Instrument Kernel
IK

January 2012

Purpose

- The Instrument Kernel serves as a repository for instrument specific information that may be useful within the SPICE context.
  – Always included:
    » Specifications for an instrument's field-of-view (FOV) size, shape, and orientation
  – Other possibilities:
    » Internal instrument timing parameters and other data relating to SPICE computations might also be placed in an I-kernel
    » Instrument geometric calibration data
- Note: instrument mounting alignment data are specified in a mission's Frames Kernel (FK)
  – (Wasn't true for some of the earliest missions that used SPICE)

I-Kernel Structure

- An I-Kernel is a SPICE text kernel. The format and structure of a typical I-Kernel is shown below.

```
KPL/IK
   Comments describing the keywords and values
   to follow, as well as any other pertinent
   information.

      \begindata
         Keyword = Value(s) Assignment
         Keyword = Value(s) Assignment

      \begintext

   More descriptive comments.

      \begindata
         Keyword = Value(s) Assignment
      \begintext

   More descriptive comments.
```

etc …

I-Kernel Contents (1)

- Examples of IK keywords, with descriptions:
  - INS-94031_FOCAL_LENGTH — MGS MOC NA focal length
  - INS-41220_IFOV — MEX HRSC SRC pixel angular size
  - INS-41130_NUMBER_OF_SECTORS — MEX ASPERA NPI number of sectors
- In general SPICE does not require any specific keywords to be present in an IK
  - One exception is a set of keywords defining an instrument's FOV, if the SPICE Toolkit's GETFOV routine is planned to be used to retrieve the FOV attributes
    - » Keywords required by GETFOV will be covered later in this tutorial
- The requirements on keywords in an IK are the following:
  - Keywords must begin with INS[#], where [#] is replaced with the NAIF instrument ID code (which is a negative number)
  - The total length of the keyword must be less than or equal to 32 characters
  - Keywords are case-sensitive (Keyword != KEYWORD)

I-Kernel Contents (2)

- **IKs should contain extensive comments regarding:**
  - Instrument overview
  - Reference source(s) for the data included in the IK
  - Names/IDs assigned to the instrument and its parts
  - Explanation of each keyword included in the file
  - Description of the FOV and detector layout
  - Where appropriate, descriptions of the algorithms in which parameters provided in the IK are used, and even fragments of source code implementing these algorithms
    - » For example optical distortion models or timing algorithms

- **This documentation exists primarily to assist users in integrating I-Kernel data into their applications**
  - One needs to know the keyword name to get its value(s) from the IK data
  - One needs to know what each value means in order to use it properly

I-Kernel Interface Routines

- **As with any SPICE kernel, an IK is loaded using FURNSH**

```
CALL FURNSH ( 'ik_file_name.ti' )
```

{ Better yet, use a FURNSH kernel }

- **By knowing the name and type (DP, integer, or character) of a keyword of interest, the value(s) associated with that keyword can be retrieved using G*POOL routines**

```
CALL GDPOOL ( NAME, START, ROOM, N, VALUES, FOUND )   for DP values
CALL GIPOOL ( NAME, START, ROOM, N, VALUES, FOUND )   for integer values
CALL GCPOOL ( NAME, START, ROOM, N, VALUES, FOUND )   for character string values
```

- **When an instrument's FOV is defined in the IK using a special set of keywords discussed later in this tutorial, the FOV shape, reference frame, boresight vector, and boundary vectors can be retrieved by calling the GETFOV routine**

```
CALL GETFOV ( INSTID, ROOM, SHAPE, FRAME, BSIGHT, N, BOUNDS)
```

FORTRAN examples are shown

FOV Definition Keywords (1)

- The following keywords defining FOV attributes for the instrument with NAIF ID (#) must be present in the IK if the SPICE Toolkit's GETFOV module will be used

  – Keyword defining shape of the FOV

```
INS#_FOV_SHAPE      = 'CIRCLE' or 'ELLIPSE' or
                      'RECTANGLE' or 'POLYGON'
```

  – Keyword defining reference frame in which the boresight vector and FOV boundary vectors are specified

```
INS#_FOV_FRAME      = 'frame name'
```

  – Keyword defining the boresight vector

```
INS#_BORESIGHT      = ( X, Y, Z )
```

continued on next page

FOV Definition Keywords (2)

– **Keyword(s) defining FOV boundary vectors, provided in either of two ways**

1) By specifying boundary vectors explicitly

```
INS#_FOV_CLASS_SPEC         = 'CORNERS'
INS#_FOV_BOUNDARY_CORNERS = ( X(1), Y(1), Z(1),
                               ...   ...   ...
                               X(n), Y(n), Z(n) )
```

where the `FOV_BOUNDARY_CORNERS` keyword provides an array of vectors that point to the "corners" of the instrument field of view.

Note: Use of the `INS#_FOV_CLASS_SPEC` keyword is optional when explicit boundary vectors are provided.

continued on next page

FOV Definition Keywords (3)

2) By providing half angular extents of the FOV (possible only for circular, elliptical or rectangular FOVs)

```
INS#_FOV_CLASS_SPEC          = 'ANGLES'
INS#_FOV_REF_VECTOR          = ( X, Y, Z )
INS#_FOV_REF_ANGLE           = halfangle1
INS#_FOV_CROSS_ANGLE         = halfangle2
INS#_FOV_ANGLE_UNITS         = 'DEGREES' or
                               'RADIANS' or …
```

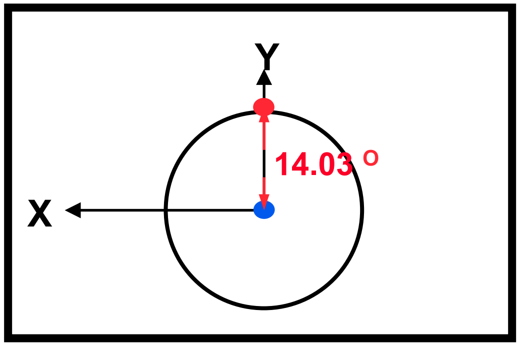
where the `FOV_REF_VECTOR` keyword specifies a reference vector that, together with the boresight vector, define the plane in which the half angle given in the `FOV_REF_ANGLE` keyword is measured. The other half angle given in the `FOV_CROSS_ANGLE` keyword is measured in the plane normal to this plane and containing the boresight vector.

FOV Definition Keywords (4)

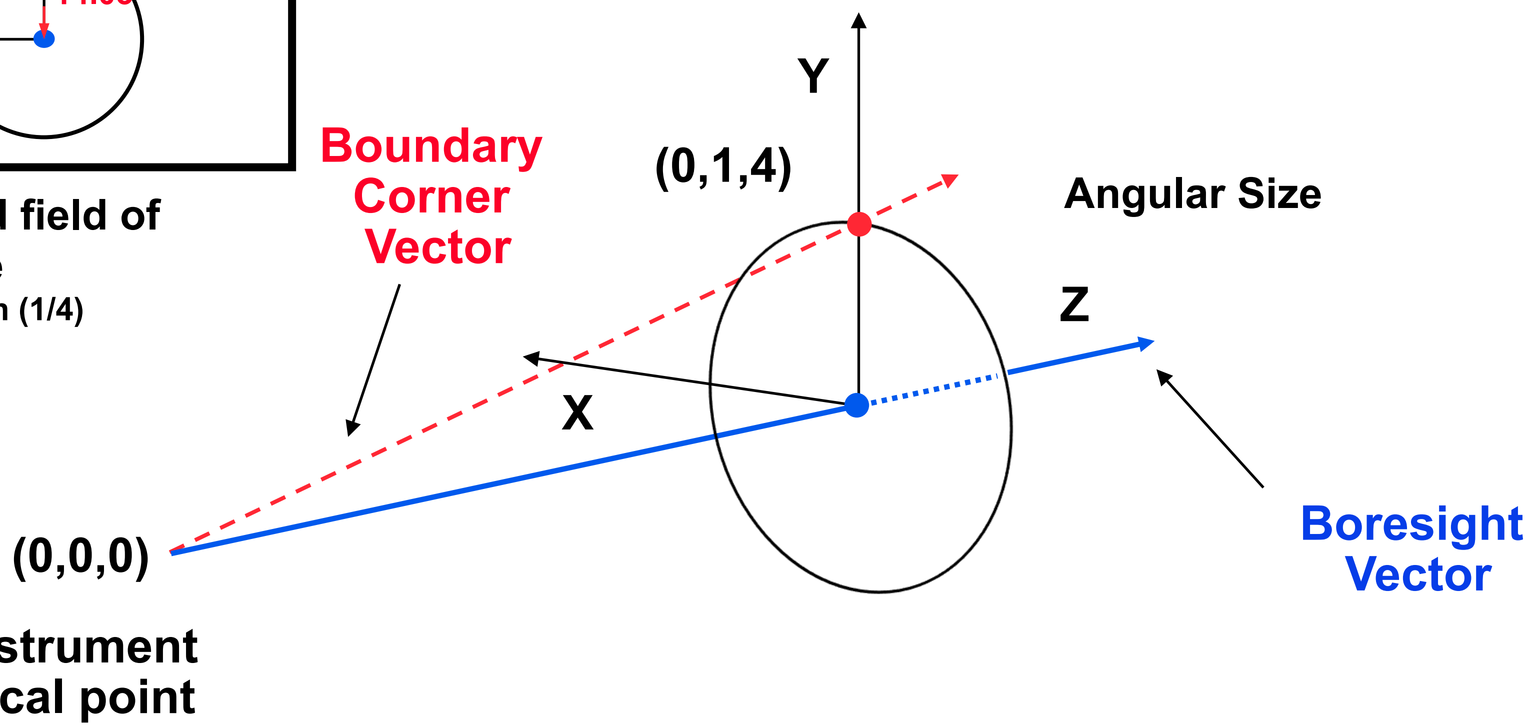
- When explicit boundary vectors are provided, they must be listed in either clockwise or counter-clockwise order, not randomly
- Neither the boresight nor reference vector has to be co-aligned with one of the FOV frame's axes
  - But for convenience, each is frequently defined to be along one of the FOV axes
- Neither the boresight nor corner nor reference vector has to be a unit vector
  - But these frequently are defined as unit vectors
- When a FOV is specified using the half angular extents method, the boresight and reference vectors have to be linearly independent but they don't have to be perpendicular
  - But for convenience the reference vector is usually picked to be normal to the boresight vector
- Half angular extents for a rectangular FOV specify the angles between the boresight and the FOV sides, i.e. they are for the middle of the FOV
- The next several pages show examples of FOV definitions

Circular Field of View

Consider an instrument with a circular field of view.



Subtended field of view angle
14.03 = arc tan (1/4)



Circular FOV Definition

The following sets of keywords and values describe this circular field of view:

Specifying boundary vectors explicitly:

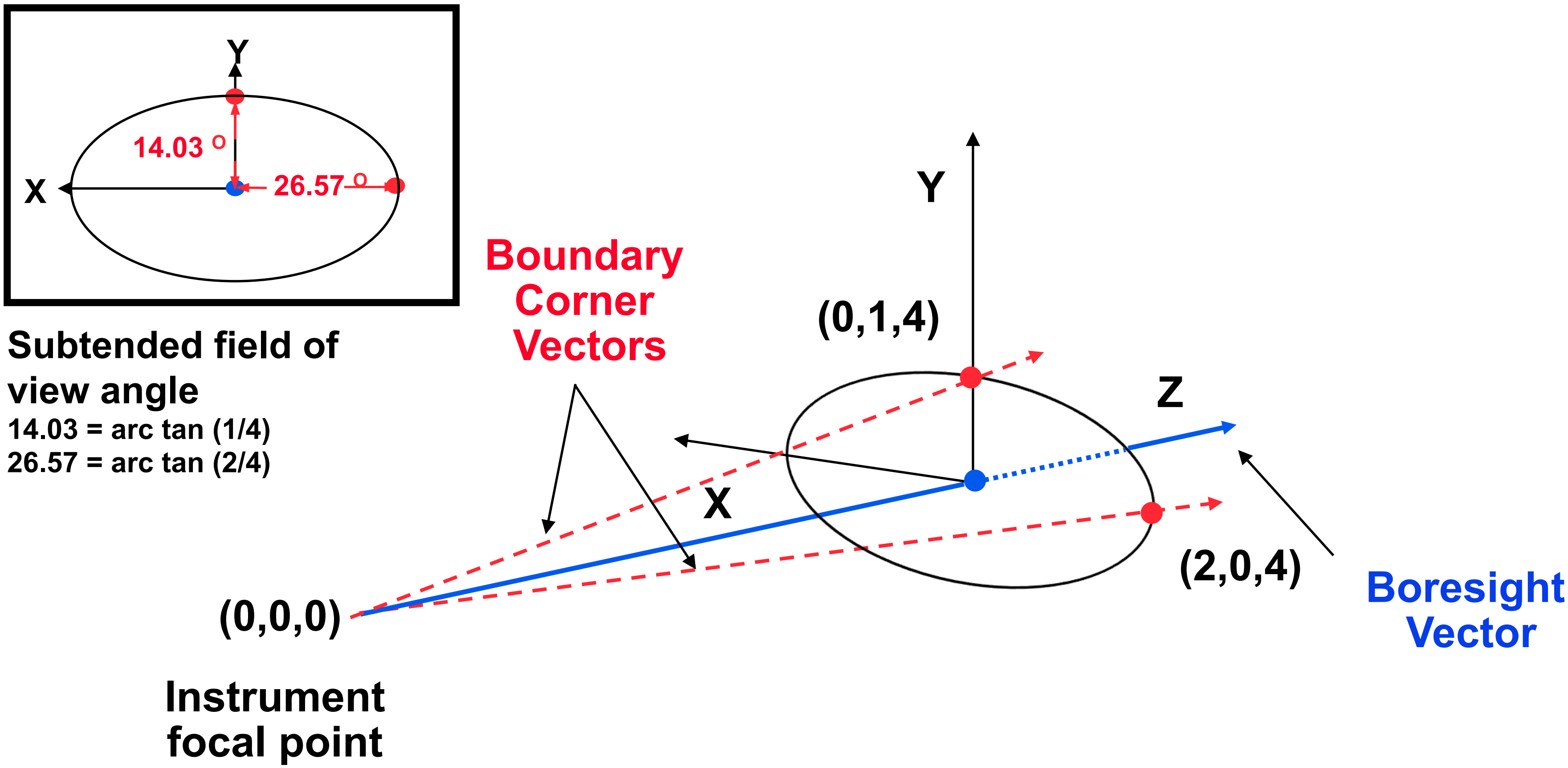
```
INS-11111_FOV_SHAPE            = 'CIRCLE'
INS-11111_FOV_FRAME            = 'FRAME_FOR_INS-11111'
INS-11111_BORESIGHT            = ( 0.0  0.0  1.0 )
INS-11111_FOV_BOUNDARY_CORNERS = ( 0.0  1.0  4.0 )
```

Specifying half angular extents of the FOV:

```
INS-11111_FOV_SHAPE            = 'CIRCLE'
INS-11111_FOV_FRAME            = 'FRAME_FOR_INS-11111'
INS-11111_BORESIGHT            = ( 0.0  0.0  1.0 )
INS-11111_FOV_CLASS_SPEC       = 'ANGLES'
INS-11111_FOV_REF_VECTOR       = ( 0.0  1.0  0.0 )
INS-11111_FOV_REF_ANGLE        = 14.03624347
INS-11111_FOV_ANGLE_UNITS      = 'DEGREES'
```

Elliptical Field of View

Consider an instrument with an elliptical field of view.



Elliptical FOV Definition

The following sets of keywords and values describe this elliptical field of view:

Specifying boundary vectors explicitly:

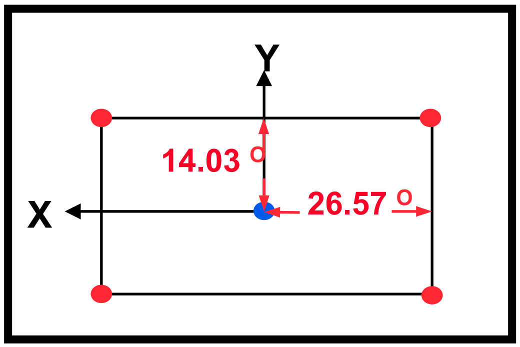
```
INS-22222_FOV_SHAPE             = 'ELLIPSE'
INS-22222_FOV_FRAME             = 'FRAME_FOR_INS-22222'
INS-22222_BORESIGHT             = ( 0.0  0.0  1.0 )
INS-22222_FOV_BOUNDARY_CORNERS = ( 0.0  1.0  4.0
                                    2.0  0.0  4.0 )
```

Specifying half angular extents of the FOV:

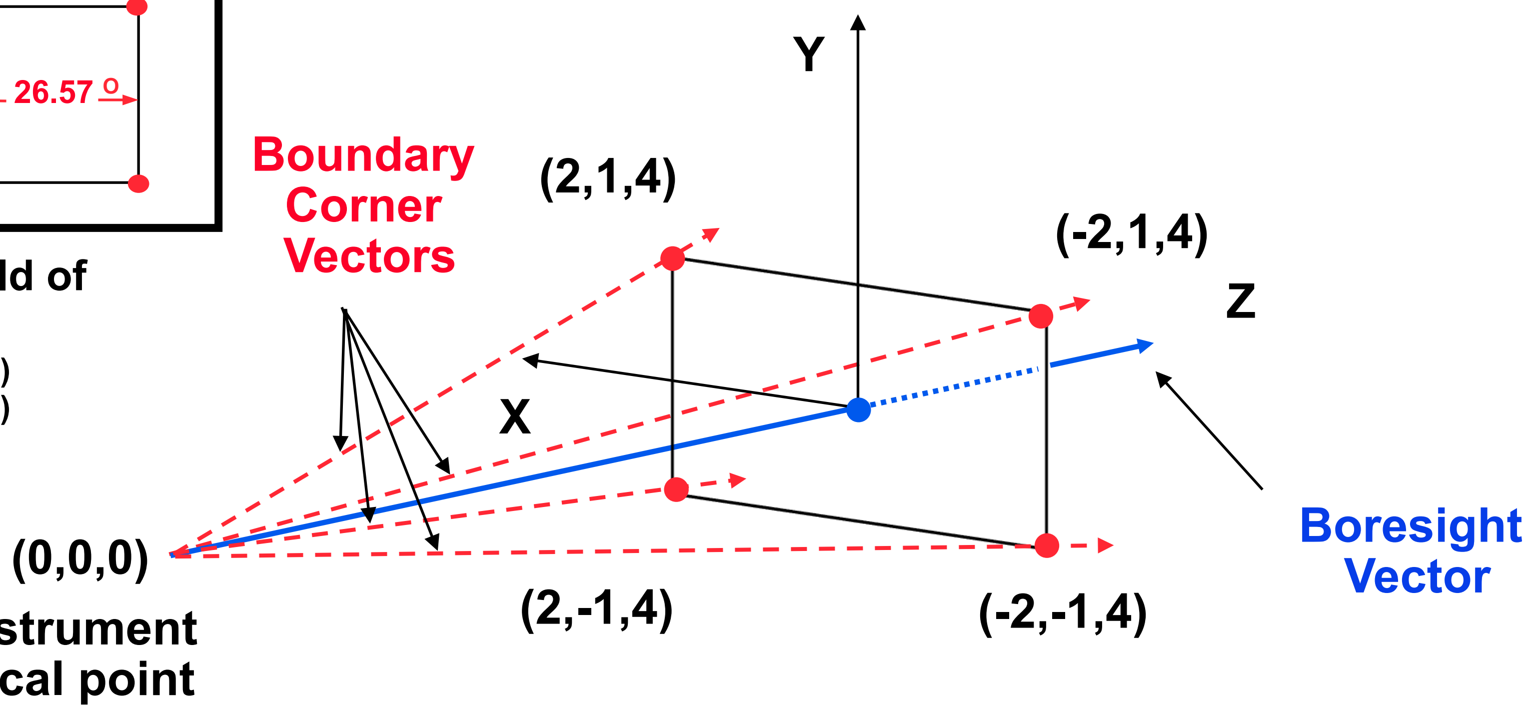
```
INS-22222_FOV_SHAPE             = 'ELLIPSE'
INS-22222_FOV_FRAME             = 'FRAME_FOR_INS-22222'
INS-22222_BORESIGHT             = ( 0.0  0.0  1.0 )
INS-22222_FOV_CLASS_SPEC        = 'ANGLES'
INS-22222_FOV_REF_VECTOR        = ( 0.0  1.0  0.0 )
INS-22222_FOV_REF_ANGLE         = 14.03624347
INS-22222_FOV_CROSS_ANGLE       = 26.56505118
INS-22222_FOV_ANGLE_UNITS       = 'DEGREES'
```

Rectangular Field of View

Consider an instrument with a rectangular field of view.



Subtended field of view angle
14.03 = arc tan (1/4)
26.57 = arc tan (2/4)



Rectangular FOV Definition

The following sets of keywords and values describe this rectangular field of view:

Specifying boundary vectors explicitly:

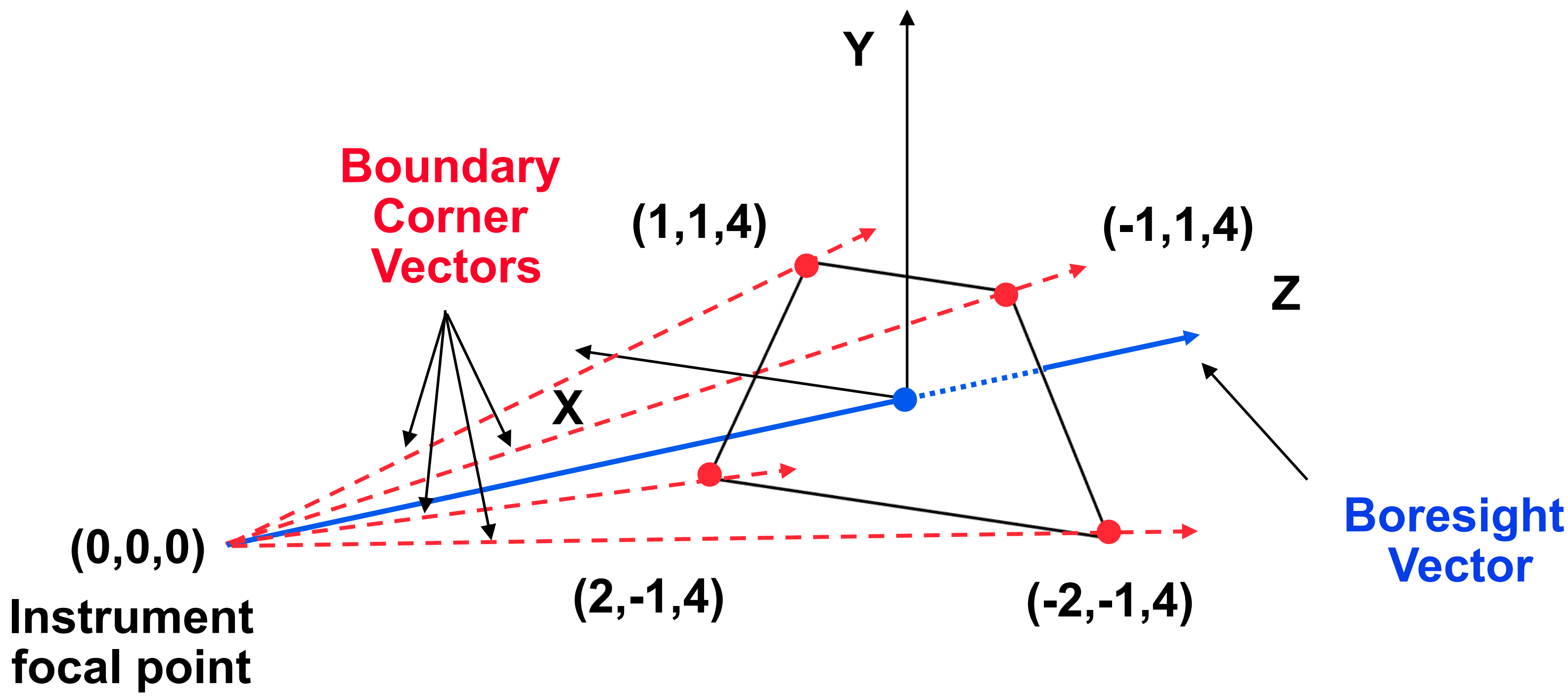
```
INS-33333_FOV_SHAPE             = 'RECTANGLE'
INS-33333_FOV_FRAME             = 'FRAME_FOR_INS-33333'
INS-33333_BORESIGHT             = ( 0.0  0.0  1.0 )
INS-33333_FOV_BOUNDARY_CORNERS = ( 2.0  1.0  4.0
                                   -2.0  1.0  4.0
                                   -2.0 -1.0  4.0
                                    2.0 -1.0  4.0 )
```

Specifying half angular extents of the FOV:

```
INS-33333_FOV_SHAPE             = 'RECTANGLE'
INS-33333_FOV_FRAME             = 'FRAME_FOR_INS-33333'
INS-33333_BORESIGHT             = ( 0.0  0.0  1.0 )
INS-33333_FOV_CLASS_SPEC        = 'ANGLES'
INS-33333_FOV_REF_VECTOR        = ( 0.0  1.0  0.0 )
INS-33333_FOV_REF_ANGLE         = 14.03624347
INS-33333_FOV_CROSS_ANGLE       = 26.56505118
INS-33333_FOV_ANGLE_UNITS       = 'DEGREES'
```

Polygonal Fields of View

Consider an instrument with a trapezoidal field of view.



Polygonal FOV Definition

The following sets of keywords and values describe this polygonal field of view:

Specifying boundary vectors explicitly:

```
INS-44444_FOV_SHAPE            = 'POLYGON'
INS-44444_FOV_FRAME            = 'FRAME_FOR_INS-44444'
INS-44444_BORESIGHT            = ( 0.0  0.0  1.0 )
INS-44444_FOV_BOUNDARY_CORNERS = ( 1.0  1.0  4.0
                                  -1.0  1.0  4.0
                                  -2.0 -1.0  4.0
                                   2.0 -1.0  4.0 )
```

- **A polygonal FOV cannot be specified using half angular extents.**

IK Utility Programs

- No IK utility programs are included in the Toolkit
- Two IK utility programs are provided on the NAIF Web site (http://naif.jpl.nasa.gov/naif/utilities.html)
  - OPTIKS displays field-of-view summary for all FOVs defined in a collection of IK files.
  - BINGO converts IK files between UNIX and DOS text formats

Additional Information on IK

- The best way to learn more about IKs is to examine some found in the NAIF Node archives.
  - Start looking here:

    http://naif.jpl.nasa.gov/naif/data_archived.html
- Unfortunately NAIF does not yet have an "I-Kernel Required Reading" document
- But information about IKs is available in other documents:
  - header of the GETFOV routine
  - Kernel Required Reading
  - OPTIKS User's Guide
  - Porting_kernels tutorial
  - NAIF IDs Tutorial
  - Frames Required Reading

Backup

- IK file example
- Computing angular extents from corner vectors returned by GETFOV

Sample IK Data

The following LEMMS1 FOV definition was taken from the Cassini MIMI IK (cas_mimi_v11.ti):

```
Low Energy Magnetospheric Measurements System 1 (LEMMS1)

   Since the MIMI_LEMMS1 detector's FOV is circular and it's diameter is 15.0
   degrees, looking down the X-axis in the CASSINI_MIMI_LEMMS1 frame, we have:
   (Note we are arbitrarily choosing a vector that terminates in the Z=1
   plane.)


                                ^ Y
                                |  ins
                                |
                                |        /|
                                |      /  |
                                |    /    |
                                |  /    o |
                                |/  7.50  |
                                x--------------->
                              X  \        |    Z
                               ins \      |     ins
                                     \    |
                                       \  |
                                         \|

                                |-- 1.0 --|
```

continues

FOV definition from the Cassini MIMI IK (continued):

```
The Y component of one 'boundary corner' vector is:

           Y Component = 1.0 * tan ( 7.50 degrees )
                       = 0.131652498

   The boundary corner vector as displayed below is
   normalized to unit length:
\begindata

 INS-82762_FOV_FRAME  = 'CASSINI_MIMI_LEMMS1'
 INS-82762_FOV_SHAPE  = 'CIRCLE'
 INS-82762_BORESIGHT  = (

  0.0000000000000000   0.0000000000000000  +1.0000000000000000

                        )
 INS-82762_FOV_BOUNDARY_CORNERS = (

  0.0000000000000000  +0.1305261922200500  +0.9914448613738100

                                  )

\begintext
```

Circular FOV Angular Size

Navigation and Ancillary Information Facility

The angular separation between the boundary corner vector and the boresight is the angular size.

FORTRAN EXAMPLE

```
C     Retrieve FOV parameters.
      CALL GETFOV(-11111, 1, SHAPE, FRAME, BSGHT, N, BNDS)

C     Compute the angular size.
      ANGSIZ = VSEP( BSGHT, BNDS(1,1) )
```

C EXAMPLE

```
/*  Define the string length parameter. */
    #define STRSIZ         80

/*  Retrieve the field of view parameters. */
    getfov_c(-11111, 1, STRSIZ, STRSIZ, shape, frame,
             bsght, &n, bnds);

/*  Compute the angular separation. */
    angsiz = vsep_c( bsght, &(bnds[0][0]));
```

Elliptical FOV Angular Size - 1

Navigation and Ancillary Information Facility

The angular sizes are the angular separations between the boresight and the boundary vectors.

FORTRAN EXAMPLE

```
C     Retrieve the FOV parameters from the kernel pool.
      CALL GETFOV(-22222, 2, SHAPE, FRAME, BSGHT, N, BNDS)

C     Compute the angular separations.
      ANG1   = VSEP( BSGHT, BNDS(1,1) )
      ANG2   = VSEP( BSGHT, BNDS(1,2) )

C     The angle along the semi-major axis is the larger
C     of the two separations computed.
      LRGANG = MAX( ANG1, ANG2)
      SMLANG = MIN( ANG1, ANG2)
```

Elliptical FOV Angular Size - 2

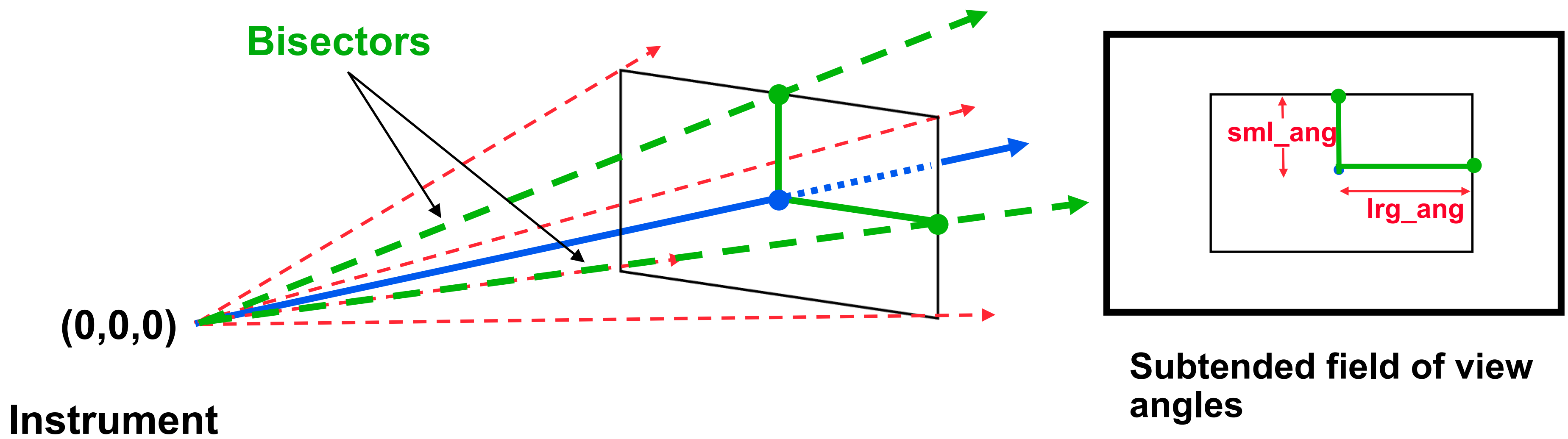
Navigation and Ancillary Information Facility

C EXAMPLE

```
/*  Define the string length parameter. */
    #define STRSIZ         80

/*  Retrieve the FOV parameters from the kernel pool. */
    getfov_c(-22222, 2, STRSIZ, STRSIZ, shape, frame,
             bsght, &n, bnds);

/*  Compute the angular separations. */
    ang1 = vsep_c( bsght, &(bnds[0][0]));
    ang2 = vsep_c( bsght, &(bnds[1][0]));

/*  The angle along the semi-major axis is the larger of the
    two separations computed. */
    if ( ang1 > ang2 ) {
       lrgang = ang1; smlang = ang2; }
    else {
       lrgang = ang2; smlang = ang1; }
```

Rectangular FOV Angular Size - 1

The angular extents of the FOV are computed by calculating the angle between the bisector of adjacent unit boundary vectors and the boresight.



Rectangular FOV Angular Size - 2

FORTRAN EXAMPLE

```
C     Retrieve FOV parameters from the kernel pool.
      CALL GETFOV(-33333, 4, SHAPE, FRAME, BSGHT, N, BNDS)

C     Normalize the 3 boundary vectors
      CALL UNORM(BNDS(1,1), UNTBND(1,1), MAG)
      CALL UNORM(BNDS(1,2), UNTBND(1,2), MAG)
      CALL UNORM(BNDS(1,3), UNTBND(1,3), MAG)

C     Compute the averages.
      CALL VADD(UNTBND(1,1), UNTBND(1,2), VEC1)
      CALL VSCL(0.5, VEC1, VEC1)

      CALL VADD(UNTBND(1,2), UNTBND(1,3), VEC2)
      CALL VSCL(0.5, VEC2, VEC2)

C     Compute the angular separations
      ANG1   = VSEP( BSGHT, VEC1 )
      ANG2   = VSEP( BSGHT, VEC2 )

C     Separate the larger and smaller angles.
      LRGANG = MAX( ANG1, ANG2)
      SMLANG = MIN( ANG1, ANG2)
```

Rectangular FOV Angular Size - 3

Navigation and Ancillary Information Facility

C EXAMPLE

```
/*  Define the string length parameter. */
    #define STRSIZ          80

/*  Retrieve the FOV parameters from the kernel pool. */
    getfov_c(-33333, 4, STRSIZ, STRSIZ, shape, frame,
             bsght, &n, bnds);

/*  Normalize the 3 boundary vectors. */
    unorm_c(&(bnds[0][0]), &(untbnd[0][0]), &mag);
    unorm_c(&(bnds[1][0]), &(untbnd[1][0]), &mag);
    unorm_c(&(bnds[2][0]), &(untbnd[2][0]), &mag);

/*  Compute the averages */
    vadd_c(&(untbnd[0][0]), &(untbnd[1][0]), vec1);
    vscl_c(0.5, vec1, vec1);
    vadd_c(&(untbnd[1][0]), &(untbnd[2][0]), vec2);
    vscl_c(0.5, vec2, vec2);

/*  Compute the angular separations. */
    ang1 = vsep_c( bsght, vec1);
    ang2 = vsep_c( bsght, vec2);

/*  Separate the larger and smaller angles. */
    if ( ang1 > ang2 ) {
       lrgang = ang1; smlang = ang2; }
    else {
       lrgang = ang2; smlang = ang1; }
```