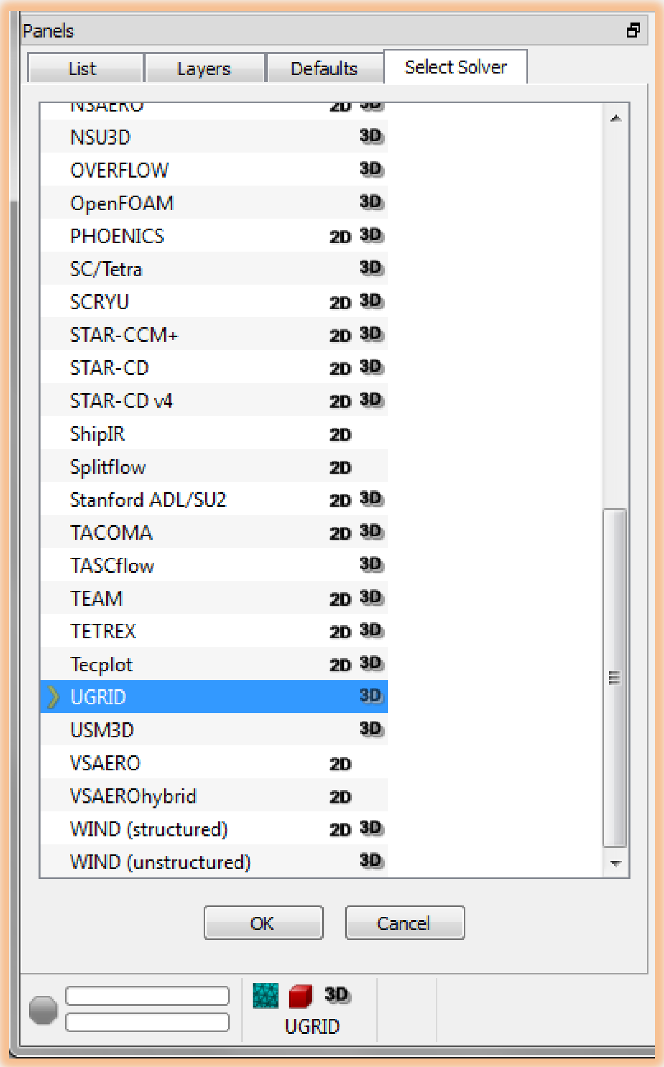
Using Plugins to Streamline Your Process

David Garlisch

Pointwise User Group Meeting

Mar 19, 2013

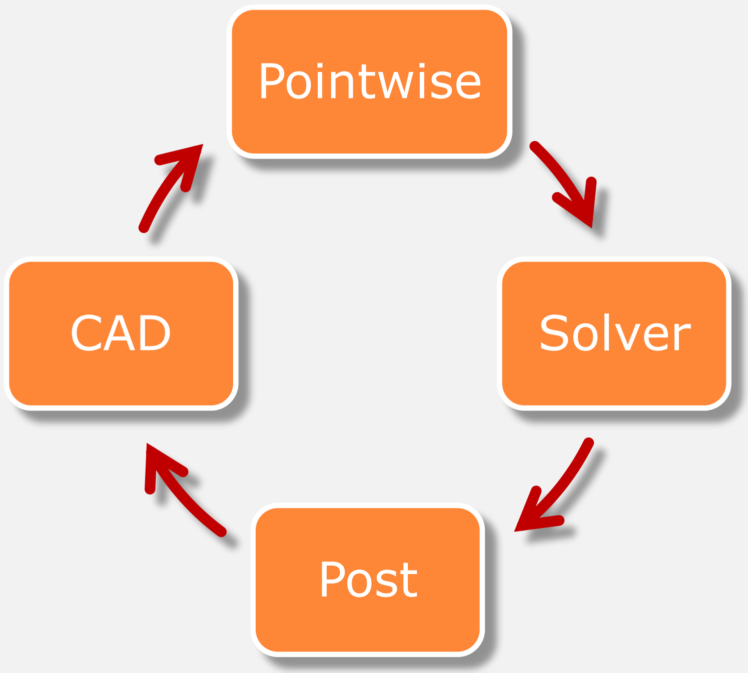


NSAERO	2D	3D
NSU3D		3D
OVERFLOW		3D
OpenFOAM		3D
PHOENICS	2D	3D
SC/Tetra		3D
SCRYU	2D	3D
STAR-CCM+	2D	3D
STAR-CD	2D	3D
STAR-CD v4	2D	3D
ShipIR	2D	
Splitflow	2D	
Stanford ADL/SU2	2D	3D
TACOMA	2D	3D
TASCflow		3D
TEAM	2D	3D
TETREX	2D	3D
Tecplot	2D	3D
UGRID		3D
USM3D		3D
VSAERO	2D	
VSAEROhybrid	2D	
WIND (structured)	2D	3D
WIND (unstructured)		3D

POINTWISE2013 USER GROUP MEETING

Typical CFD Workflow

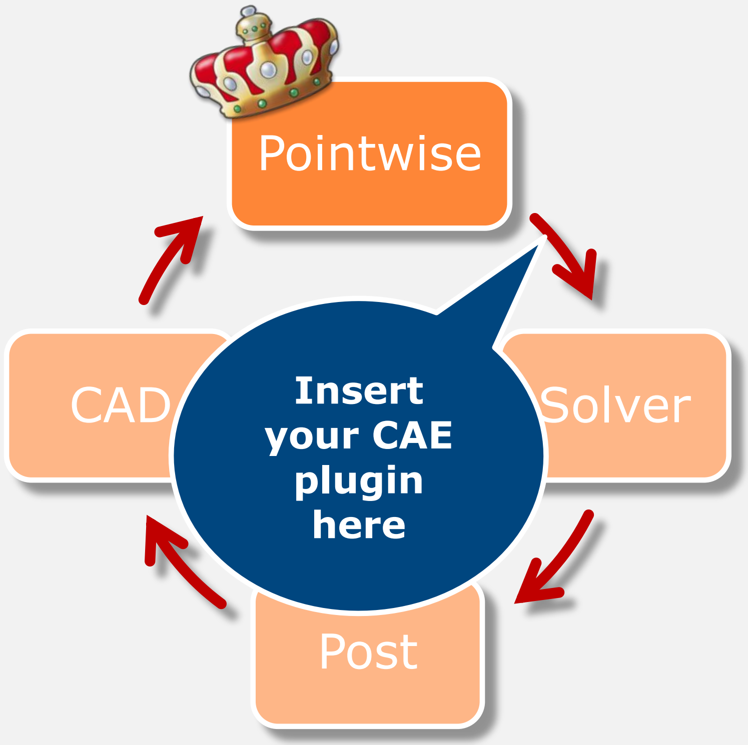
- Involves a suite of 3rd party and proprietary tools
- Data needs to flow from one step to the next
- Sometimes these tools don't play nicely together



- ➢ Non-standard standards
- ➢ Format conversion
- ➢ Byte ordering
- ➢ Data precision
- ➢ Undocumented features (bugs)
- ➢ Legacy systems
- ➢ Wasted disk space
- ➢ Wasted conversion time

Pointwise CAE Plugins

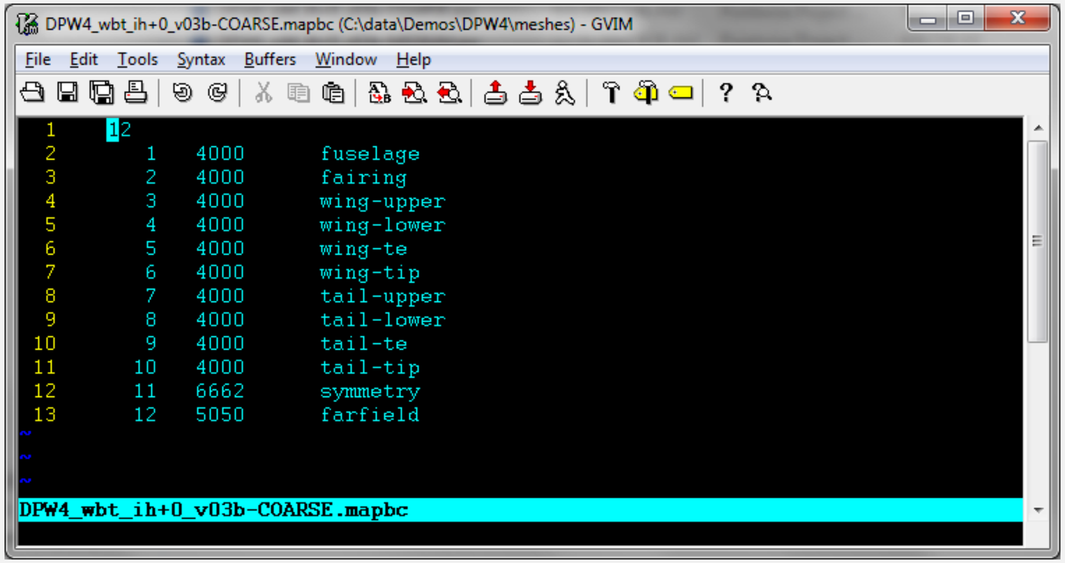
- Gain complete control over the exporting of grid data to your downstream processes



- ➢ Faster time to production
  - We already have many requests
- ➢ Works seamlessly with UI
- ➢ Works seamlessly with Glyph
- ➢ Custom BCs and VCs
- ➢ Data precision
- ➢ Data format (ASCII, binary)
- ➢ Byte order

UGRID (aka AFLR3) Plugin

- Grid file well documented
  - www.simcenter.msstate.edu/docs/solidmesh/ugridformat.html
  - www.simcenter.msstate.edu/docs/solidmesh/ugridconnectivity.html
- BC file not documented
  - Chose FUN3D's .mapbc (http://fun3d.larc.nasa.gov/chapter-4.html)


DPW4_wbt_ih+0_v03b-COARSE.mapbc (C:\data\Demos\DPW4\meshes) - GVIM
File Edit Tools Syntax Buffers Window Help

```
12
     1   4000       fuselage
     2   4000       fairing
     3   4000       wing-upper
     4   4000       wing-lower
     5   4000       wing-te
     6   4000       wing-tip
     7   4000       tail-upper
     8   4000       tail-lower
     9   4000       tail-te
    10   4000       tail-tip
    11   6662       symmetry
    12   5050       farfield
~
~
~
DPW4_wbt_ih+0_v03b-COARSE.mapbc
```

UGRID File Information

Before coding the plugin, gather some solver file information

Grid Type

Str [✓ UnsCell] UnsFace

Supported Dimensions

2D [✓ 3D]

Supported Precisions

[✓ Single] [✓ Double]

Miscellaneous Options

[✓ BCs only] Byteorder

File Destination Type

Filename [✓ Basename] Folder

Supported File Encodings

[✓ ASCII] [✓ Binary] [✓ Unformatted]

Supported Elements

[✓ TRI] [✓ QUAD]

[✓ TET] [✓ PYRAMID] [✓ PRISM] [✓ HEX]

UGRID File Information (continued)

Boundary Conditions

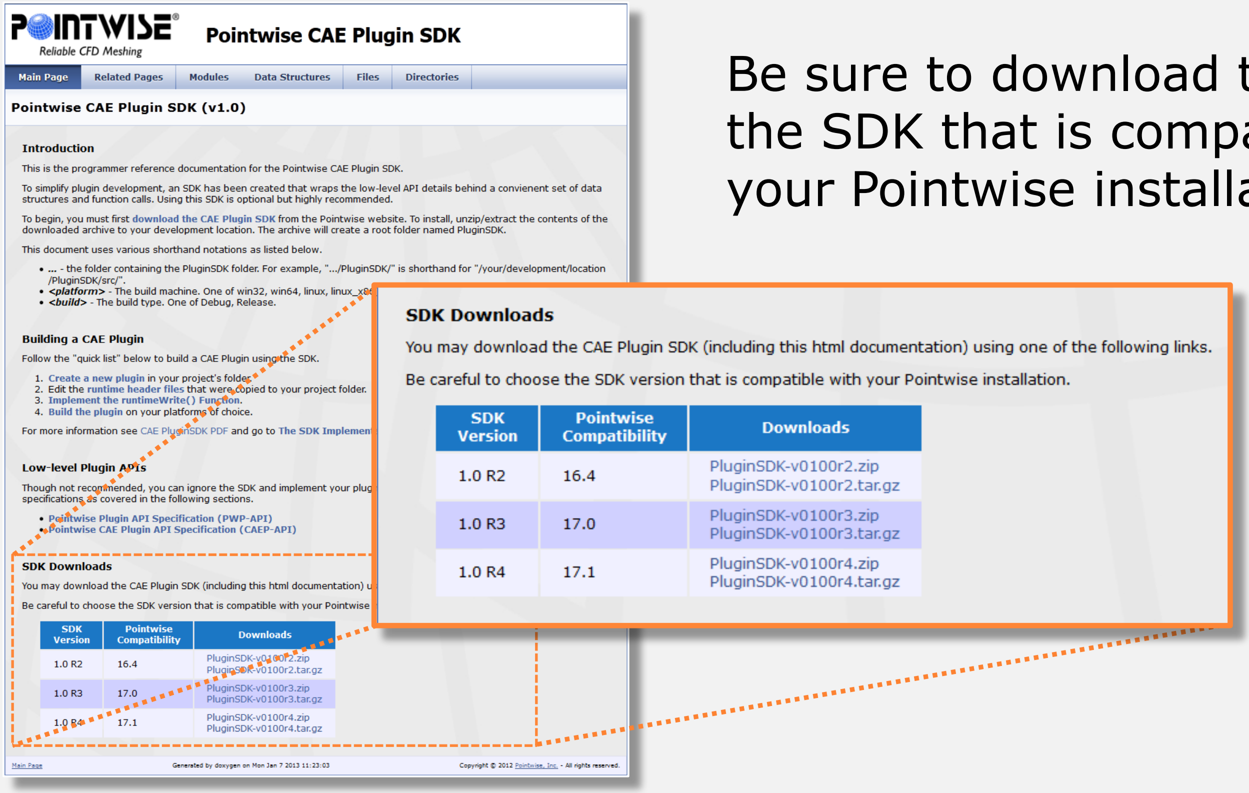
Type	Id
Back pressure (static)	5051
Back pressure (Mach)	5052
Extrapolate	5026
Farfield (Riemann node)	5000
Farfield (Riemann element)	5025
Farfield (freestream)	5050
Inflow (fixed)	7100
Inflow (mass)	7036
Inflow (subsonic)	7011
Outflow (mass)	7031
Outflow (subsonic)	7012
Periodicity	6100
Symmetry Plane (X)	6661
Symmetry Plane (Y)	6662
Symmetry Plane (Z)	6663
Wall (inviscid)	3000
Wall (viscous)	4000

Volume Conditions

Type	Id

Download CAE Plugin SDK

http://www.pointwise.com/plugins/



Pointwise CAE Plugin SDK

Pointwise CAE Plugin SDK (v1.0)

Introduction

This is the programmer reference documentation for the Pointwise CAE Plugin SDK.

To simplify plugin development, an SDK has been created that wraps the low-level API details behind a convienent set of data structures and function calls. Using this SDK is optional but highly recommended.

To begin, you must first **download the CAE Plugin SDK** from the Pointwise website. To install, unzip/extract the contents of the downloaded archive to your development location. The archive will create a root folder named PluginSDK.

This document uses various shorthand notations as listed below.

- ***...*** - the folder containing the PluginSDK folder. For example, ".../PluginSDK/" is shorthand for "/your/development/location /PluginSDK/src/".
- ***<platform>*** - The build machine. One of win32, win64, linux, linux_x8
- ***<build>*** - The build type. One of Debug, Release.

Building a CAE Plugin

Follow the "quick list" below to build a CAE Plugin using the SDK.

1. **Create a new plugin** in your project's folder.
2. Edit the **runtime header files** that were copied to your project folder.
3. **Implement the runtimeWrite() Function**.
4. **Build the plugin** on your platforms of choice.

For more information see CAE PluginSDK PDF and go to **The SDK Implement**

Low-level Plugin APIs

Though not recommended, you can ignore the SDK and implement your plug specifications as covered in the following sections.

- **Pointwise Plugin API Specification (PWP-API)**
- **Pointwise CAE Plugin API Specification (CAEP-API)**

SDK Downloads

You may download the CAE Plugin SDK (including this html documentation) u

Be careful to choose the SDK version that is compatible with your Pointwise

SDK Version	Pointwise Compatibility	Downloads
1.0 R2	16.4	PluginSDK-v0100r2.zip PluginSDK-v0100r2.tar.gz
1.0 R3	17.0	PluginSDK-v0100r3.zip PluginSDK-v0100r3.tar.gz
1.0 R4	17.1	PluginSDK-v0100r4.zip PluginSDK-v0100r4.tar.gz

Main Page Generated by doxygen on Mon Jan 7 2013 11:23:03 Copyright © 2012 Pointwise, Inc. - All rights reserved.

SDK Downloads

You may download the CAE Plugin SDK (including this html documentation) using one of the following links.

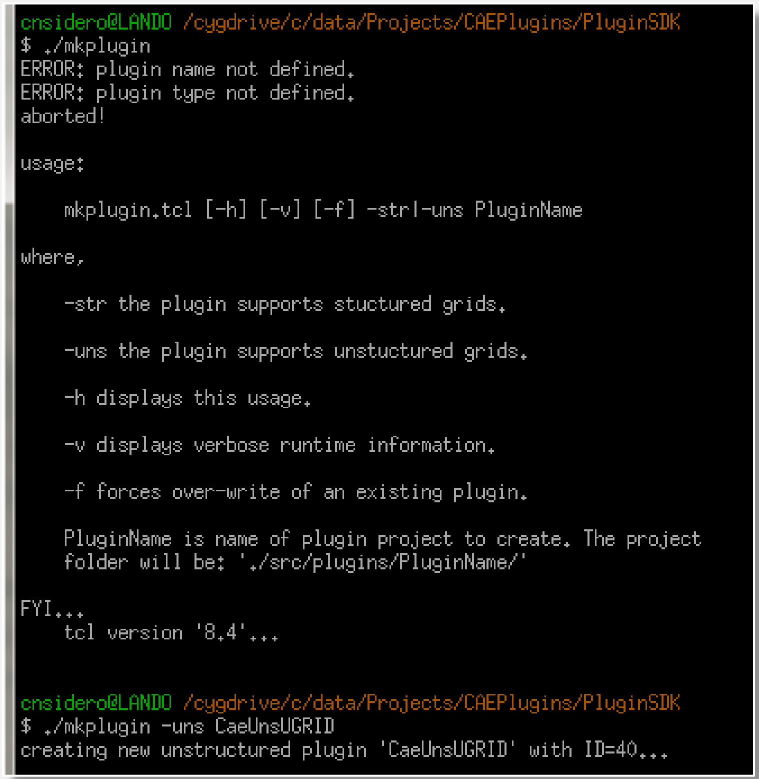
Be careful to choose the SDK version that is compatible with your Pointwise installation.

SDK Version	Pointwise Compatibility	Downloads
1.0 R2	16.4	PluginSDK-v0100r2.zip PluginSDK-v0100r2.tar.gz
1.0 R3	17.0	PluginSDK-v0100r3.zip PluginSDK-v0100r3.tar.gz
1.0 R4	17.1	PluginSDK-v0100r4.zip PluginSDK-v0100r4.tar.gz

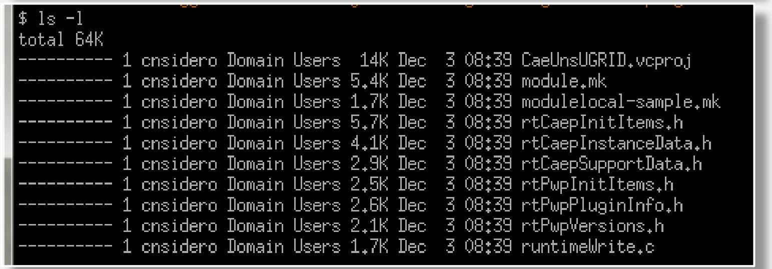
Be sure to download the version of the SDK that is compatible with your Pointwise installation.

Hint: When building for multiple platforms, install the CAE Plugin SDK on a network drive!

Create Empty CAE Plugin



```
cnsidero@LANDO /cygdrive/c/data/Projects/CAEPlugins/PluginSDK
$ ./mkplugin
ERROR: plugin name not defined.
ERROR: plugin type not defined.
aborted!

usage:

    mkplugin.tcl [-h] [-v] [-f] -str|-uns PluginName

where,

    -str the plugin supports stuctured grids.

    -uns the plugin supports unstuctured grids.

    -h displays this usage.

    -v displays verbose runtime information.

    -f forces over-write of an existing plugin.

    PluginName is name of plugin project to create. The project
    folder will be: './src/plugins/PluginName/'

FYI...
    tcl version '8.4'...


cnsidero@LANDO /cygdrive/c/data/Projects/CAEPlugins/PluginSDK
$ ./mkplugin -uns CaeUnsUGRID
creating new unstructured plugin 'CaeUnsUGRID' with ID=40...
```



```
$ ls -l
total 64K
---------- 1 cnsidero Domain Users  14K Dec  3 08:39 CaeUnsUGRID.vcproj
---------- 1 cnsidero Domain Users 5.4K Dec  3 08:39 module.mk
---------- 1 cnsidero Domain Users 1.7K Dec  3 08:39 modulelocal-sample.mk
---------- 1 cnsidero Domain Users 5.7K Dec  3 08:39 rtCaepInitItems.h
---------- 1 cnsidero Domain Users 4.1K Dec  3 08:39 rtCaepInstanceData.h
---------- 1 cnsidero Domain Users 2.9K Dec  3 08:39 rtCaepSupportData.h
---------- 1 cnsidero Domain Users 2.5K Dec  3 08:39 rtPwpInitItems.h
---------- 1 cnsidero Domain Users 2.6K Dec  3 08:39 rtPwpPluginInfo.h
---------- 1 cnsidero Domain Users 2.1K Dec  3 08:39 rtPwpVersions.h
---------- 1 cnsidero Domain Users 1.7K Dec  3 08:39 runtimeWrite.c
```

`mkplugin -uns|-str PluginName`

`$ mkplugin #` ***`returns full usage`***

Preferred **`PluginName`** Convention:

- Starts with *CaeUns* or *CaeStr*
- Ends with format name (UGRID)

Example:

```
$ ./mkplugin –uns CaeUnsUGRID
$ make CaeUnsUGRID-dr
```

Note:

```
$ tclsh # tclsh must be in path
% puts “hello world!”
% hello world!
% exit
$
```

CAE Plugin Runtime Header Files

To define your plugin's capabilities, you must edit the configuration header files that were copied to your plugin’s project folder.

At runtime, this information is loaded by Pointwise. It is used by the GUI and is accessible through Glyph scripts*.

- site.h
- rtPwpVersions.h
- rtPwpPluginInfo.h
- rtPwpInitItems.h
- rtCaepInitItems.h
- rtCaepSupportData.h
- rtCaepInstanceData.h

* See the `pw::Application/CAE Solver Attributes` documentation in the `Glyph2 manual` for more information.

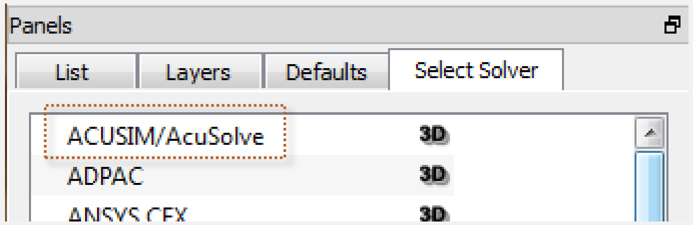
site.h

`PWP_SITE_GROUPID`

- Globally unique, 16-bit, integer site-identifier value.
- Combined with the plugin's id value to construct a globally unique id (GUID).
- If you plan on releasing a plugin outside your site, you need to obtain a **`PWP_SITE_GROUPID`** value from Pointwise support.
- Pointwise will not load plugins properly if GUID conflicts are detected.

`PWP_SITE_GROUPNAME`

- A string representing your company or group.
- All Pointwise developed plugins use the group name "Pointwise".
- Pointwise uses this name to distinguish between similarly named plugins from different authors (group ids).
- A CAE plugin's full name is ***GroupName/CaeName*** (e.g. Alpha/CGNS, Alpha/xml, Beta/CGNS, Beta/xml).
- Defaults to **PWP_GROUPNAME_DEFAULT**.



ACUSIM/AcuSolve 3D
ADPAC 3D
ANSYS CFX 3D

rtPwpVersions.h

`VERSION_PWP_MAJOR`

`VERSION_PWP_MINOR`

- The PWP-API major/minor version values.
- Pointwise uses these values to determine plugin compatibility.
- Typically, you will not change these values. They are set correctly for a given SDK release.

`VERSION_LIB_MAJOR`

`VERSION_LIB_MINOR`

- The software release major/minor version value.
- Typically, you WILL change these values every time a new plugin version is released.
- The version scheme is determined by the plugin author.
- Pointwise does not use these values directly.

rtPwpPluginInfo.h

- Currently, not used by the Pointwise GUI.
- You should only edit the company, support, and copyright strings.
- May be any suitable text.

```
VERSION_PWP_INIT,           // conforms to this PWP-API version
VERSION_LIB_INIT,           // software library release version
"Pointwise, Inc.",          // company/author description
"support.pointwise.com",    // support description (phone, web-link).
"Copyright(c) 2008-2011",   // copyright description
0,                          // number of APIs (set at runtime)
0,                          // default msg callback (set at runtime)
0,                          // spy msg callback (set at runtime)
```

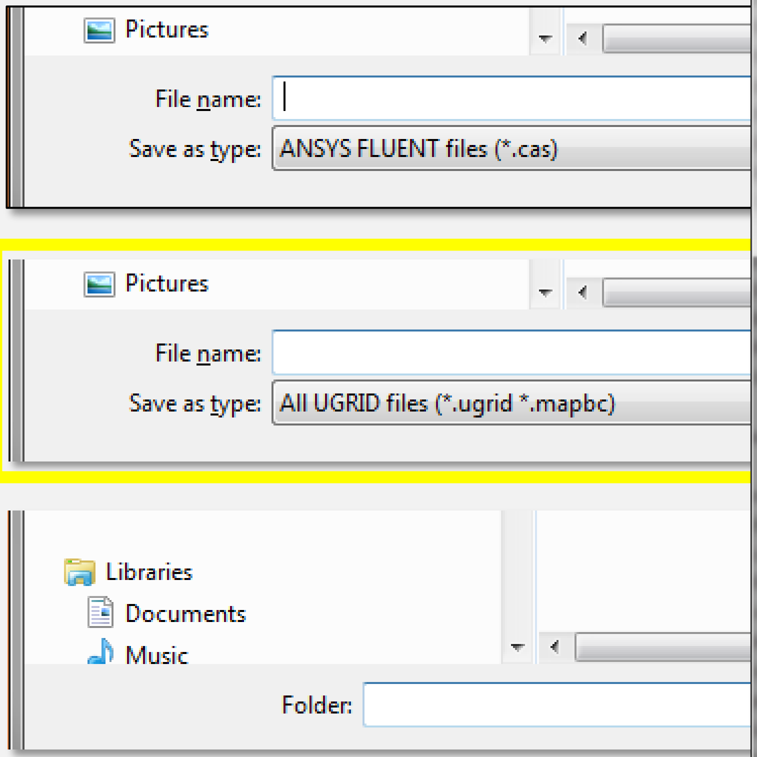
rtPwpInitItems.h

- Pointwise uses these values to determine the plugin type(s).
- Typically, you will not change these values. They are set correctly for a given SDK release.

```
//............................
{
    {PWP_API_PLUGIN "/1.0", VERSION_PWP_INIT},
    0,
},
//............................
{
    {CAEP_API_EXPORT "/1.0", {1,0}},
},
```

rtCaepInitItems.h

File Destination Type



Pictures

File name:

Save as type: ANSYS FLUENT files (*.cas)

Pictures

File name:

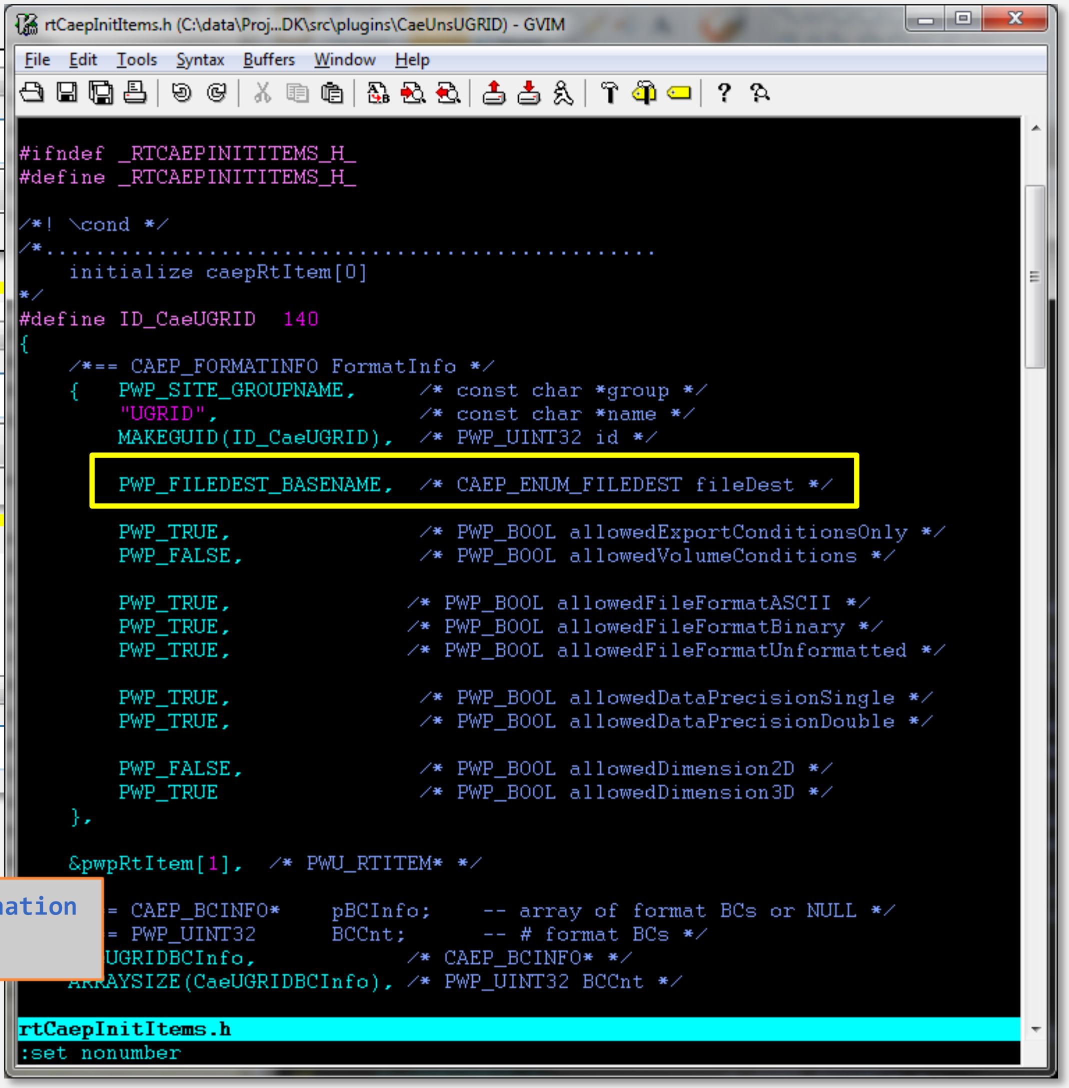
Save as type: All UGRID files (*.ugrid *.mapbc)

Libraries

Documents

Music

Folder:



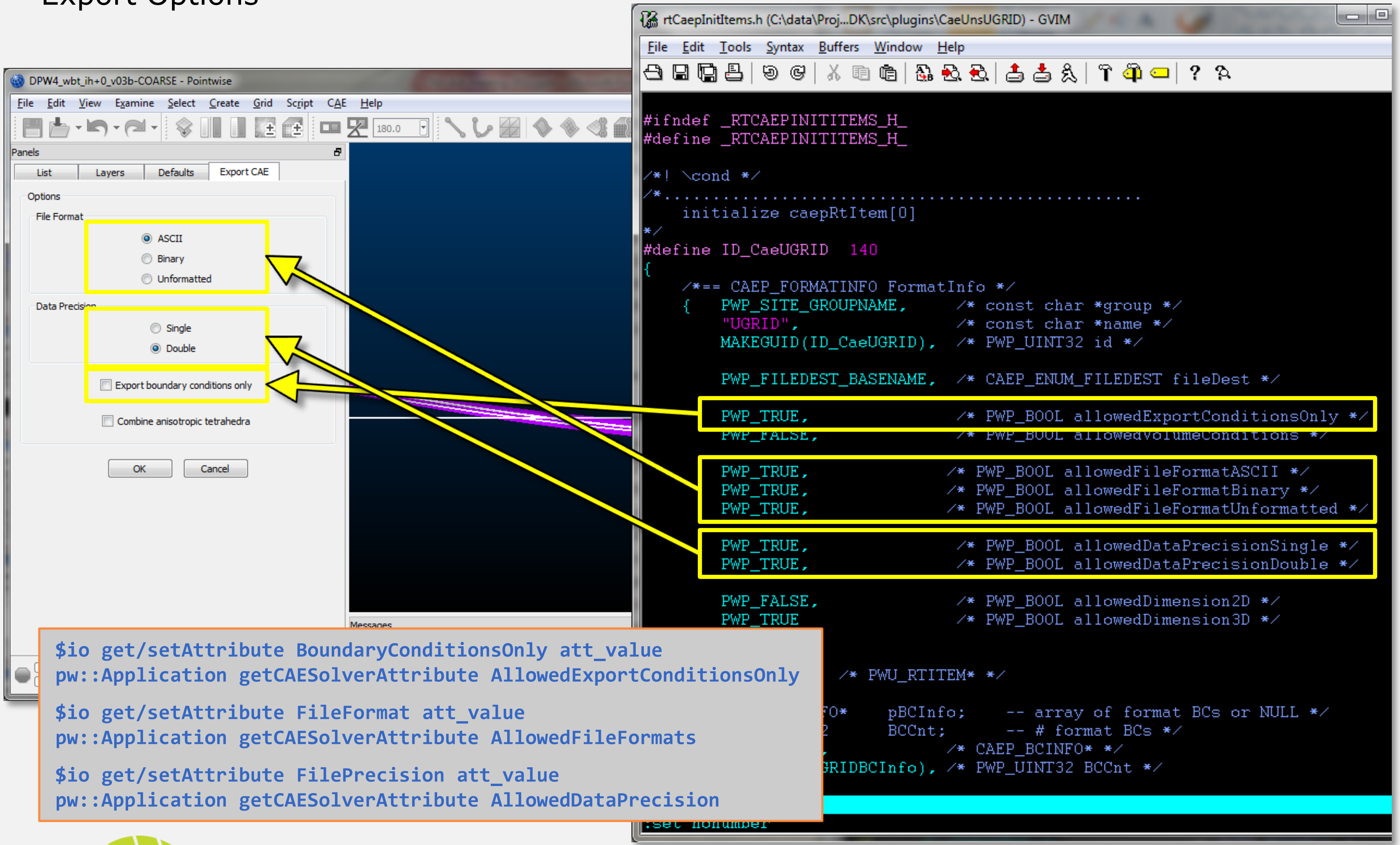
rtCaepInitItems.h (C:\data\Proj...DK\src\plugins\CaeUnsUGRID) - GVIM

```
#ifndef _RTCAEPINITITEMS_H_
#define _RTCAEPINITITEMS_H_

/*! \cond */
/*.............................................
    initialize caepRtItem[0]
*/
#define ID_CaeUGRID  140
{
    /*== CAEP_FORMATINFO FormatInfo */
    {   PWP_SITE_GROUPNAME,      /* const char *group */
        "UGRID",                 /* const char *name */
        MAKEGUID(ID_CaeUGRID),   /* PWP_UINT32 id */

        PWP_FILEDEST_BASENAME,   /* CAEP_ENUM_FILEDEST fileDest */

        PWP_TRUE,                /* PWP_BOOL allowedExportConditionsOnly */
        PWP_FALSE,               /* PWP_BOOL allowedVolumeConditions */

        PWP_TRUE,               /* PWP_BOOL allowedFileFormatASCII */
        PWP_TRUE,               /* PWP_BOOL allowedFileFormatBinary */
        PWP_TRUE,               /* PWP_BOOL allowedFileFormatUnformatted */

        PWP_TRUE,                /* PWP_BOOL allowedDataPrecisionSingle */
        PWP_TRUE,                /* PWP_BOOL allowedDataPrecisionDouble */

        PWP_FALSE,               /* PWP_BOOL allowedDimension2D */
        PWP_TRUE                 /* PWP_BOOL allowedDimension3D */
    },

    &pwpRtItem[1],  /* PWU_RTITEM* */

      = CAEP_BCINFO*    pBCInfo;    -- array of format BCs or NULL */
      = PWP_UINT32      BCCnt;      -- # format BCs */
     UGRIDBCInfo,               /* CAEP_BCINFO* */
    ARRAYSIZE(CaeUGRIDBCInfo), /* PWP_UINT32 BCCnt */

rtCaepInitItems.h
:set nonumber
```

```
pw::Application getCAESolverAttribute FileDestination
$io initialize ?-type file_type? filename
```

rtCaepInitItems.h

Export Options


DPW4_wbt_ih+0_v03b-COARSE - Pointwise
File Edit View Examine Select Create Grid Script CAE Help
Panels
List Layers Defaults Export CAE
Messages
rtCaepInitItems.h (C:\data\Proj...DK\src\plugins\CaeUnsUGRID) - GVIM
File Edit Tools Syntax Buffers Window Help

Options

File Format: [✓ ASCII] Binary Unformatted

Data Precision: Single [✓ Double]

☐ Export boundary conditions only

☐ Combine anisotropic tetrahedra

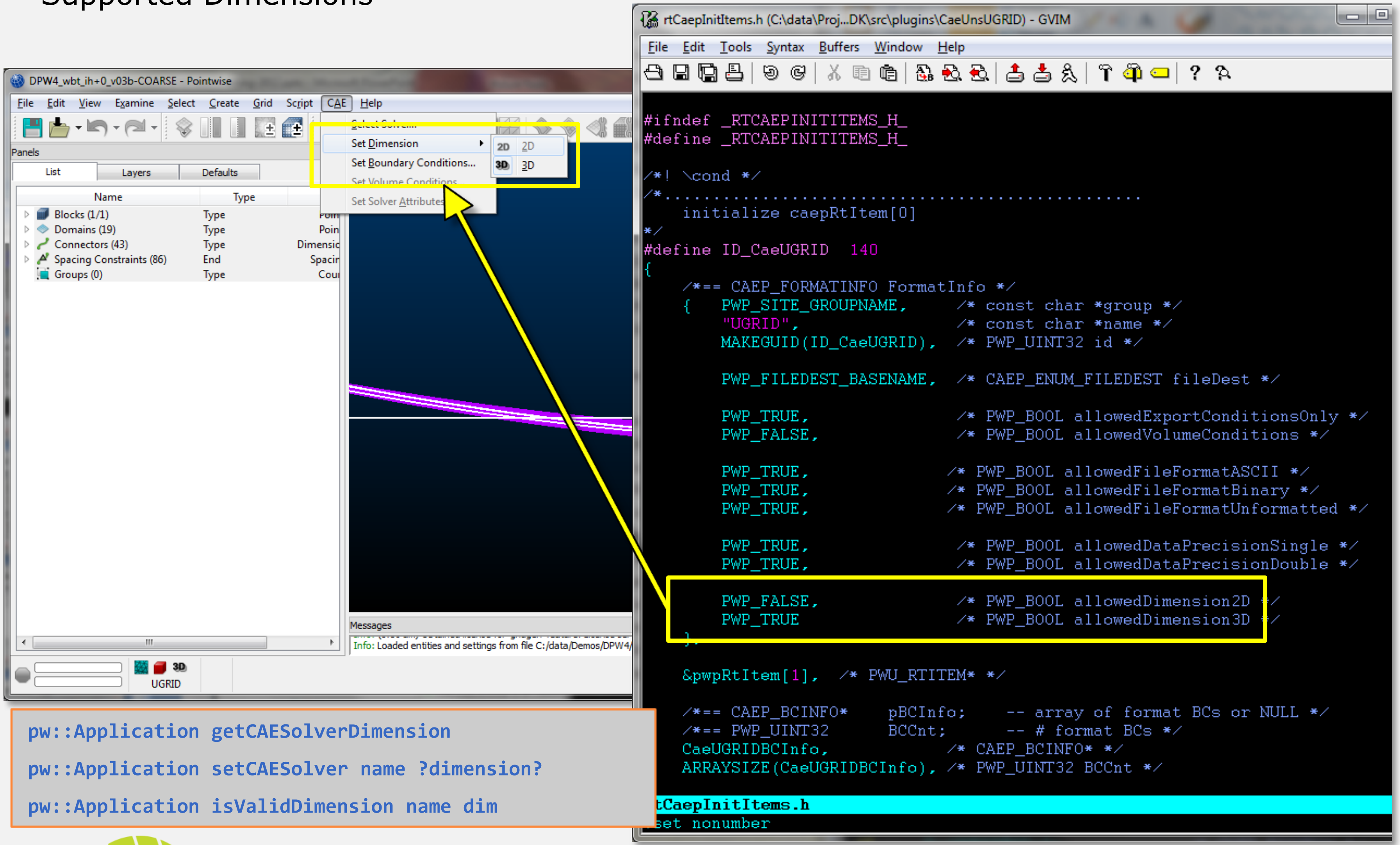
OK Cancel

```
#ifndef _RTCAEPINITITEMS_H_
#define _RTCAEPINITITEMS_H_

/*! \cond */
/*.............................................
    initialize caepRtItem[0]
*/
#define ID_CaeUGRID  140
{
    /*== CAEP_FORMATINFO FormatInfo */
    {   PWP_SITE_GROUPNAME,      /* const char *group */
        "UGRID",                 /* const char *name */
        MAKEGUID(ID_CaeUGRID),   /* PWP_UINT32 id */

        PWP_FILEDEST_BASENAME,   /* CAEP_ENUM_FILEDEST fileDest */

        PWP_TRUE,                /* PWP_BOOL allowedExportConditionsOnly */
        PWP_FALSE,               /* PWP_BOOL allowedVolumeConditions */

        PWP_TRUE,               /* PWP_BOOL allowedFileFormatASCII */
        PWP_TRUE,               /* PWP_BOOL allowedFileFormatBinary */
        PWP_TRUE,               /* PWP_BOOL allowedFileFormatUnformatted */

        PWP_TRUE,                /* PWP_BOOL allowedDataPrecisionSingle */
        PWP_TRUE,                /* PWP_BOOL allowedDataPrecisionDouble */

        PWP_FALSE,               /* PWP_BOOL allowedDimension2D */
        PWP_TRUE                 /* PWP_BOOL allowedDimension3D */

                 /* PWU_RTITEM* */

O*     pBCInfo;     -- array of format BCs or NULL */
2      BCCnt;       -- # format BCs */
,                /* CAEP_BCINFO* */
GRIDBCInfo), /* PWP_UINT32 BCCnt */

:set nonumber
```

```
$io get/setAttribute BoundaryConditionsOnly att_value
pw::Application getCAESolverAttribute AllowedExportConditionsOnly

$io get/setAttribute FileFormat att_value
pw::Application getCAESolverAttribute AllowedFileFormats

$io get/setAttribute FilePrecision att_value
pw::Application getCAESolverAttribute AllowedDataPrecision
```

rtCaepInitItems.h

Supported Dimensions


DPW4_wbt_ih+0_v03b-COARSE - Pointwise
File Edit View Examine Select Create Grid Script CAE Help
Set Dimension
Set Boundary Conditions...
Set Volume Conditions
Set Solver Attributes
2D 2D
3D 3D
Panels
List Layers Defaults
Name Type
Blocks (1/1) Type
Domains (19) Type
Connectors (43) Type
Spacing Constraints (86) End
Groups (0) Type
Messages
Info: Loaded entities and settings from file C:/data/Demos/DPW4/
3D UGRID
rtCaepInitItems.h (C:\data\Proj...DK\src\plugins\CaeUnsUGRID) - GVIM
File Edit Tools Syntax Buffers Window Help

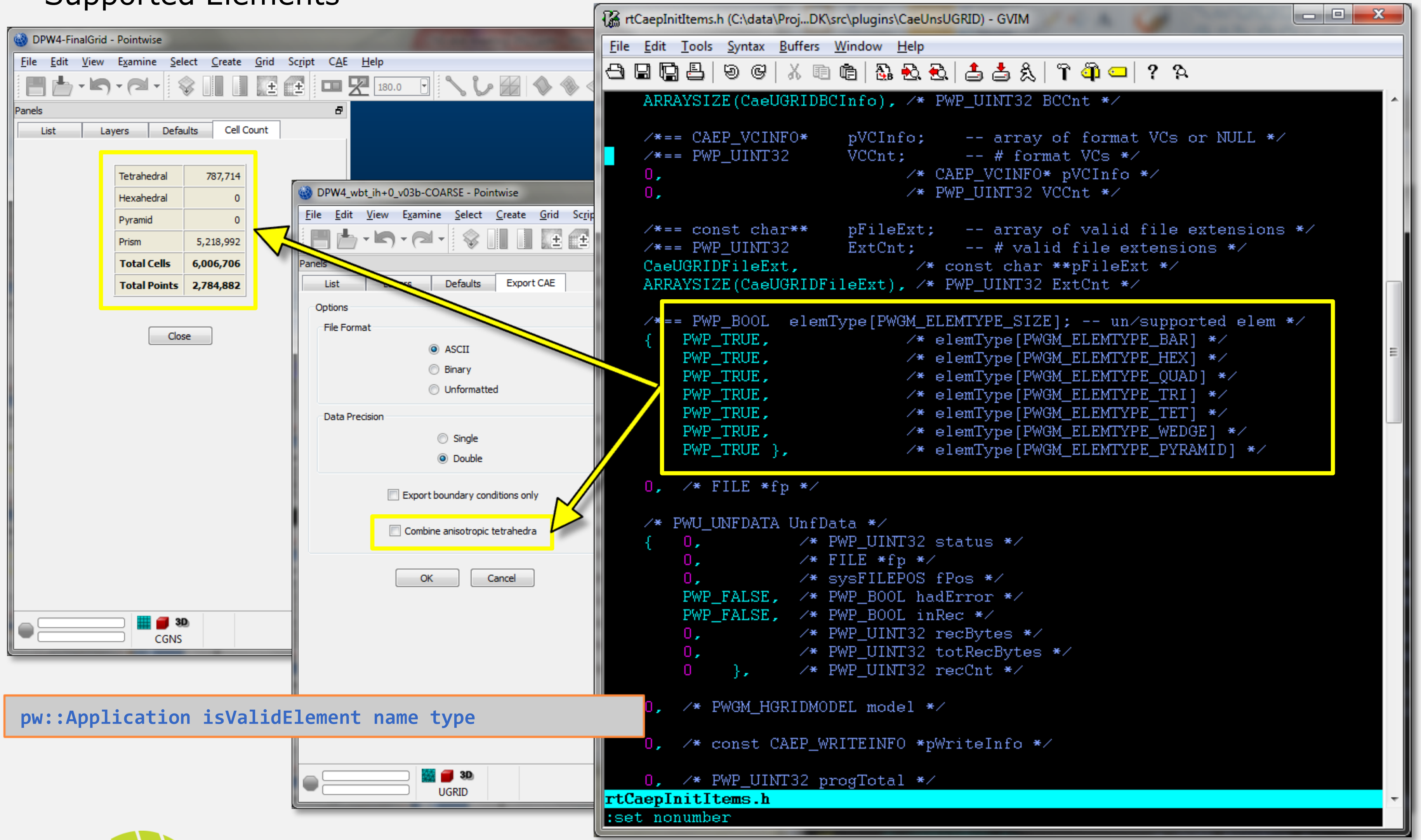
```
#ifndef _RTCAEPINITITEMS_H_
#define _RTCAEPINITITEMS_H_

/*! \cond */
/*.............................................
    initialize caepRtItem[0]
*/
#define ID_CaeUGRID  140
{
    /*== CAEP_FORMATINFO FormatInfo */
    {   PWP_SITE_GROUPNAME,      /* const char *group */
        "UGRID",                 /* const char *name */
        MAKEGUID(ID_CaeUGRID),   /* PWP_UINT32 id */

        PWP_FILEDEST_BASENAME,   /* CAEP_ENUM_FILEDEST fileDest */

        PWP_TRUE,                /* PWP_BOOL allowedExportConditionsOnly */
        PWP_FALSE,               /* PWP_BOOL allowedVolumeConditions */

        PWP_TRUE,               /* PWP_BOOL allowedFileFormatASCII */
        PWP_TRUE,               /* PWP_BOOL allowedFileFormatBinary */
        PWP_TRUE,               /* PWP_BOOL allowedFileFormatUnformatted */

        PWP_TRUE,                /* PWP_BOOL allowedDataPrecisionSingle */
        PWP_TRUE,                /* PWP_BOOL allowedDataPrecisionDouble */

        PWP_FALSE,               /* PWP_BOOL allowedDimension2D */
        PWP_TRUE                 /* PWP_BOOL allowedDimension3D */
    },

    &pwpRtItem[1],  /* PWU_RTITEM* */

    /*== CAEP_BCINFO*    pBCInfo;    -- array of format BCs or NULL */
    /*== PWP_UINT32      BCCnt;      -- # format BCs */
    CaeUGRIDBCInfo,             /* CAEP_BCINFO* */
    ARRAYSIZE(CaeUGRIDBCInfo), /* PWP_UINT32 BCCnt */
tCaepInitItems.h
set nonumber
```

```
pw::Application getCAESolverDimension
pw::Application setCAESolver name ?dimension?
pw::Application isValidDimension name dim
```

POINTWISE2013 USER GROUP MEETING

rtCaepInitItems.h

Supported Elements



DPW4-FinalGrid - Pointwise
File Edit View Examine Select Create Grid Script CAE Help
Panels
List Layers Defaults Cell Count
Close
CGNS
DPW4_wbt_ih+0_v03b-COARSE - Pointwise
List Layers Defaults Export CAE
OK Cancel
UGRID
rtCaepInitItems.h (C:\data\Proj...DK\src\plugins\CaeUnsUGRID) - GVIM
File Edit Tools Syntax Buffers Window Help

Tetrahedral	787,714
Hexahedral	0
Pyramid	0
Prism	5,218,992
Total Cells	**6,006,706**
Total Points	**2,784,882**

Options

File Format: [✓ ASCII] Binary Unformatted

Data Precision: Single [✓ Double]

☐ Export boundary conditions only

☐ Combine anisotropic tetrahedra

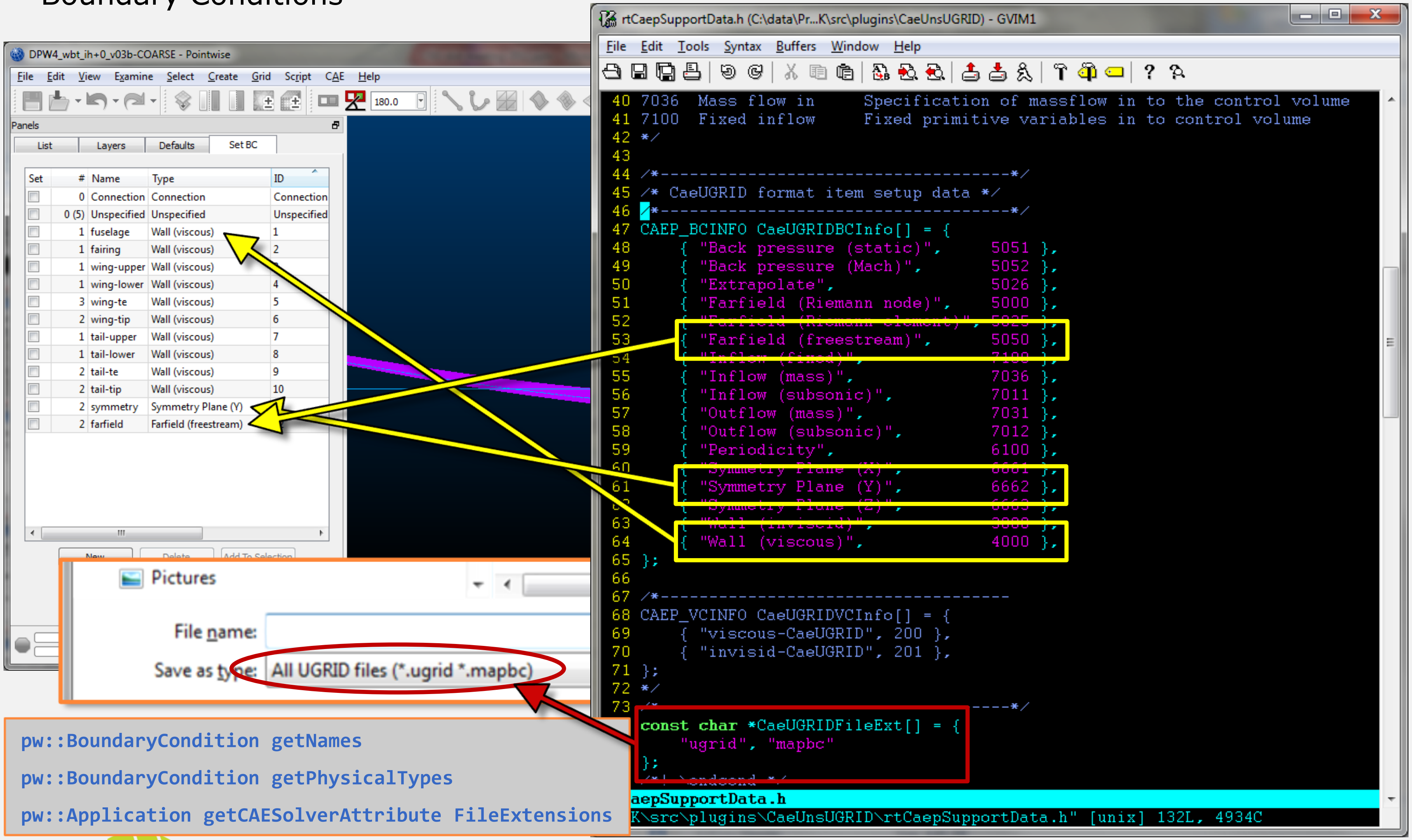
```
    ARRAYSIZE(CaeUGRIDBCInfo), /* PWP_UINT32 BCCnt */

    /*== CAEP_VCINFO*    pVCInfo;    -- array of format VCs or NULL */
    /*== PWP_UINT32      VCCnt;      -- # format VCs */
    0,                         /* CAEP_VCINFO* pVCInfo */
    0,                         /* PWP_UINT32 VCCnt */

    /*== const char**    pFileExt;   -- array of valid file extensions */
    /*== PWP_UINT32      ExtCnt;     -- # valid file extensions */
    CaeUGRIDFileExt,            /* const char **pFileExt */
    ARRAYSIZE(CaeUGRIDFileExt), /* PWP_UINT32 ExtCnt */

    /*== PWP_BOOL  elemType[PWGM_ELEMTYPE_SIZE]; -- un/supported elem */
    {   PWP_TRUE,              /* elemType[PWGM_ELEMTYPE_BAR] */
        PWP_TRUE,              /* elemType[PWGM_ELEMTYPE_HEX] */
        PWP_TRUE,              /* elemType[PWGM_ELEMTYPE_QUAD] */
        PWP_TRUE,              /* elemType[PWGM_ELEMTYPE_TRI] */
        PWP_TRUE,              /* elemType[PWGM_ELEMTYPE_TET] */
        PWP_TRUE,              /* elemType[PWGM_ELEMTYPE_WEDGE] */
        PWP_TRUE },            /* elemType[PWGM_ELEMTYPE_PYRAMID] */

    0,  /* FILE *fp */

    /* PWU_UNFDATA UnfData */
    {   0,            /* PWP_UINT32 status */
        0,            /* FILE *fp */
        0,            /* sysFILEPOS fPos */
        PWP_FALSE,    /* PWP_BOOL hadError */
        PWP_FALSE,    /* PWP_BOOL inRec */
        0,            /* PWP_UINT32 recBytes */
        0,            /* PWP_UINT32 totRecBytes */
        0    },       /* PWP_UINT32 recCnt */

    0,  /* PWGM_HGRIDMODEL model */

    0,  /* const CAEP_WRITEINFO *pWriteInfo */

    0,  /* PWP_UINT32 progTotal */
rtCaepInitItems.h
:set nonumber
```

```
pw::Application isValidElement name type
```

POINTWISE 2013 USER GROUP MEETING

rtCaepSupportData.h

Boundary Conditions



DPW4_wbt_ih+0_v03b-COARSE - Pointwise
File Edit View Examine Select Create Grid Script CAE Help
Panels
List Layers Defaults Set BC
New Delete Add To Selection
Pictures
rtCaepSupportData.h (C:\data\Pr...K\src\plugins\CaeUnsUGRID) - GVIM1
File Edit Tools Syntax Buffers Window Help

Set	#	Name	Type	ID
☐	0	Connection	Connection	Connection
☐	0 (5)	Unspecified	Unspecified	Unspecified
☐	1	fuselage	Wall (viscous)	1
☐	1	fairing	Wall (viscous)	2
☐	1	wing-upper	Wall (viscous)	3
☐	1	wing-lower	Wall (viscous)	4
☐	3	wing-te	Wall (viscous)	5
☐	2	wing-tip	Wall (viscous)	6
☐	1	tail-upper	Wall (viscous)	7
☐	1	tail-lower	Wall (viscous)	8
☐	2	tail-te	Wall (viscous)	9
☐	2	tail-tip	Wall (viscous)	10
☐	2	symmetry	Symmetry Plane (Y)	
☐	2	farfield	Farfield (freestream)	

File name:

Save as type: All UGRID files (*.ugrid *.mapbc)

```
7036  Mass flow in      Specification of massflow in to the control volume
7100  Fixed inflow      Fixed primitive variables in to control volume
*/

/*-----------------------------------*/
/* CaeUGRID format item setup data */
/*-----------------------------------*/
CAEP_BCINFO CaeUGRIDBCInfo[] = {
    { "Back pressure (static)",     5051 },
    { "Back pressure (Mach)",       5052 },
    { "Extrapolate",                5026 },
    { "Farfield (Riemann node)",    5000 },
    { "Farfield (Riemann element)", 5025 },
    { "Farfield (freestream)",      5050 },
    { "Inflow (fixed)",             7100 },
    { "Inflow (mass)",              7036 },
    { "Inflow (subsonic)",          7011 },
    { "Outflow (mass)",             7031 },
    { "Outflow (subsonic)",         7012 },
    { "Periodicity",                6100 },
    { "Symmetry Plane (X)",         6661 },
    { "Symmetry Plane (Y)",         6662 },
    { "Symmetry Plane (Z)",         6663 },
    { "Wall (inviscid)",            3000 },
    { "Wall (viscous)",             4000 },
};

/*-----------------------------------
CAEP_VCINFO CaeUGRIDVCInfo[] = {
    { "viscous-CaeUGRID", 200 },
    { "invisid-CaeUGRID", 201 },
};
*/
/*-----------------------------------*/
const char *CaeUGRIDFileExt[] = {
    "ugrid", "mapbc"
};
/*| \endcond */
aepSupportData.h
K\src\plugins\CaeUnsUGRID\rtCaepSupportData.h" [unix] 132L, 4934C
```

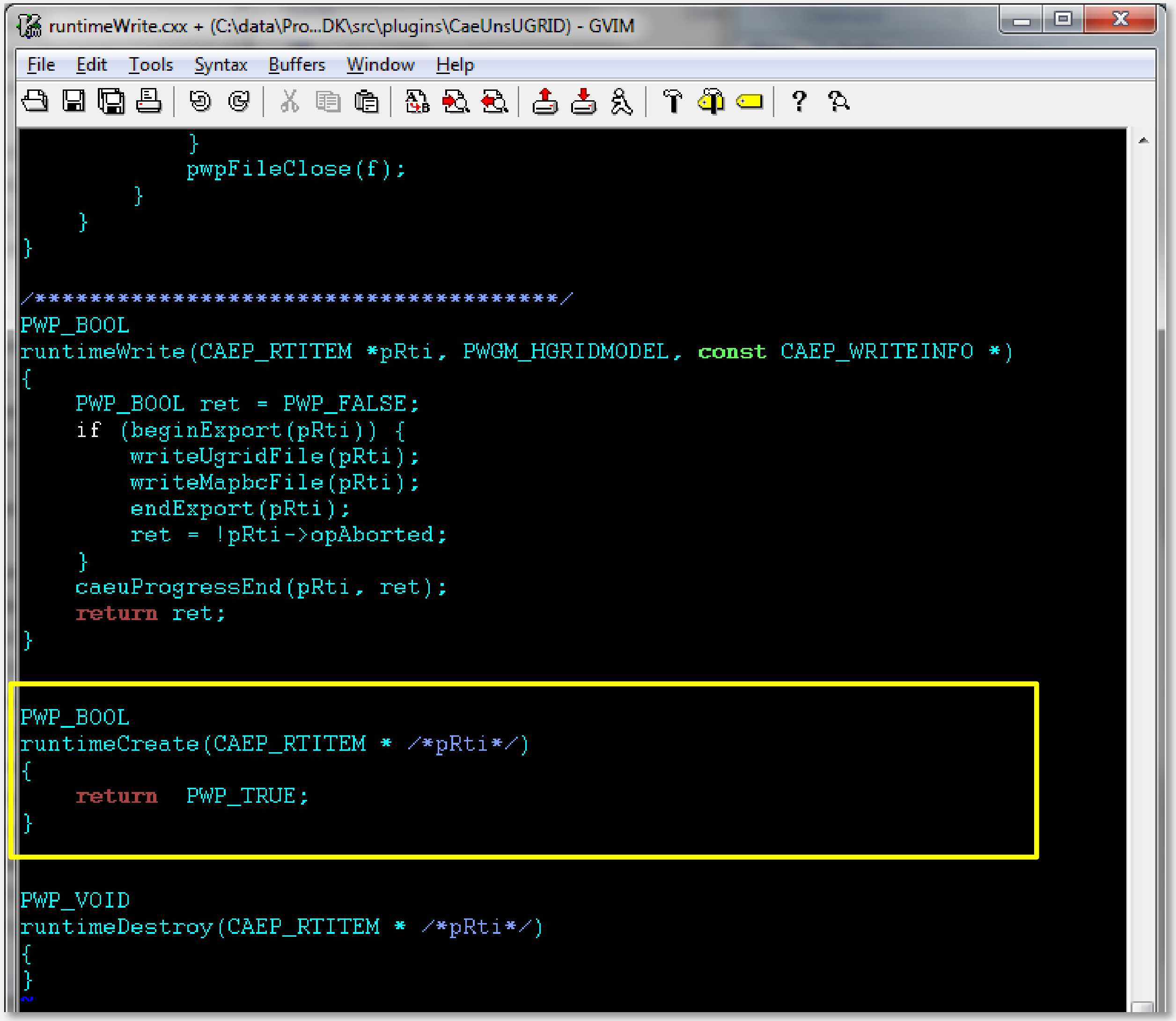
```
pw::BoundaryCondition getNames
pw::BoundaryCondition getPhysicalTypes
pw::Application getCAESolverAttribute FileExtensions
```

POINTWISE2013 USER GROUP MEETING

Implement Runtime Functions

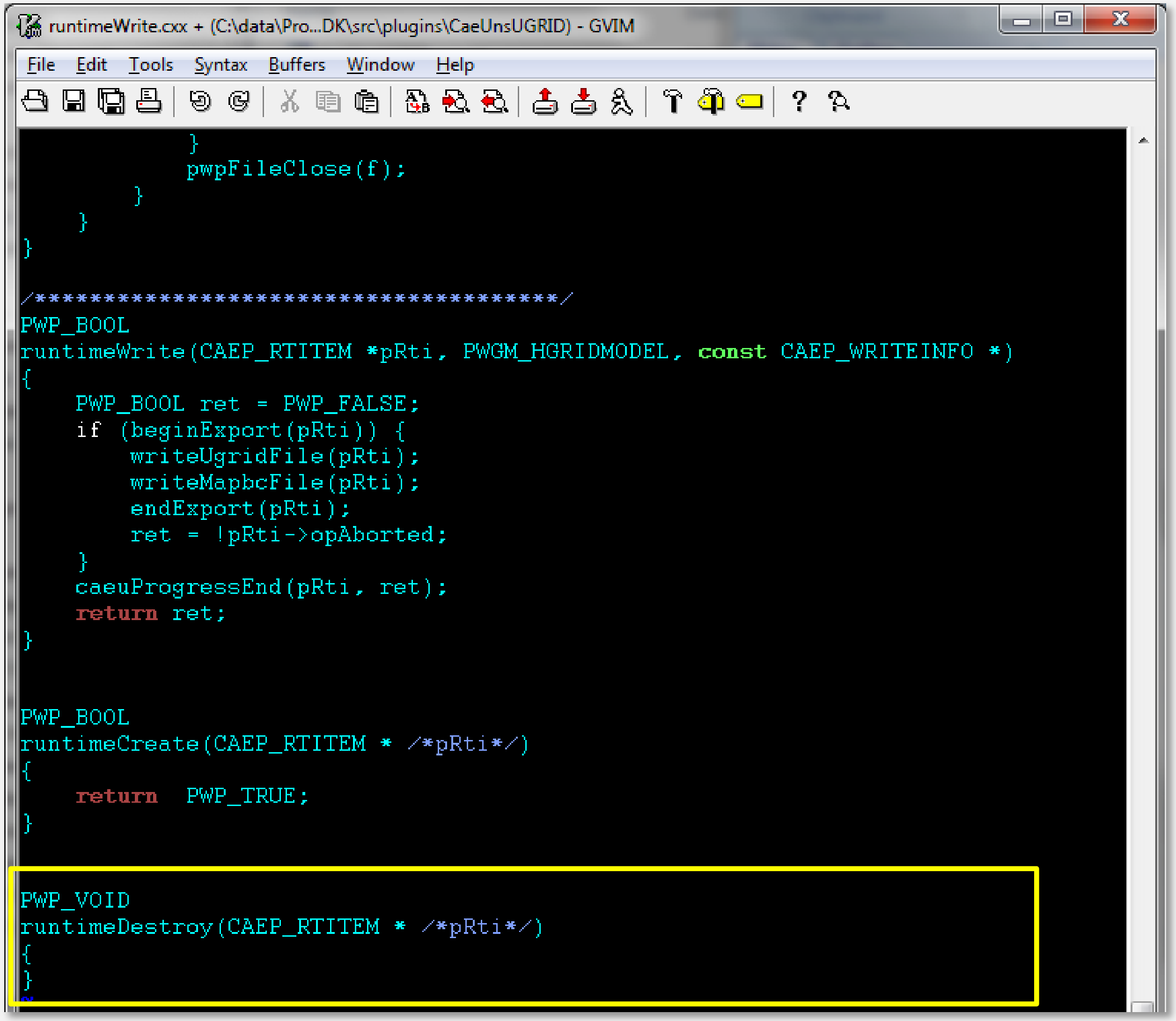
- runtimeCreate()
  - Called each time the plugin is loaded into memory.
  - Typically used to publish solver attributes (optional)
  - Can be reloaded many times.
  - Avoid time consuming operations.
- runtimeDestroy()
  - Called just before plugin is unloaded from memory.
  - Can be reloaded many times.
- runtimeWrite()
  - Called when an export is requested
    - Pointwise GUI (see user manual)
      - `File / Export / CAE…`
    - `pw::CaeExporter` object (see Glyph manual)
      - `pw::Application setCAESolver`
      - `pw::Application begin CaeExport`

UGRID runtimeCreate() Function


runtimeWrite.cxx + (C:\data\Pro...DK\src\plugins\CaeUnsUGRID) - GVIM
File Edit Tools Syntax Buffers Window Help

```
            }
            pwpFileClose(f);
        }
    }
}

/****************************************/
PWP_BOOL
runtimeWrite(CAEP_RTITEM *pRti, PWGM_HGRIDMODEL, const CAEP_WRITEINFO *)
{
    PWP_BOOL ret = PWP_FALSE;
    if (beginExport(pRti)) {
        writeUgridFile(pRti);
        writeMapbcFile(pRti);
        endExport(pRti);
        ret = !pRti->opAborted;
    }
    caeuProgressEnd(pRti, ret);
    return ret;
}

PWP_BOOL
runtimeCreate(CAEP_RTITEM * /*pRti*/)
{
    return  PWP_TRUE;
}

PWP_VOID
runtimeDestroy(CAEP_RTITEM * /*pRti*/)
{
}
```

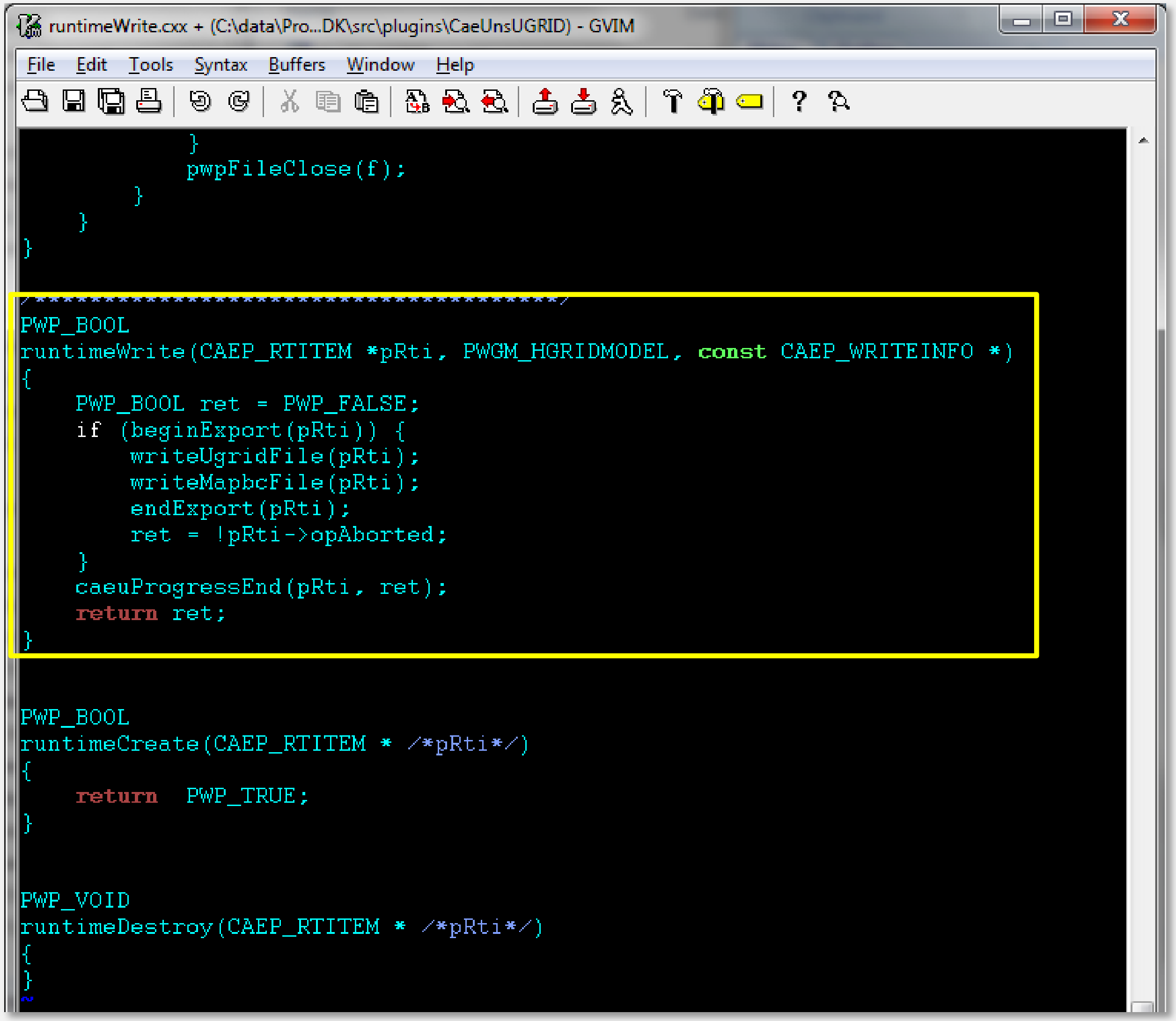
UGRID runtimeDestroy() Function


runtimeWrite.cxx + (C:\data\Pro...DK\src\plugins\CaeUnsUGRID) - GVIM

```
            }
            pwpFileClose(f);
        }
    }
}

/***************************************/
PWP_BOOL
runtimeWrite(CAEP_RTITEM *pRti, PWGM_HGRIDMODEL, const CAEP_WRITEINFO *)
{
    PWP_BOOL ret = PWP_FALSE;
    if (beginExport(pRti)) {
        writeUgridFile(pRti);
        writeMapbcFile(pRti);
        endExport(pRti);
        ret = !pRti->opAborted;
    }
    caeuProgressEnd(pRti, ret);
    return ret;
}


PWP_BOOL
runtimeCreate(CAEP_RTITEM * /*pRti*/)
{
    return  PWP_TRUE;
}


PWP_VOID
runtimeDestroy(CAEP_RTITEM * /*pRti*/)
{
}
```

POINTWISE2013 USER GROUP MEETING

UGRID runtimeWrite() Function


runtimeWrite.cxx + (C:\data\Pro...DK\src\plugins\CaeUnsUGRID) - GVIM
File Edit Tools Syntax Buffers Window Help

```
            }
            pwpFileClose(f);
        }
    }
}

/***************************************/
PWP_BOOL
runtimeWrite(CAEP_RTITEM *pRti, PWGM_HGRIDMODEL, const CAEP_WRITEINFO *)
{
    PWP_BOOL ret = PWP_FALSE;
    if (beginExport(pRti)) {
        writeUgridFile(pRti);
        writeMapbcFile(pRti);
        endExport(pRti);
        ret = !pRti->opAborted;
    }
    caeuProgressEnd(pRti, ret);
    return ret;
}


PWP_BOOL
runtimeCreate(CAEP_RTITEM * /*pRti*/)
{
    return  PWP_TRUE;
}


PWP_VOID
runtimeDestroy(CAEP_RTITEM * /*pRti*/)
{
}
~
```

POINTWISE2013 USER GROUP MEETING

UGRID runtimeWrite() Function

continued

- A simple function body is in the file

  `.../PluginSDK/src/plugins/CaeUnsUGRID/runtimeWrite.cxx`
- When a CAE export is initiated, Pointwise will:

  Populate a **`CAEP_RTITEM`** structure.

  Prepare a **`PWGM_HGRIDMODEL`** database.

  Populate a **`CAEP_WRITEINFO`** structure.

  Call **`runtimeWrite(CAEP_RTITEM *pRti, PWGM_HGRIDMODEL model, const CAEP_WRITEINFO *pWriteInfo);`**
- In **`runtimeWrite()`**, the plugin can access

  The runtime data using the **`pRti`** param.

  The grid data using the **`model`** param.

  The export options using the **`pWriteInfo`** param.
- What is the Pointwise Grid Model?

The Pointwise Grid Model

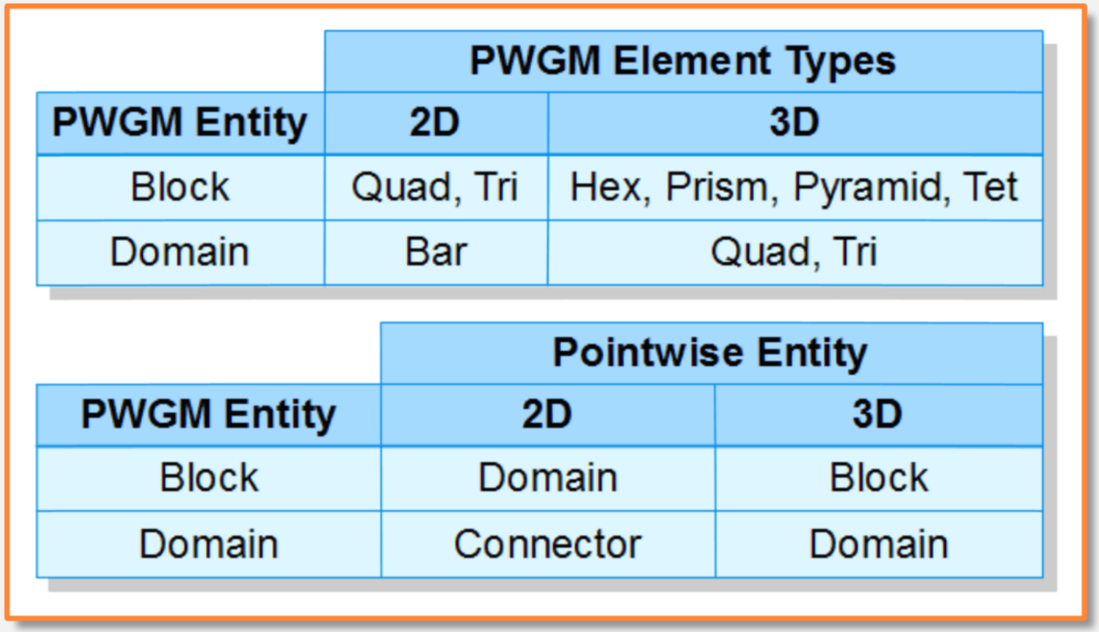
PWGM (See appendix A)

- A filtered view of the native grid data used internally by Pointwise.
- This view will stay consistent over time to reduce (eliminate?) exporter rewrites.
- Handles the construction of non-native grid data (faces).
- PWGM provides 2 topological representations
  - Multi-block structured
  - Multi-block unstructured
    - cell centric
    - face centric
- A plugin can only export as structured or unstructured. Mixed block topology in the same plugin is not supported.

The Pointwise Grid Model

PWGM (See appendix A)

- 2D and 3D grid data accessed using the same functions.
- Dimensionality is only relevant when interpreting element data.



	PWGM Element Types	
PWGM Entity	**2D**	**3D**
Block	Quad, Tri	Hex, Prism, Pyramid, Tet
Domain	Bar	Quad, Tri

	Pointwise Entity	
PWGM Entity	**2D**	**3D**
Block	Domain	Block
Domain	Connector	Domain

The UGRID Export

- Access the Pointwise grid model and export it in the UGRID format.
- Must understand the UGRID file format to write the ***.ugrid*** file.
- Must understand the FUN3D BC file format to write the ***.mapbc*** file.

The .ugrid File

```
{ # Pnts, # Surf Trias, # Surf Quads, # Tets, # Pyrs, # Prism, # Hexs }

{ [ ( X1, Y1, Z1 ),
  ( X2, Y2, Z2 ),
  ...
  ( Xn, Yn, Zn) ] }
```

← coordinate definition

```
{ [ ( TRIA1_P1,
    TRIA1_P2,
    TRIA1_P3 ),
  ( TRIA2_P1,
    TRIA2_P2,
    TRIA2_P3 ),
  ...
  ( TRIA(#Trias)_P1,
    TRIA(#Trias)_P2,
    TRIA(#Trias)_P3 ) ]
```

← tri connectivity

```
 [ ( QUAD1_P1,
    QUAD1_P2,
    QUAD1_P3,
    QUAD1_P4 ),
  ( QUAD2_P1,
    QUAD2_P2,
    QUAD2_P3,
    QUAD1_P4 ),
  ...
  ( QUAD(#Quads)_P1,
    QUAD(#Quads)_P2,
    QUAD(#Quads)_P3 ),
    QUAD(#Quads)_P4 ]
```

← quad connectivity

```
 [ Bndy1 ID,
  Bndy2 ID,
  ...
  Bndy( # Surf Trias + # Surf Quads) ID ]
```

← element-patch association

```
[ ( TET1_P1,
    TET1_P2,
    TET1_P3,
    TET1_P4 ),
  ( TET2_P1,
    TET2_P2,
    TET2_P3,
    TET1_P4 ),
  ...
  ( TET(#Tets)_P1,
    TET(#Tets)_P2,
    TET(#Tets)_P3 ),
    TET(#Tets)_P4 ]
```

← tet connectivity

```
      ...
```

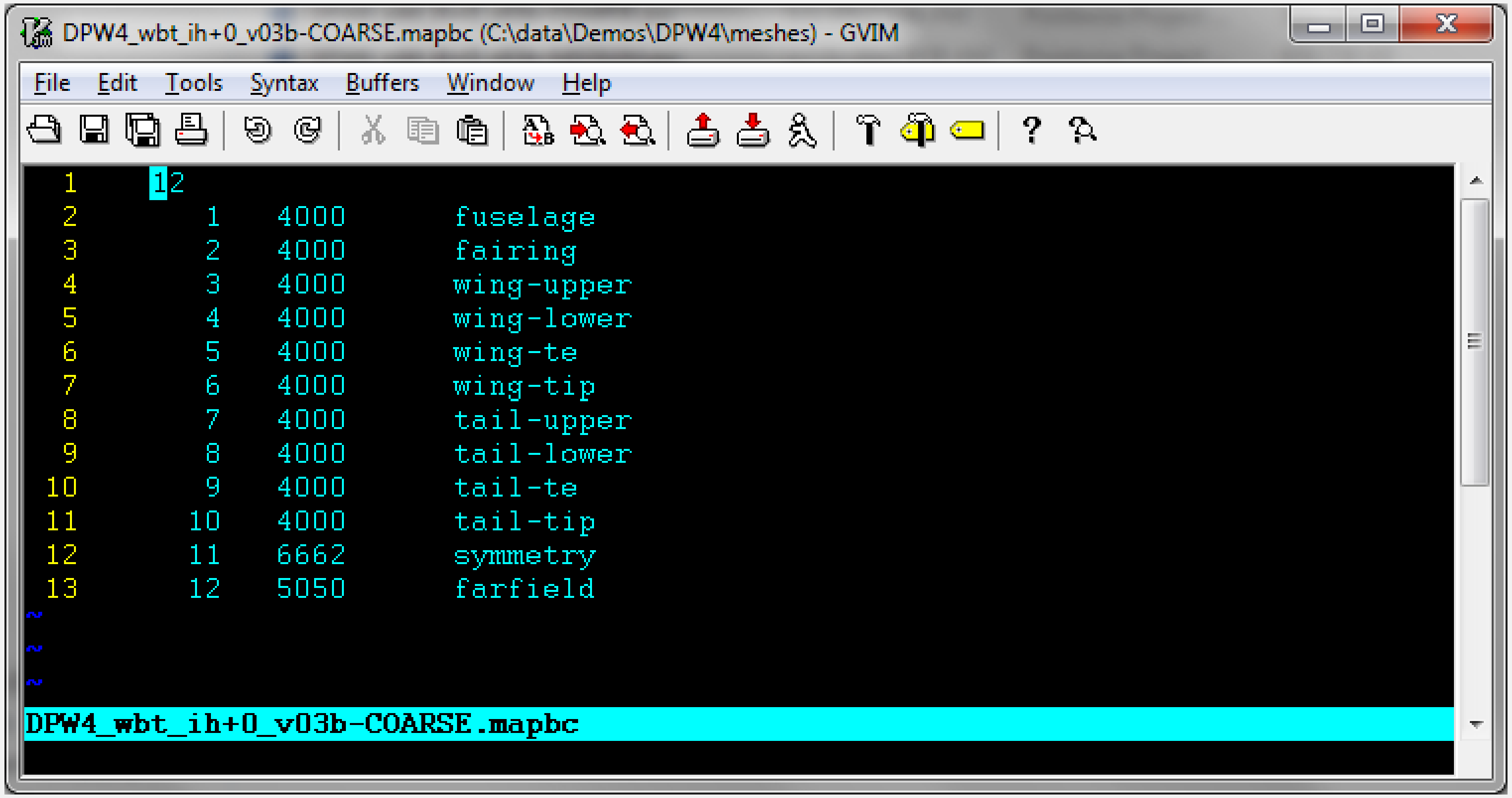
← pyr/prism connectivity

```
 [ ( HEX1_P1,
    HEX1_P2,
    HEX1_P3,
    HEX1_P4 HEX1_P5,
    HEX1_P6, HEX1_P7,
    HEX1_P8 ),
  ( HEX2_P1,
    HEX2_P2,
    HEX2_P3,
    HEX1_P4,
    HEX2_P5,
    HEX2_P6,
    HEX2_P7,
    HEX2_P8  ),
  ...
  ( HEX(#Hexes)_P1,
    HEX(#Hexes)_P2,
    HEX(#Hexes)_P3,
    HEX(#Hexes)_P4,
    HEX(#Hexes)_P5,
    HEX(#Hexes)_P6,
    HEX(#Hexes)_P7,
    HEX(#Hexes)_P8 ) ] }
```

← hex connectivity

The .mapbc File

```
#BC
[ ( BC1_Id      BC1_TYPE      BC1_Name )
  ( BC2_Id      BC2_TYPE      BC2_Name )
   ...
  ( BC(#BC)_Id  BC(#BC)_TYPE  BC(#BC)_Name ) ]
```

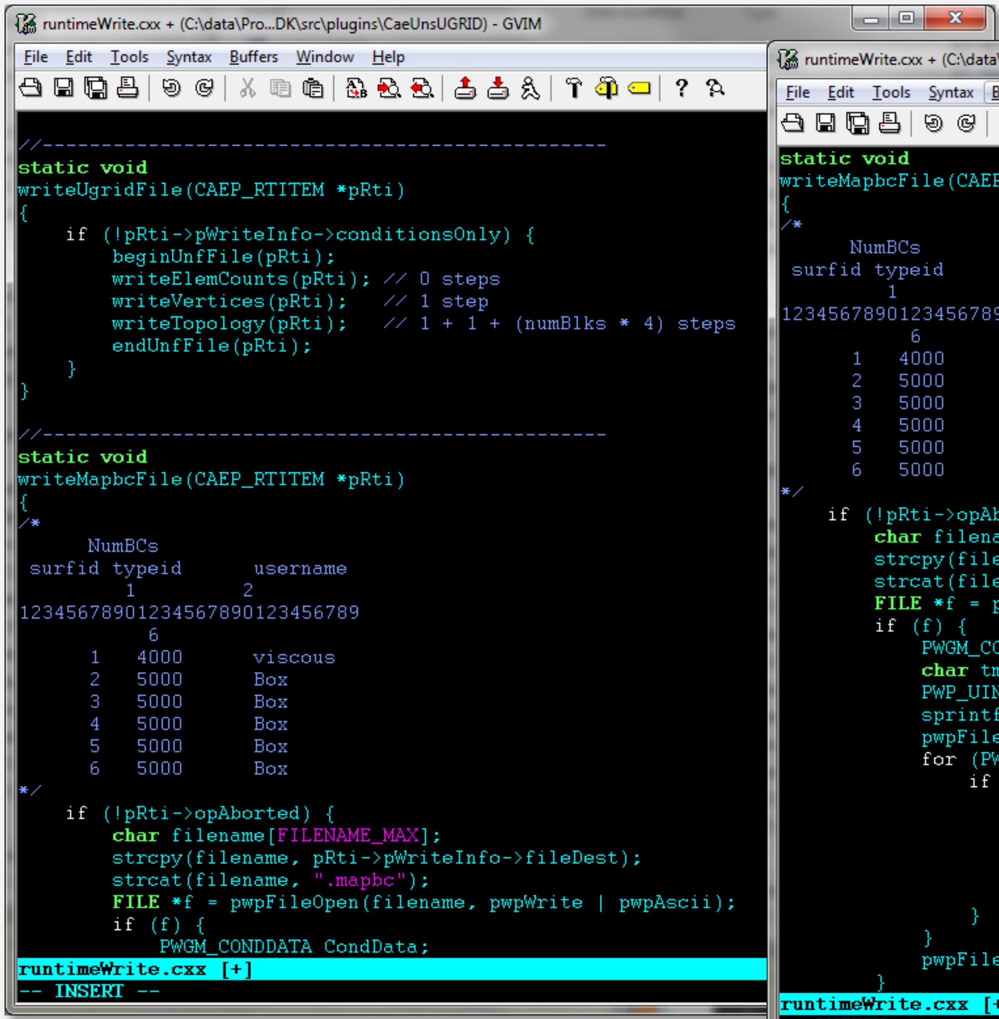

DPW4_wbt_ih+0_v03b-COARSE.mapbc (C:\data\Demos\DPW4\meshes) - GVIM
File Edit Tools Syntax Buffers Window Help

```
12
  1   4000     fuselage
  2   4000     fairing
  3   4000     wing-upper
  4   4000     wing-lower
  5   4000     wing-te
  6   4000     wing-tip
  7   4000     tail-upper
  8   4000     tail-lower
  9   4000     tail-te
 10   4000     tail-tip
 11   6662     symmetry
 12   5050     farfield
```

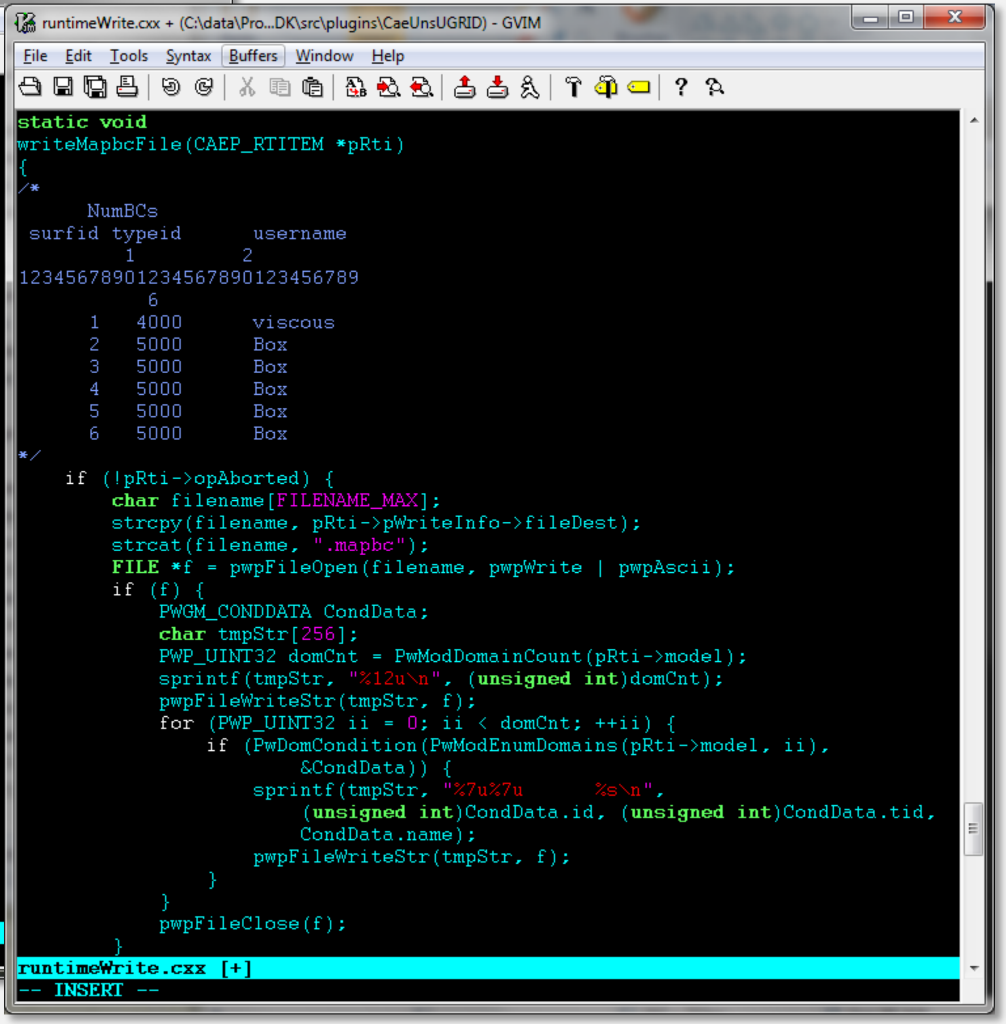
DPW4_wbt_ih+0_v03b-COARSE.mapbc

runTimeWrite.cxx

Write Files


runtimeWrite.cxx + (C:\data\Pro...DK\src\plugins\CaeUnsUGRID) - GVIM
File
Edit
Tools
Syntax
Buffers
Window
Help

```
//-----------------------------------------------
static void
writeUgridFile(CAEP_RTITEM *pRti)
{
    if (!pRti->pWriteInfo->conditionsOnly) {
        beginUnfFile(pRti);
        writeElemCounts(pRti); // 0 steps
        writeVertices(pRti);   // 1 step
        writeTopology(pRti);   // 1 + 1 + (numBlks * 4) steps
        endUnfFile(pRti);
    }
}

//-----------------------------------------------
static void
writeMapbcFile(CAEP_RTITEM *pRti)
{
/*
      NumBCs
 surfid typeid      username
         1         2
12345678901234567890123456789
           6
      1   4000      viscous
      2   5000      Box
      3   5000      Box
      4   5000      Box
      5   5000      Box
      6   5000      Box
*/
    if (!pRti->opAborted) {
        char filename[FILENAME_MAX];
        strcpy(filename, pRti->pWriteInfo->fileDest);
        strcat(filename, ".mapbc");
        FILE *f = pwpFileOpen(filename, pwpWrite | pwpAscii);
        if (f) {
            PWGM_CONDDATA CondData;
runtimeWrite.cxx [+]
-- INSERT --
```


runtimeWrite.cxx + (C:\data\Pro...DK\src\plugins\CaeUnsUGRID) - GVIM
File
Edit
Tools
Syntax
Buffers
Window
Help

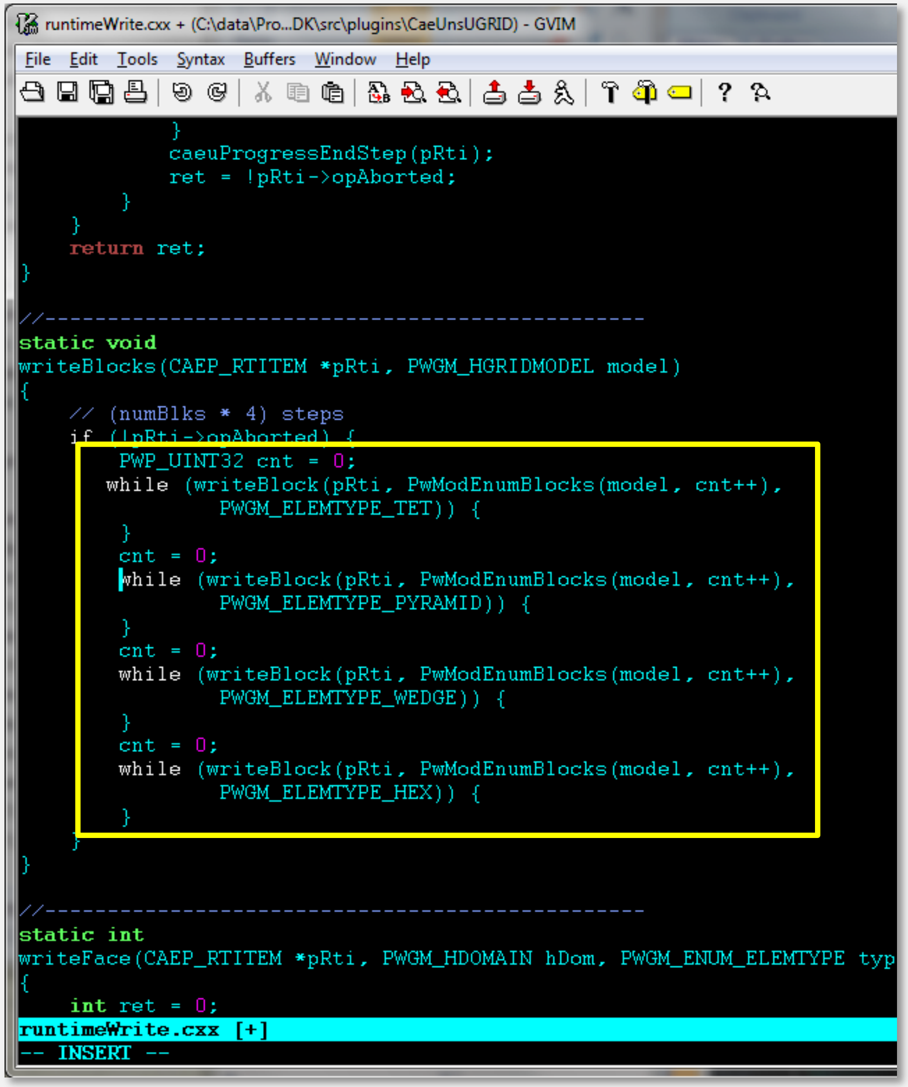
```
static void
writeMapbcFile(CAEP_RTITEM *pRti)
{
/*
      NumBCs
 surfid typeid      username
         1         2
12345678901234567890123456789
           6
      1   4000      viscous
      2   5000      Box
      3   5000      Box
      4   5000      Box
      5   5000      Box
      6   5000      Box
*/
    if (!pRti->opAborted) {
        char filename[FILENAME_MAX];
        strcpy(filename, pRti->pWriteInfo->fileDest);
        strcat(filename, ".mapbc");
        FILE *f = pwpFileOpen(filename, pwpWrite | pwpAscii);
        if (f) {
            PWGM_CONDDATA CondData;
            char tmpStr[256];
            PWP_UINT32 domCnt = PwModDomainCount(pRti->model);
            sprintf(tmpStr, "%12u\n", (unsigned int)domCnt);
            pwpFileWriteStr(tmpStr, f);
            for (PWP_UINT32 ii = 0; ii < domCnt; ++ii) {
                if (PwDomCondition(PwModEnumDomains(pRti->model, ii),
                        &CondData)) {
                    sprintf(tmpStr, "%7u%7u      %s\n",
                        (unsigned int)CondData.id, (unsigned int)CondData.tid,
                        CondData.name);
                    pwpFileWriteStr(tmpStr, f);
                }
            }
            pwpFileClose(f);
        }
runtimeWrite.cxx [+]
-- INSERT --
```

POINTWISE2013 USER GROUP MEETING

runTimeWrite.cxx

Element Separation

- UGRID block elements must be written in a specific, cell-type order.
  - TET
  - PYRAMID
  - WEDGE
  - HEX
- For each type, loop over all blocks in the model.


runtimeWrite.cxx + (C:\data\Pro...DK\src\plugins\CaeUnsUGRID) - GVIM
File Edit Tools Syntax Buffers Window Help

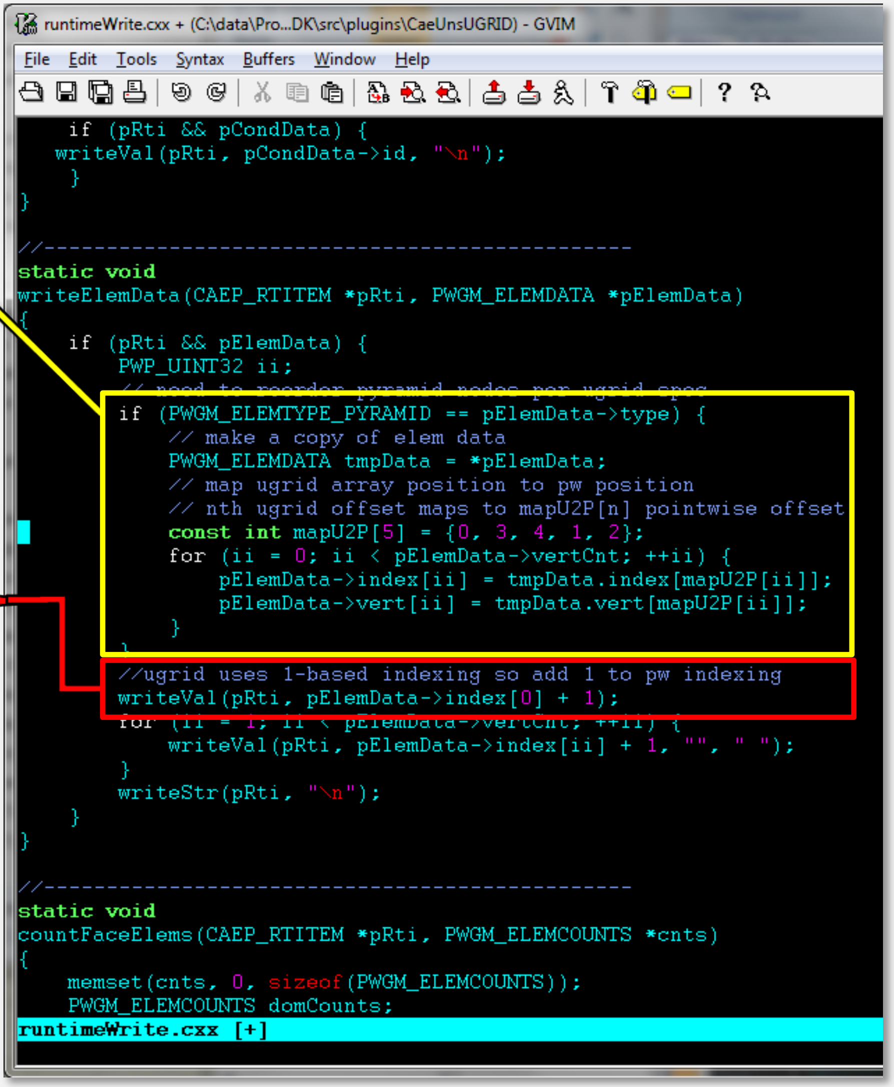
```
            }
            caeuProgressEndStep(pRti);
            ret = !pRti->opAborted;
        }
    }
    return ret;
}

//------------------------------------------------
static void
writeBlocks(CAEP_RTITEM *pRti, PWGM_HGRIDMODEL model)
{
    // (numBlks * 4) steps
    if (!pRti->opAborted) {
        PWP_UINT32 cnt = 0;
       while (writeBlock(pRti, PwModEnumBlocks(model, cnt++),
                PWGM_ELEMTYPE_TET)) {
        }
        cnt = 0;
        while (writeBlock(pRti, PwModEnumBlocks(model, cnt++),
                PWGM_ELEMTYPE_PYRAMID)) {
        }
        cnt = 0;
        while (writeBlock(pRti, PwModEnumBlocks(model, cnt++),
                PWGM_ELEMTYPE_WEDGE)) {
        }
        cnt = 0;
        while (writeBlock(pRti, PwModEnumBlocks(model, cnt++),
                PWGM_ELEMTYPE_HEX)) {
        }
    }
}

//------------------------------------------------
static int
writeFace(CAEP_RTITEM *pRti, PWGM_HDOMAIN hDom, PWGM_ENUM_ELEMTYPE typ
{
    int ret = 0;
runtimeWrite.cxx [+]
-- INSERT --
```

POINTWISE2013 USER GROUP MEETING

runTimeWrite.cxx

Pyramid Connectivity

- UGRID elements have the same connectivity as Pointwise elements *except* for pyramids.
  - Must reorder points for pyramids.
- UGRID uses 1-based index. Pointwise is 0-based.
- Use *writeVal()* to handle the data encoding.


runtimeWrite.cxx + (C:\data\Pro...DK\src\plugins\CaeUnsUGRID) - GVIM
File Edit Tools Syntax Buffers Window Help

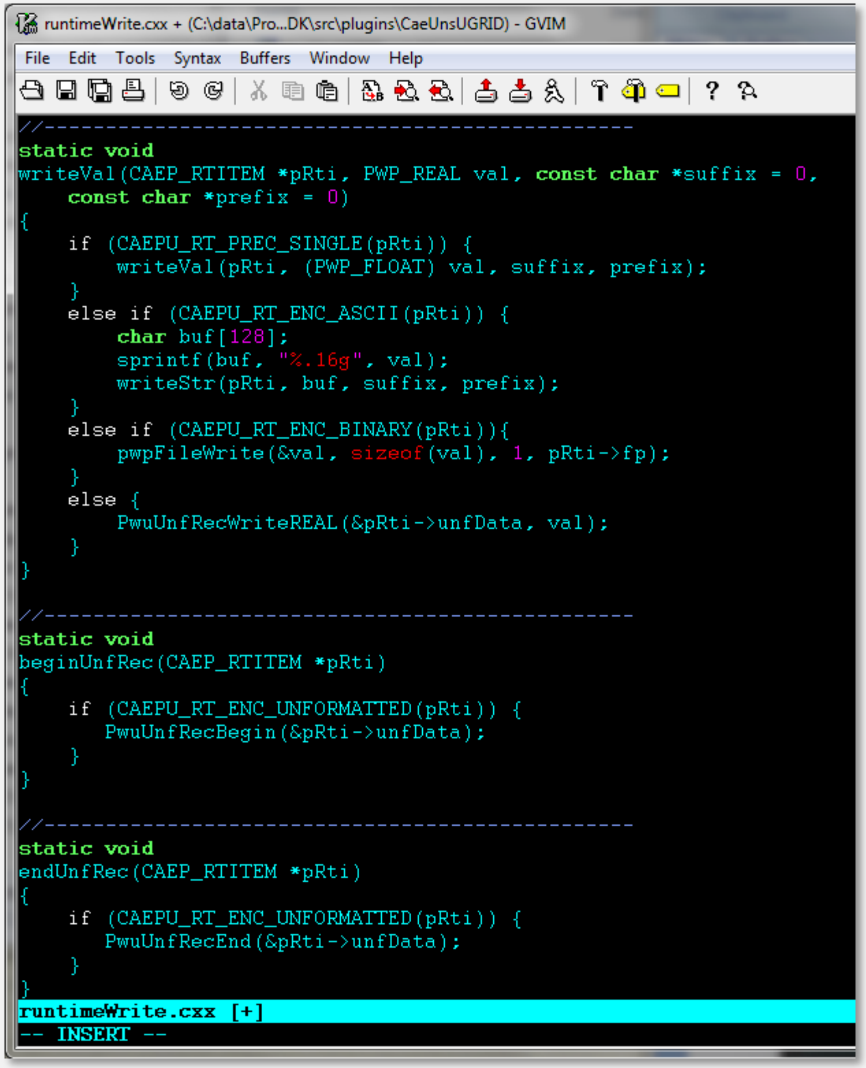
```
    if (pRti && pCondData) {
   writeVal(pRti, pCondData->id, "\n");
    }
}

//-----------------------------------------------
static void
writeElemData(CAEP_RTITEM *pRti, PWGM_ELEMDATA *pElemData)
{
    if (pRti && pElemData) {
        PWP_UINT32 ii;
        // need to reorder pyramid nodes per ugrid spec
        if (PWGM_ELEMTYPE_PYRAMID == pElemData->type) {
            // make a copy of elem data
            PWGM_ELEMDATA tmpData = *pElemData;
            // map ugrid array position to pw position
            // nth ugrid offset maps to mapU2P[n] pointwise offset
            const int mapU2P[5] = {0, 3, 4, 1, 2};
            for (ii = 0; ii < pElemData->vertCnt; ++ii) {
                pElemData->index[ii] = tmpData.index[mapU2P[ii]];
                pElemData->vert[ii] = tmpData.vert[mapU2P[ii]];
            }
        }
        //ugrid uses 1-based indexing so add 1 to pw indexing
        writeVal(pRti, pElemData->index[0] + 1);
        for (ii = 1; ii < pElemData->vertCnt; ++ii) {
            writeVal(pRti, pElemData->index[ii] + 1, "", " ");
        }
        writeStr(pRti, "\n");
    }
}

//-----------------------------------------------
static void
countFaceElems(CAEP_RTITEM *pRti, PWGM_ELEMCOUNTS *cnts)
{
    memset(cnts, 0, sizeof(PWGM_ELEMCOUNTS));
    PWGM_ELEMCOUNTS domCounts;
runtimeWrite.cxx [+]
```

POINTWISE2013 USER GROUP MEETING

runTimeWrite.cxx

Auto Encoding

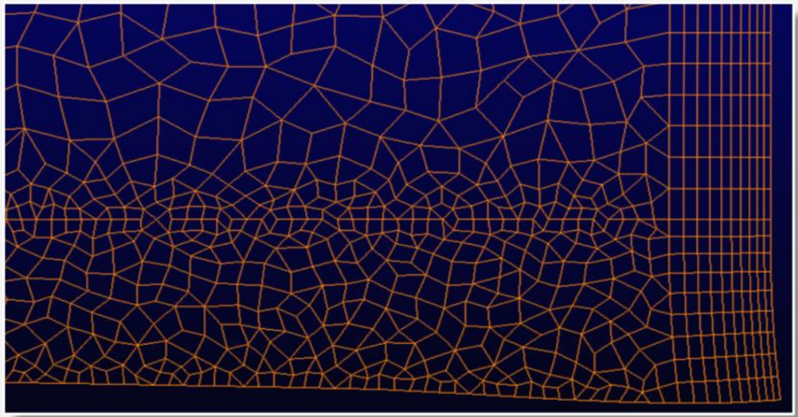
- UGRID supports 3 encodings in 2 precisions.
  - 6 combinations!
- *writeVal()* automatically writes data using the proper encoding and precision.
- Relies on C++ function overloading.
- Works best when data items written in the same order for all encoding and precision combinations.


runtimeWrite.cxx + (C:\data\Pro...DK\src\plugins\CaeUnsUGRID) - GVIM
File Edit Tools Syntax Buffers Window Help

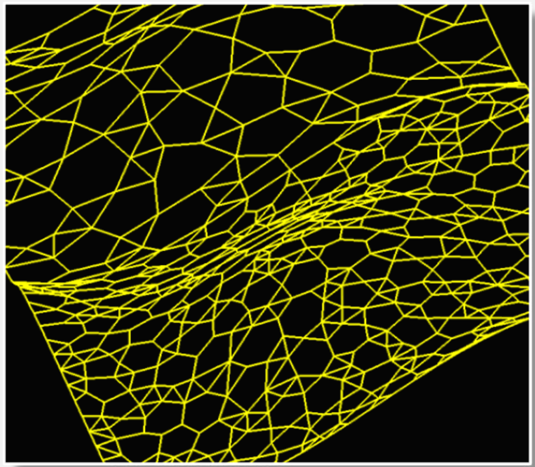
```
//------------------------------------------------
static void
writeVal(CAEP_RTITEM *pRti, PWP_REAL val, const char *suffix = 0,
    const char *prefix = 0)
{
    if (CAEPU_RT_PREC_SINGLE(pRti)) {
        writeVal(pRti, (PWP_FLOAT) val, suffix, prefix);
    }
    else if (CAEPU_RT_ENC_ASCII(pRti)) {
        char buf[128];
        sprintf(buf, "%.16g", val);
        writeStr(pRti, buf, suffix, prefix);
    }
    else if (CAEPU_RT_ENC_BINARY(pRti)){
        pwpFileWrite(&val, sizeof(val), 1, pRti->fp);
    }
    else {
        PwuUnfRecWriteREAL(&pRti->unfData, val);
    }
}

//------------------------------------------------
static void
beginUnfRec(CAEP_RTITEM *pRti)
{
    if (CAEPU_RT_ENC_UNFORMATTED(pRti)) {
       PwuUnfRecBegin(&pRti->unfData);
    }
}

//------------------------------------------------
static void
endUnfRec(CAEP_RTITEM *pRti)
{
    if (CAEPU_RT_ENC_UNFORMATTED(pRti)) {
       PwuUnfRecEnd(&pRti->unfData);
    }
}
runtimeWrite.cxx [+]
-- INSERT --
```

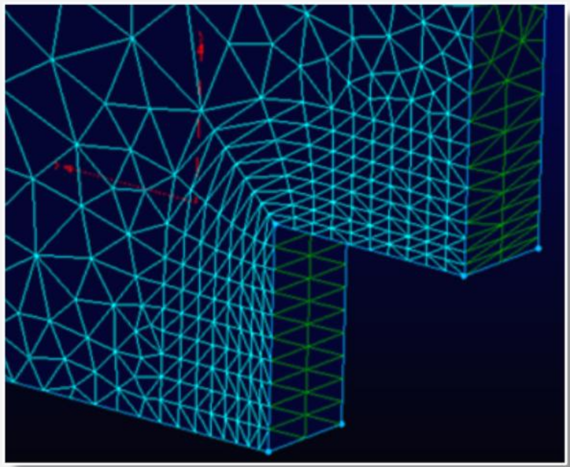
POINTWISE2013 USER GROUP MEETING

Think Outside The Box


CaeUnsRecomb2D plugin

CaePrint3D plugin


CaeUnsRecombPoly2D plugin


CaeUnsTRex2D plugin

POINTWISE2013 USER GROUP MEETING

Additional Resources

Another Fine Mesh
http://blog.pointwise.com

A wonderful resource for all things "grid".

CAE Plugin SDK Related Posts:

- **Dr. Strangegrid or: How I Learned to Stop Waiting and Write a CAE Plugin**
  http://blog.pointwise.com/2012/02/22/dr-strangegrid-or-how-i-learned-to-stop-waiting-and-write-a-cae-plugin/
- **Creating a CAE Plugin – Understanding the Pointwise Grid Model API (Part 1)**
  http://blog.pointwise.com/2012/05/23/creating-a-cae-plugin-understanding-the-pointwise-grid-model-api-part-1/
- **Creating a CAE Plugin – Understanding The Pointwise Grid Model API (Part 2)**
  http://blog.pointwise.com/2012/06/14/creating-a-cae-plugin-understanding-the-pointwise-grid-model-api-part-2/
- **Printing Grids in 3D**
  http://blog.pointwise.com/2012/03/12/printing-grids-in-3d/

Appendix A – The Pointwise Grid Model

Appendix B – Building Your Plugin

The Pointwise Grid Model

Appendix A

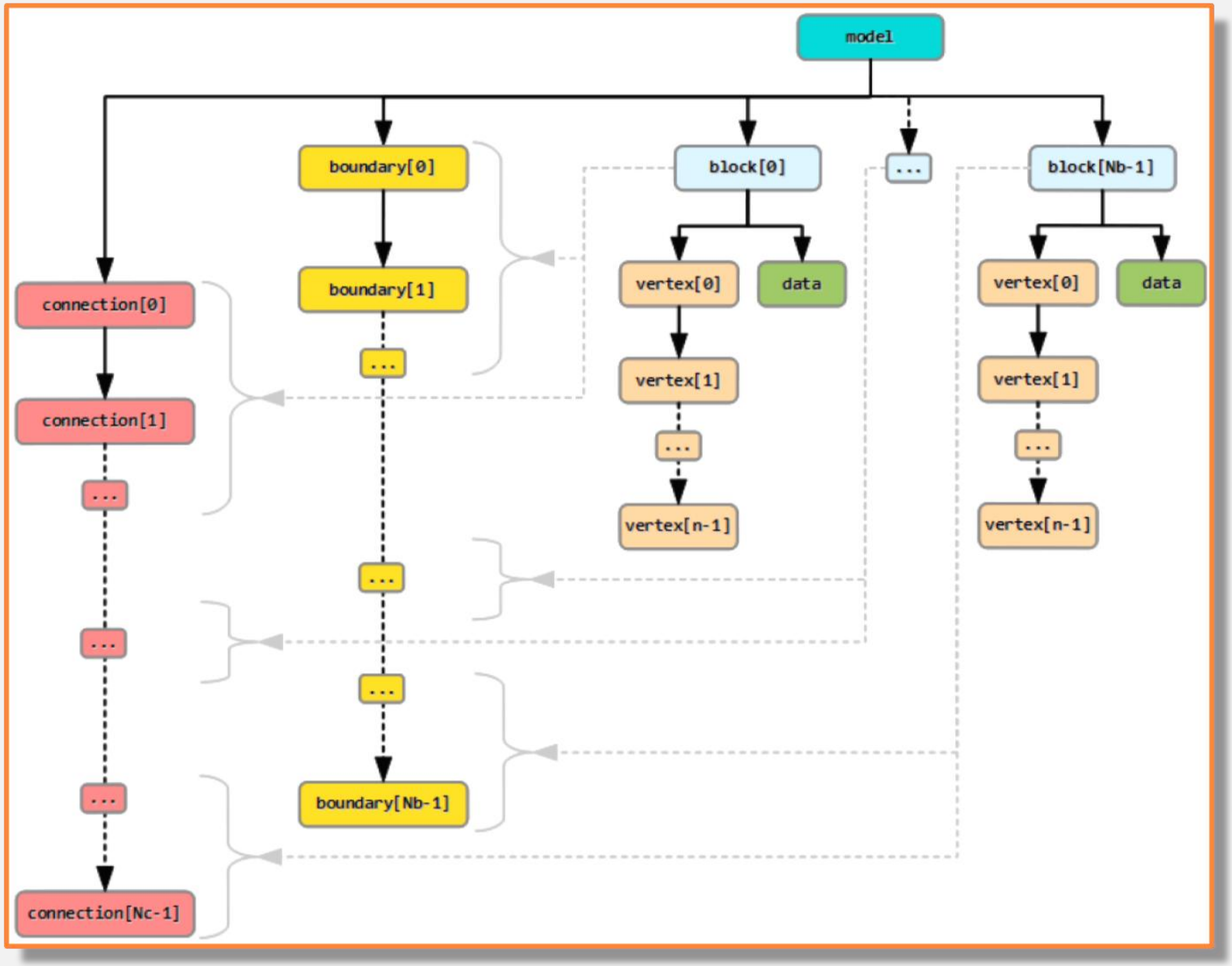
The Pointwise Grid Model

PWGM / Structured

- Blocks have a rectilinear (i, j, k) structure.
- Internal block connectivity is explicit.
- Block to block connectivity must be defined.
- Each block has its own set of vertices.
- Each block has six faces (boundaries).
  - Faces at i-min, j-min, k-min, i-max, j-max, z-max
  - A face may be subdivided into rectangular subsets.
    - Subset may be point-match connected with a neighboring block.
  - Transformations used to map ijk values across block connections.
- Each block has condition data.

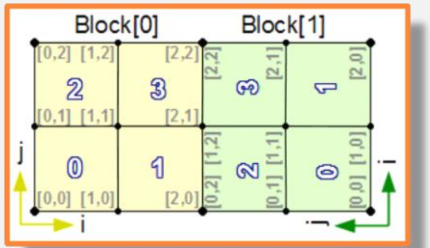
The Pointwise Grid Model

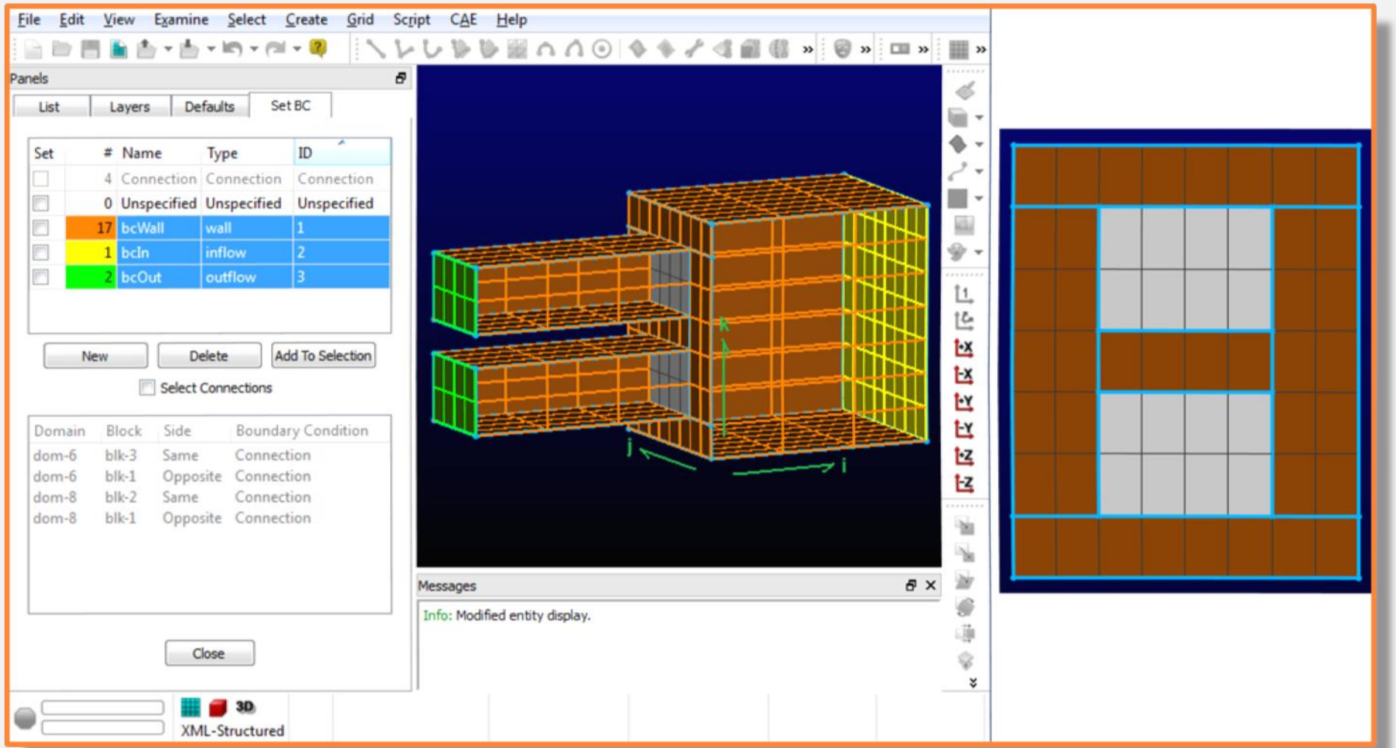
PWGM / Structured



The Pointwise Grid Model

PWGM / Structured





Set	#	Name	Type	ID
☐	4	Connection	Connection	Connection
☐	0	Unspecified	Unspecified	Unspecified
☐	17	bcWall	wall	1
☐	1	bcIn	inflow	2
☐	2	bcOut	outflow	3

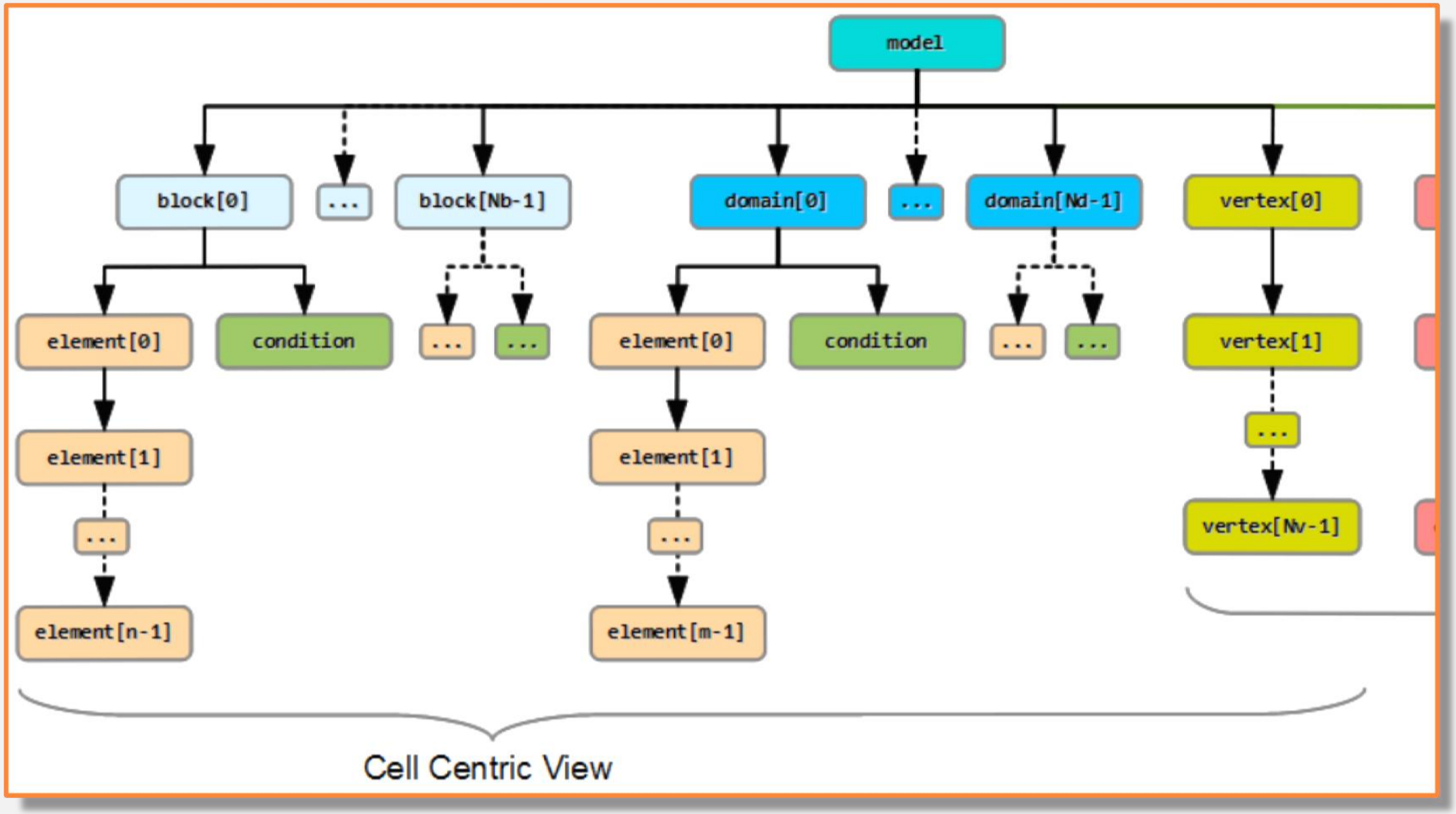
Domain	Block	Side	Boundary Condition
dom-6	blk-3	Same	Connection
dom-6	blk-1	Opposite	Connection
dom-8	blk-2	Same	Connection
dom-8	blk-1	Opposite	Connection

Info: Modified entity display.

XML-Structured

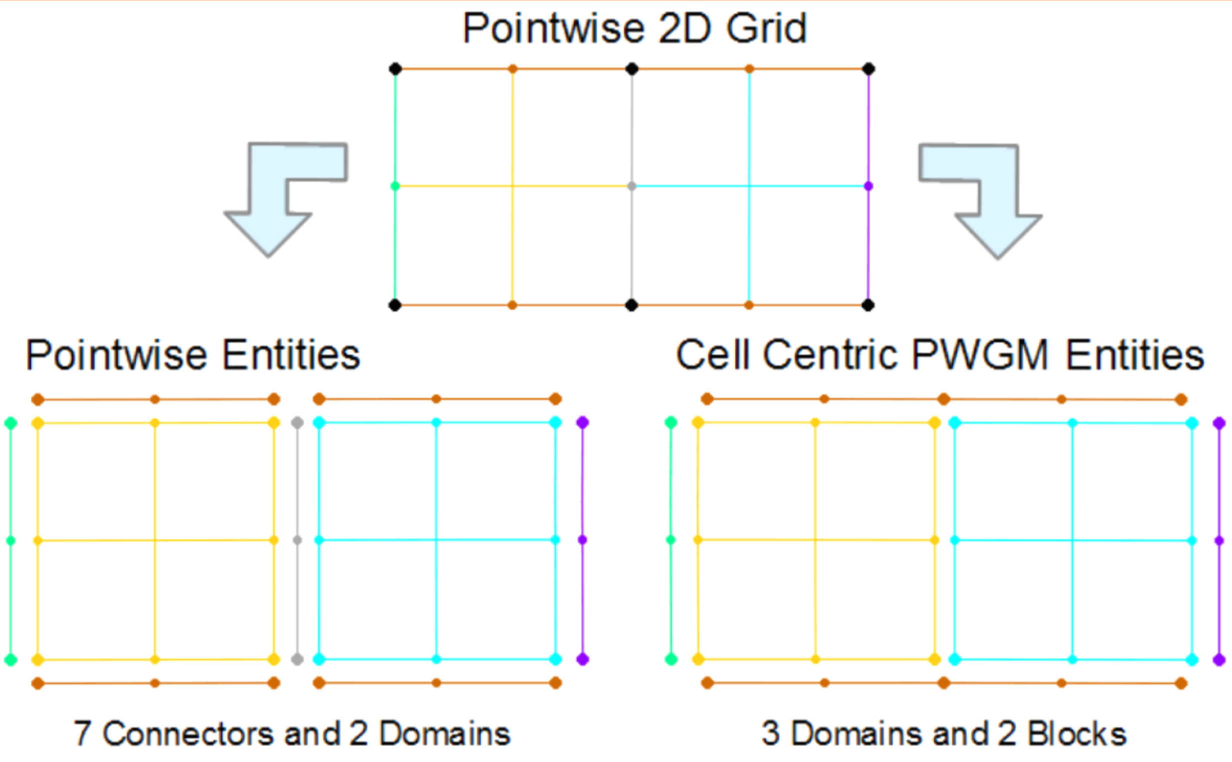
The Pointwise Grid Model

PWGM / Unstructured / Cell Centric



The Pointwise Grid Model

PWGM / Unstructured / Cell Centric



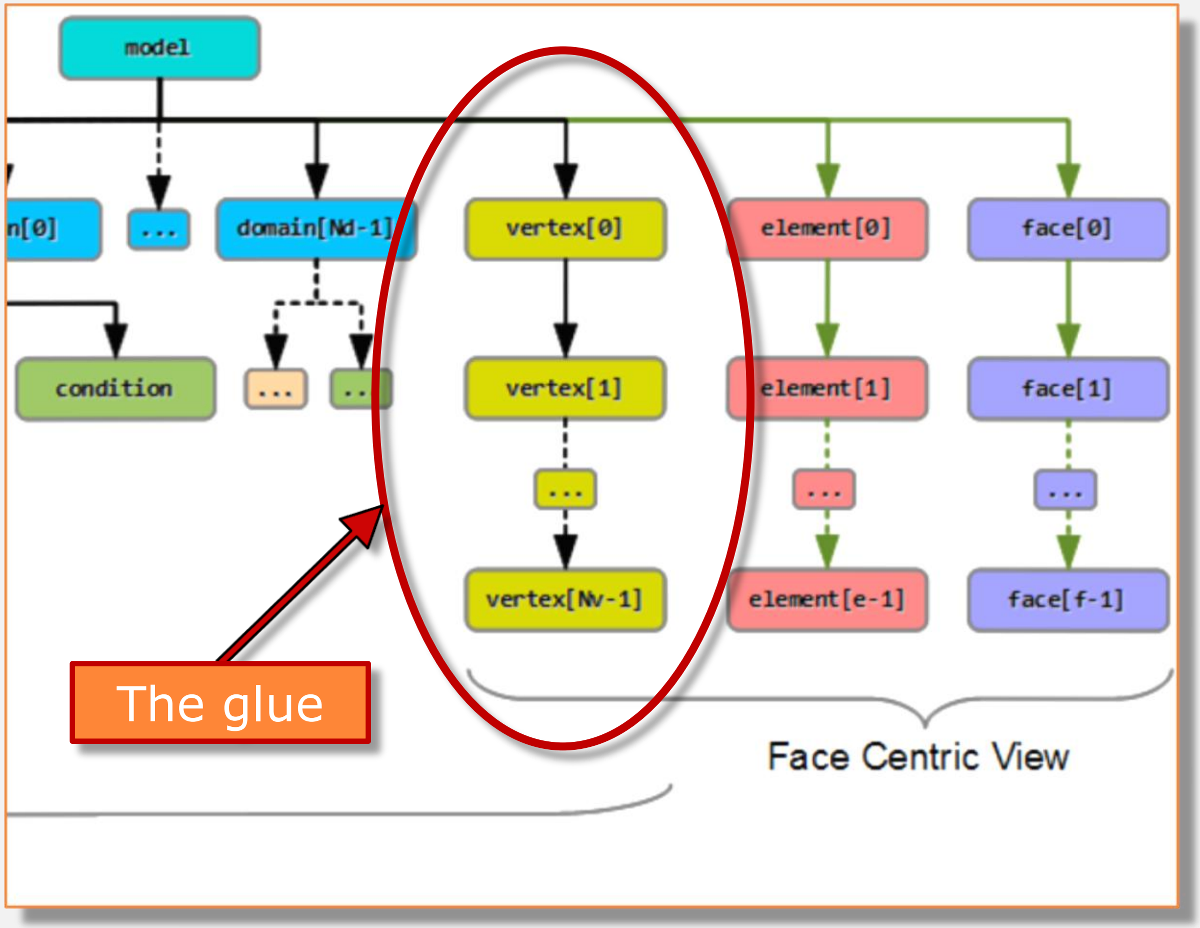
Entities of the same color share the same condition.

Even though the upper and lower brown entities are not physically connected, they are merged into a single boundary in the PWGM.

The gray entity between the blocks is a connection and is not part of any PWGM boundary.

The Pointwise Grid Model

PWGM / Unstructured / Face Centric

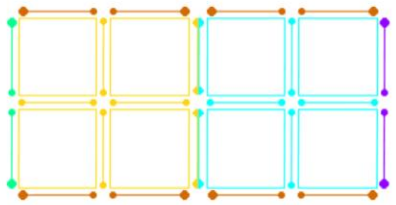


The Pointwise Grid Model

PWGM / Unstructured / Face Centric

- Pointwise's native grid data does *not* include face information.
- Pointwise must generate face information when it is requested by a plugin.
- Generating face data can be time consuming and memory intensive for large grids.
- Do not request face data unless your plugin really needs it.
- To minimize memory usage, face information is streamed to the plugin as it is generated and then deleted.

Face Centric PWGM Entities



8 Cell Elements and 22 Face Elements

Face elements that only touch one cell element are boundary face elements.

Boundary face elements of the same color share the same boundary condition.

Cell elements of the same color share the same volume condition.

The single-colored face elements between two cell elements of the same color are internal block faces.

The bi-colored face elements between two cell elements with different colors are internal, block connection faces.

The Pointwise Grid Model

PWGM / Unstructured / Face Streaming

- A plugin obtains grid model face data by calling **`PwModStreamFaces()`** with three plugin defined callback functions.
  - **`beginCB(`**`PWGM_BEGINSTREAM_DATA *data`**`)`**
    Called once ***before*** first face is streamed.
  - **`faceCB(`**`PWGM_FACESTREAM_DATA *data`**`)`**
    Called once for ***each*** face in the grid model.
  - **`endCB(`**`PWGM_ENDSTREAM_DATA *data`**`)`**
    Called once after the last face is streamed.
- **`PwModStreamFaces()`** does not return until streaming is completed or aborted.
- Inside **`PwModStreamFaces()`**, Pointwise processes its native grid data and invokes the face callbacks as needed.

The Pointwise Grid Model

PWGM / Unstructured / Face Streaming

```
static PWP_UINT32 beginCB(PWGM_BEGINSTREAM_DATA *data)
{
    CAEP_RTITEM *pRti = (CAEP_RTITEM*)data->userData; // recover pointer
    // prepare to receive the face stream
    return PWP_TRUE;
}

static PWP_UINT32 faceCB(PWGM_FACESTREAM_DATA *data)
{
    CAEP_RTITEM *pRti = (CAEP_RTITEM*)data->userData; // recover pointer
    // handle the face information
    return PWP_TRUE;
}

static PWP_UINT32 endCB(PWGM_ENDSTREAM_DATA *data)
{
    CAEP_RTITEM *pRti = (CAEP_RTITEM*)data->userData; // recover pointer
    // cleanup after streaming
    return PWP_TRUE;
}

PWP_BOOL runtimeWrite(CAEP_RTITEM *pRti, PWGM_HGRIDMODEL model, const CAEP_WRITEINFO *pWriteInfo)
{
    // Must set element order BEFORE streaming faces! Since TET and PYRAMID are
    // not explicitly appended, they will enumerate last in an unspecified order.
    PwModAppendEnumElementOrder(model, PWGM_ELEMORDER_HEX);
    PwModAppendEnumElementOrder(model, PWGM_ELEMORDER_WEDGE);

    // Faces will be streamed to faceCB in an unspecified order
    const PWGM_ENUM_FACEORDER faceOrder = PWGM_FACEORDER_DONTCARE;

    // Passing pRti as the userData pointer
    PwModStreamFaces(model, faceOrder, beginCB, faceCB, endCB, pRti);

    PWGM_ELEMDATA ElemData;
    PWGM_ENUMELEMDATA EnumElemData;
    PWP_UINT32 ndx = 0;
    PWGM_HELEMENT hElem = PwModEnumElements(model, ndx);
    while (PWGM_HELEMENT_ISVALID(hElem)) {
        // get basic element connectivity information
        if (PwElemDataMod(hElem, &ElemData)) {
            // ...etc...
        }
        // get detailed element ownership and connectivity information
        if (PwElemDataModEnum(hElem, &EnumElemData)) {
            // ...etc...
        }
        hElem = PwModEnumElements(model, ++ndx);
    }
}
```

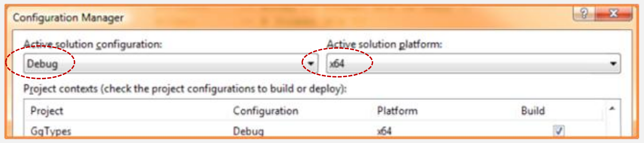
Building Your Plugin

Appendix B

Building Your Plugin

Microsoft Visual Studio 2008*

- Open ***.../PluginSDK/PluginSDK.sln***.
- Add ***CaeUnsUGRID.vcproj*** if needed.
  - Menu *File/Add/Existing Project…*
  - In **.../PluginSDK/src/plugins/CaeUnsUGRID/**
- Set the build configuration and platform.
  - Menu ***Build/Configuration Manager…***
    - *Debug or Release*
    - *x64 (64 bit) or Win32 (32-bit)*
  - Can also use drop lists in build toolbar.
- Build the solution
  - Menu ***Build/Build Solution***
  - Builds all plugins including the sample XML plugins.



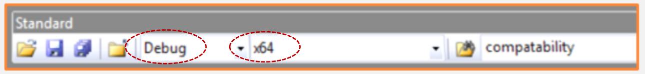
Configuration Manager

Active solution configuration: Debug

Active solution platform: x64

Project contexts (check the project configurations to build or deploy):

Project	Configuration	Platform	Build
GgTypes	Debug	x64	✓



Standard

Debug | x64 | compatability

* Pointwise is compiled using Visual Studio 2008. For compatibility, plugins should also be compiled with Visual Studio 2008. Using other compilers is untested and unsupported. See http://www.pointwise.com/pw/port.shtml for full details.

Building Your Plugin

gmake (linux / macOSX)*

- Start a command shell.
- Change the working directory to ***.../PluginSDK***.
- Run ***gmake BUILD=help*** to see build usage information.
- Run ***gmake machine=M BUILD=B*** [***target***] where
  - ***M*** is one of the Pointwise supported platforms

• ***linux***	32-bit linux-based OS
• ***linux_x86_64***	64-bit linux-based OS
• ***macosx***	mac OSX universal binary

  - ***B*** is one of the build types
    - ***Debug*** (the default)
    - ***Release***
  - You can also set the ***machine*** and ***BUILD*** environment vars.
- Useful make targets (***machine*** set in env)

```
$ gmake plugins              // build all plugins (the default)
$ gmake PluginName           // build only PluginName
$ gmake PluginName_clean     // clean only PluginName
$ gmake clean                // clean all plugins
$ gmake clobber              // rm -rf {dist,src}/$(machine)
$ gmake nuke                 // rm -rf {dist,src}/$(machine) dist
$ gmake Target-dr            // build Target for both Debug and Release
$ gmake install_validate     // validates PW_RUNTIME_ROOT setting
$ gmake print.macro          // prints value of gmake macro (for make file debugging)
```

* Pointwise is compiled using Red Hat Enterprise Linux 5 or MacOSX V10.5/6/7. For compatibility, plugins should also be compiled with the same versions. Using other versions is untested and unsupported. See http://www.pointwise.com/pw/port.shtml for full details.

Building Your Plugin

General Information

- On all platforms, the build files are generated in the following locations.
  - ***.../PluginSDK/src/<machine>/<BUILD>/plugins/PluginName/***
    - Object files (.o, .obj).
    - Deleted by the ***clean*** make target.
  - ***.../PluginSDK/dist/<machine>/plugins/***
    - The release build plugins (.dll, .so, dylib).
    - Deleted by the ***clobber*** and ***nuke*** make target.
  - ***.../PluginSDK/dist/<machine>/plugins/debug/***
    - The *Debug* build plugins (.dll, .so, dylib).
    - Deleted by the ***clobber*** and ***nuke*** make target.
- To facilitate the development of plugins, the SDK build system supports two environment variables.
  - ***PW_RUNTIME_ROOT***
    - Build system copies newly compiled plugins to this location.
    - Must have write permissions in your Pointwise installation.
  - ***PWI_PLUGINS_SEARCH_PATH***
    - Defines the location(s) from which Pointwise will load plugins at runtime.
    - Do not need write permissions in your Pointwise installation.
    - Possible (but unlikely) security hole.

Building Your Plugin

PW_RUNTIME_ROOT

If defined, the build system automatically copies all newly built plugins to the Pointwise installation for testing.

The plugin is only copied if the build is successful.

Your user account must have write permissions in the plugins folder.

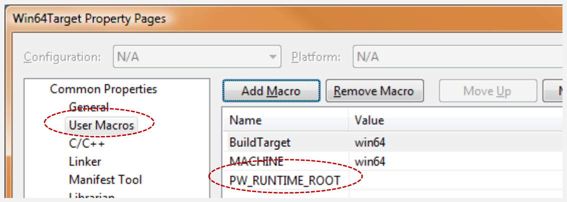
Set value to the folder that contains your Pointwise installation's ***plugins*** folder.

Typical paths would be similar to

- `c:/Program Files (x86)/Pointwise/win64/`
- `c:/Program Files/Pointwise/win32/`
- `/home/<username>/Pointwise/Pointwise<version>/linux/`
- `/home/<username>/Pointwise/Pointwise<version>/linux_x86_64/`
- `/home/<username>/Pointwise/Pointwise<version>/macosx/Pointwise.app/Contents/`

On Windows

- Set value in ***Property Pages/User Macros***.
- Be sure to *Save* property page after editing.



Name	Value
BuildTarget	win64
MACHINE	win64
PW_RUNTIME_ROOT	

On linux and macOSX

- Set value in shell.
- Check the setting with ***`make install_validate`***.

Building Your Plugin

PWI_PLUGINS_SEARCH_PATH

- If defined, Pointwise will load all plugins from the path(s) specified in PWI_PLUGINS_SEARCH_PATH.
- Paths and folder ids are delimited with a comma or semicolon.
- At startup, Pointwise searches these folders in the specified order and loads any plugins found.
- If there are any loading conflicts, the first plugin found will be loaded. Any subsequent, conflicting plugins will be ignored.

Building Your Plugin

PWI_PLUGINS_SEARCH_PATH

```
PWI_PLUGINS_SEARCH_PATH=[{@FolderId|Folder}{,|;}]...
```

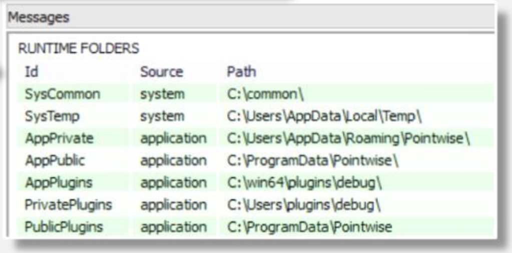
where,

FolderId

One of the standard folder ids preceded by the **@** char. Typically, the **PrivatePlugins** or **PublicPlugins** folder id is used. However, any valid folder id will be accepted.

Use **@** by itself or **@AppPlugins** to include the plugins shipped with PW. **@** is shorthand for **@AppPlugins**. Invalid folder ids are silently ignored.

To see the full list of valid folder ids run the ***Ctrl-0/Show Installation Info*** tool from within Pointwise.



Messages

RUNTIME FOLDERS

Id	Source	Path
SysCommon	system	C:\common\
SysTemp	system	C:\Users\AppData\Local\Temp\
AppPrivate	application	C:\Users\AppData\Roaming\Pointwise\
AppPublic	application	C:\ProgramData\Pointwise\
AppPlugins	application	C:\win64\plugins\debug\
PrivatePlugins	application	C:\Users\plugins\debug\
PublicPlugins	application	C:\ProgramData\Pointwise

Folder

An explicit folder path. Invalid paths are silently ignored.

Example plugin search path settings:

@PublicPlugins;@PrivatePlugins;@AppPlugins

Load plugins from the **PublicPlugins** folder, followed by the **PrivatePlugins** folder, and finally, the **AppPlugins** folder.

@PublicPlugins;@PrivatePlugins;@

Same as previous example except for using the shorthand @AppPlugins id.

h:/path/to/plugins*;@;*c:/another/path/to/plugins

Load plugins from the ***h:/path/to/plugins*** folder, followed by plugins from the **AppPlugins** folder, and finally, plugins from the ***c:/another/path/to/plugins*** folder.