Not used if set_struct=0. Note that a copy of the local surface ID label, local_idef, connectivity, tria, and UV coordinates, uv, are saved within the UV mapping data structure. Also, note that the tria-face connectivity that is stored within the UV mapping structure may have been reordered for ordering consistency and therefore may differ from that of the input connectivity, tria.

Find location of given UV coordinates (EGADS style data).

int ***EG_uvmapFindUV*** (int idef, double uv[2], void *ptr,
int *local_idef, int *itria, int ivert[3], double s[3]);

INPUT ARGUMENTS

idef	Surface ID label.
uv	UV coordinate location to find (2 in length).
ptr	UV mapping data structure.

RETURN VALUE

EGADS_SUCCESS	Normal completion without errors.
EGADS_NOTFOUND	UV coordinate location was not found.
EGADS_MALLOC	Unable to allocate required memory.
EGADS_UVMAP	An error occurred.

OUTPUT ARGUMENTS

itria	Tria-face index on surface idef of the tria-face that contains the given UV coordinates.
local_idef	Local surface ID label of the tria-face that contains the given UV coordinates.

If the surface idef is a virtual/composite surface, then it is composed of one or more local surfaces with differing surface ID labels. If the surface idef is a standard surface, then the local surface ID label is the same as that for surface idef.

ivert	Node/Vertex of the tria-face that contains the given UV coordinates (3 in length).
s	Linear interpolation shape functions for the tria-face that contains the given UV coordinates (3 in length). For example, given data in array data stored at nodes/vertices of the surface mesh, the interpolated value at the location found can be determined from the following expression.

```
data_intp = s[0]*data[ivert[0]-1] + s[1]*data[ivert[1]-1] + s[2]*data[ivert[2]-1];
```

Free UV mapping data structure (EGADS style data).

void ***EG_uvmapStructFree*** (void *ptr);

INPUT ARGUMENTS

ptr	UV mapping data structure.

Generate a UV coordinate map for a given tria-face triangulation (AFLR style data).

INT_***uvmap_gen*** (INT_ idef, INT_ nbface, INT_ nnode, INT_ set_struct, INT_ verbosity,
INT_ *idibf, INT_3D **inibf, DOUBLE_3D **x, DOUBLE_2D **u, void **ptr);

INPUT ARGUMENTS

idef	Surface ID label. Not used if set_struct=0.
nbface	Number of tria-faces.
nnode	Number of nodes/vertices.
set_struct	UV mapping data structure flag. If set_struct=1, then create UV mapping data structure. Required to use uvmap_find_uv. If set_struct=0, then do not create UV mapping data structure.
verbosity	Message flag. If verbosity=0, then do not output progress information. If verbosity=1, then output progress information to stdout. If verbosity=2, then output progress and additional CPU usage information to stdout.
idibf	Local surface ID label for each tria-face of surface idef (nbface+1 in length). If the surface idef is a virtual/composite surface, then it is composed of one or more local surfaces with differing surface ID labels. If the surface idef is a standard surface, then the local surface ID label need not be set as it is the same as that for surface idef. Not used if set_struct=0.
inibf	Tria-face connectivity (nbface+1 in length).
x	XYZ coordinates (nnode+1 in length). The XYZ coordinates are used only to determine discontinuous locations on the outer and inner (if any) boundary edges. If the XYZ coordinates are NULL on input, then discontinuity on the outer and inner (if any) boundary edges is not considered.
ptr	UV mapping data structure.

Set ptr to NULL if this is the first call to this routine and use previously set ptr on subsequent calls.
Not used if set_struct=0.

RETURN VALUE

0	Normal completion without errors.
>0	An error occurred.

OUTPUT ARGUMENTS

u	Generated UV coordinates (nnode+1 in length).
inibf	Tria-face connectivity (nbface+1 in length). Connectivity may be reordered for ordering consistency. The address may change as input arrays are temporarily reallocated to fill holes if there are inner closed curves.
x	XYZ coordinates (nnode+1 in length). The address may change as input arrays are temporarily reallocated to fill holes if there are inner closed curves.
ptr	UV mapping data structure. UV mapping data structure with entry added for surface idef. If the structure ptr is NULL on input, then the structure is allocated and surface idef is added. If the structure ptr is not NULL on input (from a previous call to this routine), then the structure is reallocated and surface idef is added. Not used if set_struct=0. Note that a copy of the local surface ID label, idibf, connectivity, inibf, and UV coordinates, u, are saved within the UV mapping data structure. Also, note that the tria-face connectivity that is stored within the UV mapping structure may have been reordered for ordering consistency and therefore may differ from that of the input connectivity, inibf.

Find location of given UV coordinates (AFLR style data).

INT_ ***uvmap_find_uv*** (INT_ idef, double u_[2], void *ptr,
INT_ *local_idef, INT_ *ibface, INT_ inode_[3], double s[3]);

INPUT ARGUMENTS

idef	Surface ID label.
u	UV coordinate location to find (2 in length).
ptr	UV mapping data structure.

RETURN VALUE

0	UV coordinate location was found.
-1	UV coordinate location was not found.
>0	An error occurred.

OUTPUT ARGUMENTS

local_idef	Local surface ID label of the tria-face that contains the given UV coordinates. If the surface idef is a virtual/composite surface, then it is composed of one or more local surfaces with differing surface ID labels. If the surface idef is a standard surface, then the local surface ID label is the same as that for surface idef.
ibface	Tria-face index on surface idef of the tria-face that contains the given UV coordinates.
Inode_	Node/Vertex of the tria-face that contains the given UV coordinates (3 in length).
s	Linear interpolation shape functions for the tria-face that contains the given UV coordinates (3 in length). For example, given data in array data stored at nodes/vertices of the surface mesh, the interpolated value at the location found can be determined from the following expression.

data_intp = s[0]*data[inode_ [0] + s[1]*data[inode_ [1]] + s[2]*data[inode_ [2];

Free UV mapping data structure (AFLR style data).

void uvmap_struct_free (void *ptr);

INPUT ARGUMENTS

ptr	UV mapping data structure.

UVMAP Home