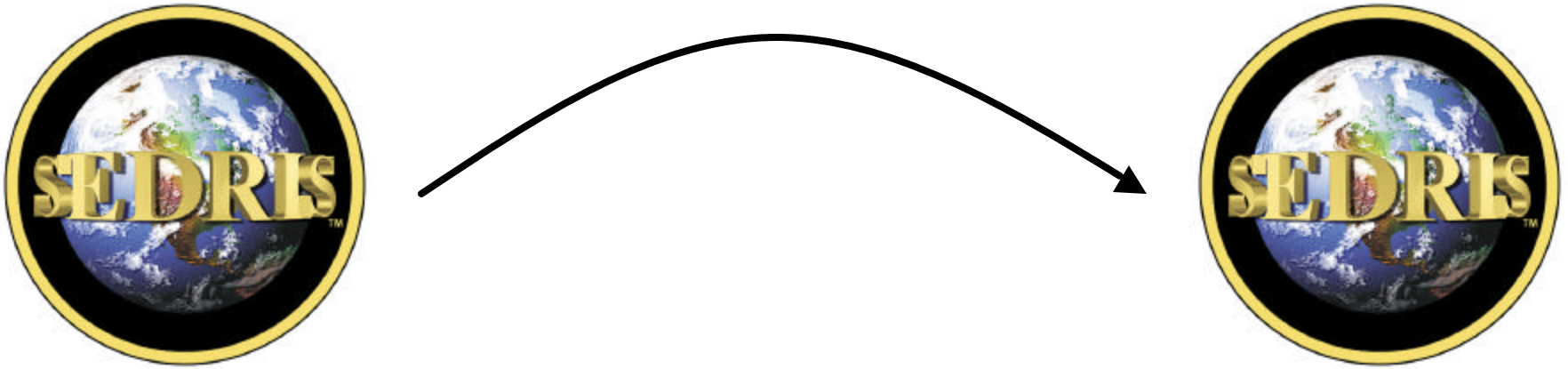


The SRM For Programmers

January 6, 2004

Lake Buena Vista, Florida

Cameron D. Kellough

Research Engineer

SRI International

cameron.kellough@sri.com

Goals

- Identify SRM API Implementations
- Identify Important Concepts
- Identify Spatial Operations
- Identify Details Affecting Applications
- Work Examples Performing Operations

SRM API Implementations

- SRM C API (functional methodology)
- SRM C++ API (object methodology)
- SRM JAVA API (object methodology)

- The remainder of this presentation covers conceptual information about the API's and uses examples from the C++ API
- Other API versions are similar

Important Concepts

- Spatial Reference Frame (SRF)
  - SRF Parameters
  - Object Reference Model (ORM)
  - Transformation Parameters (H_{SR})
- Spatial Primitives
  - Coordinate
  - Direction
  - Orientation

SRF: General Definition

- An SRF specifies the context in which to interpret spatial primitives (Coordinates, Directions, and Orientations)
- An SRF is to spatial information what a BNF grammar is to a programming language
- An SRF embodies the information traditionally distributed between one or more of the following:
  - Earth Model
  - Coordinate System
  - Datum
  - Datum Shift Parameters

SRF: In the SEDRIS Lexicon

- In SEDRIS, an SRF consists of:
  - SRF Parameters
    - Describe parameters such as the central meridian of a map projection SRF
  - An Object Reference Model (ORM)
    - Determines the position-space embedding
    - Optionally describes the shape of the Celestial Object being modeled (Many celestial objects modeled besides earth)
  - Optionally: Transformation Parameters (H_{SR})
    - Describe relationship between chosen ORM for the Celestial Object being modeled and other ORM's for the same object
      - Works by Expressing Chosen ORM ? Reference ORM

SRF's without H_{SR}

- A source SRF with only SRF Parameters and ORM disambiguates spatial primitives in its context allowing the construction of corresponding target spatial primitives in any target SRF with the same ORM for which the transformed source spatial primitive remains defined
- No "Horizontal Datum Shifts" Possible

SRF's with H_{SR}

- A SRF that specifies SRF parameters ORM, and Transformation Parameters H_{SR} disambiguates a source spatial primitive in its context allowing the construction of corresponding target spatial primitives in any target SRF for which the transformed source spatial primitive remains defined
- "Horizontal Datum Shifts" Possible

SRF: Conclusion

- Application must supply values for SRF parameters and ORM to create SRF; should supply H_{SR} if known
- API returns new SRF object for later use in:
  - The Creation of Coordinates
  - The Creation of Directions
  - The Creation of Orientations
  - Spatial Operations
- If no H_{SR} is provided then no "Horizontal Datum Shifts" are available for primitives in this context
- SRF is immutable once created by the API

Coordinate: General Definition

- A Coordinate contains a tuple of numbers which in the context of an SRF can be interpreted as a spatial location
- In SEDRIS, any given Coordinate can be in the context of exactly one SRF
- The API uses three types of Coordinates
  - Two Dimensional (2-tuple)
  - Surface (2-tuple)
  - Three Dimensional (3-tuple)

Dimensionality Boot Camp: Week 1

- CS Type: Two Dimensional
  - Location trapped on a sheet of paper
  - Two axes
  - Both axes co-planar
  - No notion of height whatsoever allowed by the CS type

Dimensionality Boot Camp: Week 2

- CS Type: Surface
  - Location trapped on a two dimensional rubber sheet stretched around [or occasionally tangent to] a three dimensional object
  - Two orthogonal axes only [NOT three]
  - Notion of Height But . . .
    - Constrained by CS type: to surface of the two dimensional rubber sheet
    - Cannot be specified by Application

Dimensionality Boot Camp: Week 3

- CS Type: Three Dimensional
  - Location unconstrained
  - Really three dimensional: Three axes
  - Location not constrained to a surface
  - Third coordinate component [height for SRF's defining height] not constrained by CS type

API: Coordinate

- A Coordinate is specific to a given SRF
- A Coordinate is created by the appropriate method on an SRF object of the desired type
- Once created by the API, the values in a Coordinate may be modified at will by the application

API: Coordinate Specifics

- API permits Coordinate values to be changed by the application:
  - Allows OOP applications to create object once and change the values rather than creating a new coordinate for each operation
  - API cannot thus ensure the validity of a given coordinate from API call to API call
  - Application responsible for the contents of the coordinate

Direction: General Definition Part 1

- A direction contains two three-tuples
  - The first three-tuple is a Reference Location coordinate specifying the origin of the direction vector
  - The second three-tuple contains a vector specifying a direction at the given Reference Location
- In SEDRIS, a given Direction can be in the context of exactly one SRF

Direction: General Definition Part 2

- The meaning of the second (bound vector) tuple is determined by the characteristics of the SRF in whose context the Direction is being created
- The three different interpretations are for:
  - True Vector Spaces
  - Translated/Rotated True Vector Spaces
  - Non Vector Spaces

Direction: General Definition Part 3

- Vector Spaces
  - Celestiocentric
- Reference Location Doesn't affect direction because vectors are invariant under translation
- [Free] Vector (is bound to origin) and pointy end is at the three-tuple representing direction from {0,0,0}

Direction: General Definition Part 4

- Translated/Rotated True Vector Spaces
  - Local Tangent Space Euclidean [LTSE]
- Reference Location is LTSE SRF origin
- Bound vector tuple bound to LTSE SRF origin
- Pointy end at bound vector three tuple representing direction from LTSE SRF origin

Direction: General Definition Part 4A

- In an LTSE SRF
  - X points at the LTSE azimuth + ?/2
  - Y points at the LTSE azimuth
  - Z points Up as defined by Normal to Ellipsoid Surface

Direction: General Definition Part 5

- Non Vector Spaces use constructed CLTSE
  - Celestiodetic
  - Map Projections
    - Mercator, Transverse Mercator, Oblique Mercator
    - Equidistant Cylindrical, Lambert Conformal Conic
- Reference Location used as origin in construction of new CLTSE SRF
- Bound vector tuple bound to CLTSE SRF's origin
- Pointy end is at bound vector three tuple representing direction from CLTSE SRF origin

Direction: General Definition Part 5A

- CLTSE Like LTSE Case with zero azimuth
  - X points East
  - Y points North
  - Z points Up as defined by Normal to Ellipsoid Surface

Interpreting Directions

- In the context of CLTSE, Celestiodetic, and Map Projection SRF's, the bound vector tuple {0,1,0} represents true north at any Reference Location! Bound vector tuple {0,0,1} represents Up.
- A North Pointing Direction in the context of an CLTSE, Celestiodetic, or Map Projection SRF, when transformed to the Celestiocentric SRF, is unique! (except on the equator)
- Map north equals true north plus Convergence of the Meridian.

API: Direction

- Direction is specific to given SRF
- Direction created by appropriate method on SRF object of desired type
- Once created by the API, the reference location and vector become immutable
- Transforming a direction creates a new direction in the context of a target SRF
- Directions are Normalized

Orientation: General Definition

- An Orientation contains a three-tuple of numbers and a 3x3 Matrix
  - The three-tuple contains a Reference Location coordinate
  - The 3x3 Matrix is a rotation matrix that specifies the values required to rotate a mathematical vector or matrix in the context of this SRF to a particular absolute spatial orientation
- In SEDRIS, any given Orientation can be in the context of exactly one SRF

Orientation: General Definition (2 of 4)

- Orientations like Directions are interpreted differently depending on the SRF to which they belong
- Since an Orientation behaves just like a compilation of three Directions, the three contexts for interpreting Orientations are analogous to those for Directions

Orientation: General Definition (3 of 4)

- Celestiocentric
  - Matrix is compilation of three direction vector components (free vectors w/respect to origin)
  - Reference Location Immaterial
- LTSE
  - Matrix is compilation of three direction vector components (bound vectors w/respect to Reference Location) rotated around the Z axis by the azimuth value

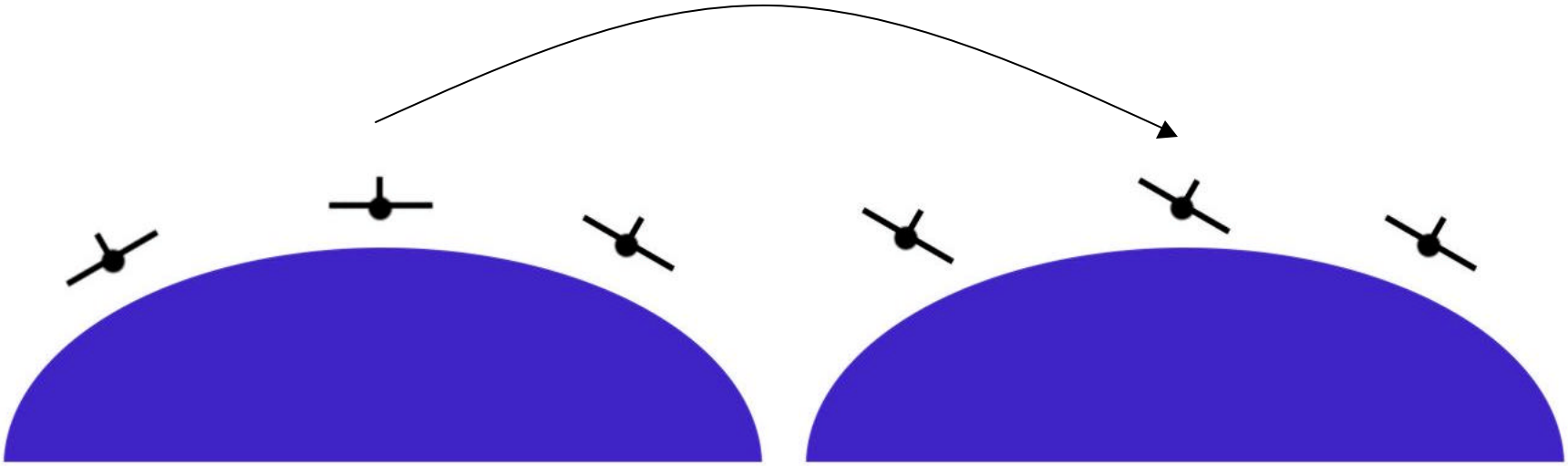
Orientation: General Definition (4 of 4)

- Non Vector Spaces using CLTSE
  - Matrix is compilation of three direction vector components (bound vectors w/respect to constructed CLTSE origin)

Orientation: Example (1 of 2)

- Imagine 3 Airplanes
  - Flight Leader [FL] (in context of CLTSE 1)
  - Wingman 1 [W1] (in context of CLTSE 2)
  - Wingman 2 [W2] (in context of CLTSE 3)
- FL applies an orientation that Rolls him 30 degrees right in CLTE1
  - Leader transforms this orientation to CLTSE 2
- W1 applies new orientation, rolls 60 deg right in CLTSE 2
  - Leader transforms this orientation to CLTSE 3
- W2 applies new orientation, doesn't roll at all
- All planes end up in same absolute orientation
- [This contrived example rotates about only one axis]

Orientation Example: (2 of 2)



Absolute Spatial Orientation is equalized for vectors in different SRF's:

- Apply an orientation in the context of a source SRF to a vector in the same SRF
- Transform the source orientation to the context of a target SRF
- Apply the transformed orientation to the vector in the target SRF

Interpreting Orientations

- Transforming an Orientation containing the Identity Matrix between source and target SRF's creates Orientation whose matrix values can be used to transform a direction from the context of a source SRF to the context of a target SRF
- Applications may store the matrix values from this computation to perform direction conversion without invoking the API

API: Orientation

- Orientation is specific to given SRF
- Orientation created by appropriate method on SRF object of desired type
- Transforming an Orientation creates a new Orientation in the context of a target SRF
- Once created by API, the reference location and matrix in an Orientation become immutable

Spatial Operations: 3D Coordinates

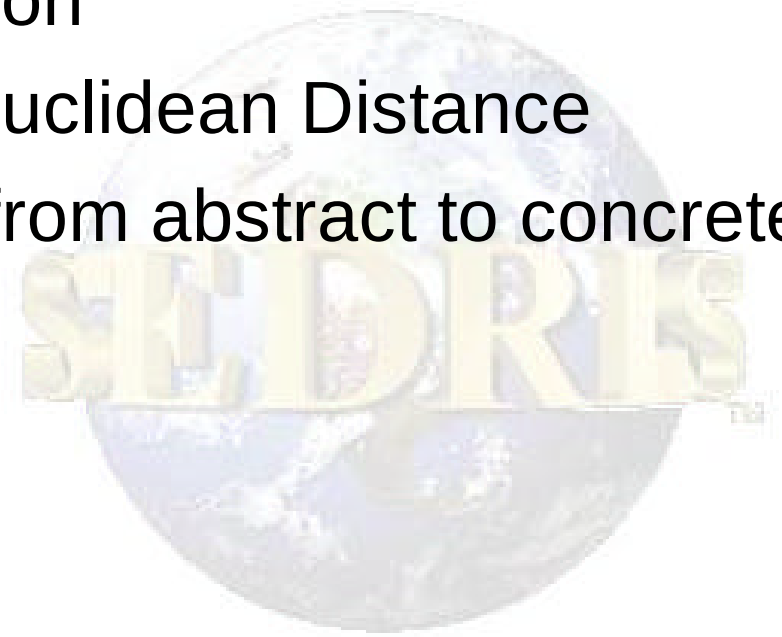
- Transformation
  - Conversion
  - ORM Change [Horizontal Datum Shift]
- Calculate Euclidean Distance
- Truncation to Surface Coordinate
- Instancing from abstract to concrete space

Spatial Operations: Surface Coordinates

- Surface Coordinates
  - Calculate Euclidean Distance
  - Calculate Geodesic Distance
  - Calculate Vertical Separation Offset
  - Promote to 3D Coordinate
  - Additionally for Surface Map Coordinates:
    - Calculate Convergence of the Meridian
    - Calculate Map Azimuth
    - Calculate Point Scale

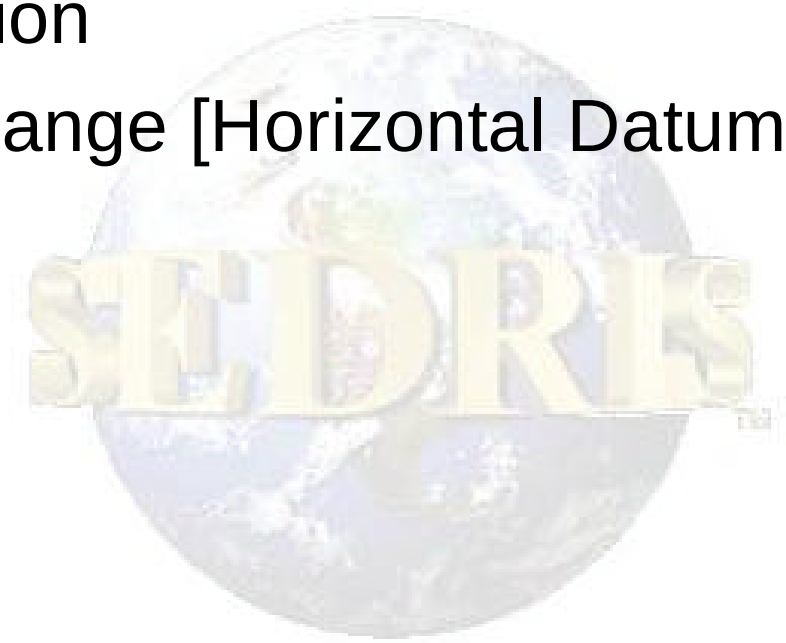
Spatial Operations: 2D Coordinates

- Transformation
  - Conversion
- Calculate Euclidean Distance
- Instancing from abstract to concrete space



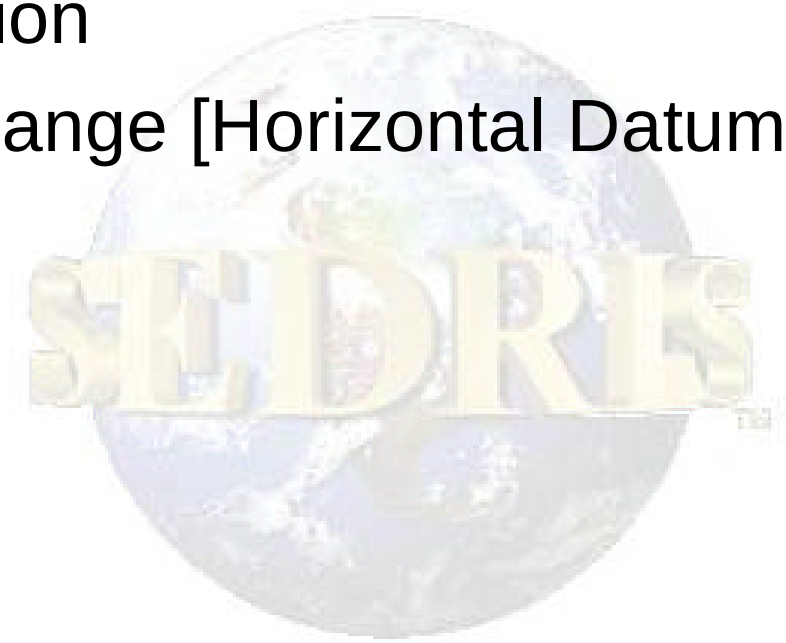
Spatial Operations: Directions

- Transformation
  - Conversion
  - ORM Change [Horizontal Datum Shift]



Spatial Operations: Orientations

- Transformation
  - Conversion
  - ORM Change [Horizontal Datum Shift]

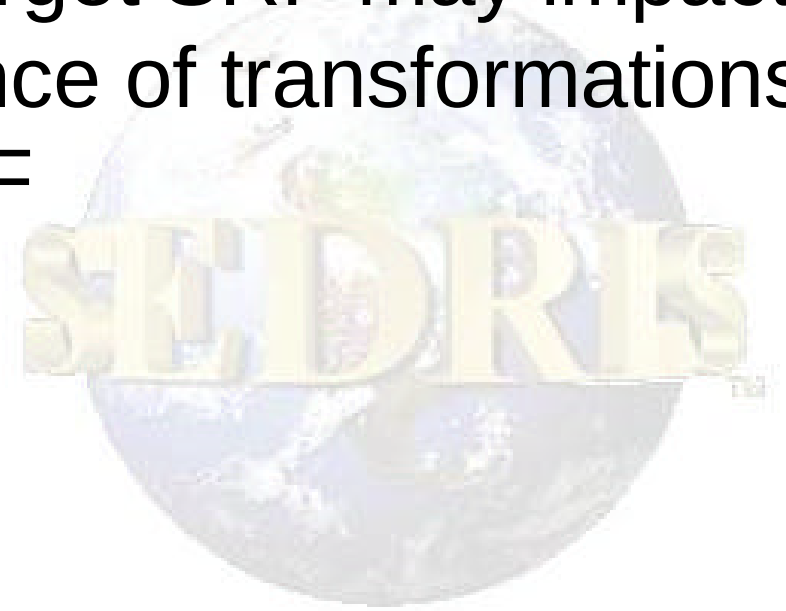


Implementation Details Affecting Applications

- No Transformation initialization is necessary at the Application level
  - Transformation Initialization Done Internally!
- The SRM API caches initialization data associated with Transformations on the target SRF

Implementation Details: Performance

- Large numbers of source SRF's cached on a given target SRF may impact the performance of transformations to that target SRF



SRM Implementation Details: Threads

- The SRM implementation is thread safe if and only if:
  - No thread uses objects that were created by another thread

One Possible Application Optimization

- The SRM API can produce the transformation matrix to change directions and orientations from source to target SRF
- A performance sensitive Application with software access to vector hardware could compute a transformation matrix with the API and apply it to the data through vector hardware for a substantial improvement when transforming large direction datasets

On to the Examples!!!!

- The following examples are written using the SEDRIS SRM version 4.0.0 Alpha.
- The semantics of specifying ORM data and H_{ST} are expected to deviate from that shown here in the upcoming final release
  - Expected changes are due to changes in ISO 18026 on which the SRM implementation is based

Ex 1: Changing TM Coordinate to CD Coordinate

- A Transverse Mercator Coordinate is changed to a Celestiodetic [called geodetic elsewhere] Coordinate by the following steps:
  - Create TM SRF
  - Create CD SRF
  - Create TM Coordinate
  - Create CD Coordinate
  - Do the Transformation

Creating a TM SRF

- Create TM SRF
  - ORM
  - Origin Longitude
  - Origin Latitude
  - Central Scale Factor
  - False Easting
  - False Northing

Creating a CD SRF

- Create CD SRF
  - ORM

Changing SRF's

- Use ChangeCoordinate3DSRF method
  - Method located on Target Celestiodetic SRF
- To print the returned values
  - Must cast to the specific concrete type!
- Example 1 also reconverts the point to the original Source SRF showing numerical accuracy round trip for this operation to be less than .03 mm

ORM Changes: Example 1½

- Were an ORM change [Horizontal Datum Shift] required, it's still easy:
  - Put the desired ORM in the Target SRF
  - Now when the transformation runs, it now transparently performs the ORM change
- Lets Try it using:
  - SRM_ORM_N_AM_1927_WESTERN_US

Input Mechanisms

- Example 1 shows SRF constructors taking SRF parameters as function arguments
- Example 2 shows SRF constructors can also take structures

Error Handling: Exceptions

- Example 3 demonstrates catching an SRM exception
  - Application tries creating a TM direction
  - Unfortunately SRM version 4.0.0 Alpha encounters an error in this example
  - SRM API throws srm::Exception
  - Application catches the exception, prints the description, and continues

Conclusion:

- The SRM API provides a high-level black-box interface to complex algorithms that perform Spatial Operations

- SRF's and spatial primitives are easily created & passed to routines providing all Spatial Operations Defined in ISO 18026!

The End