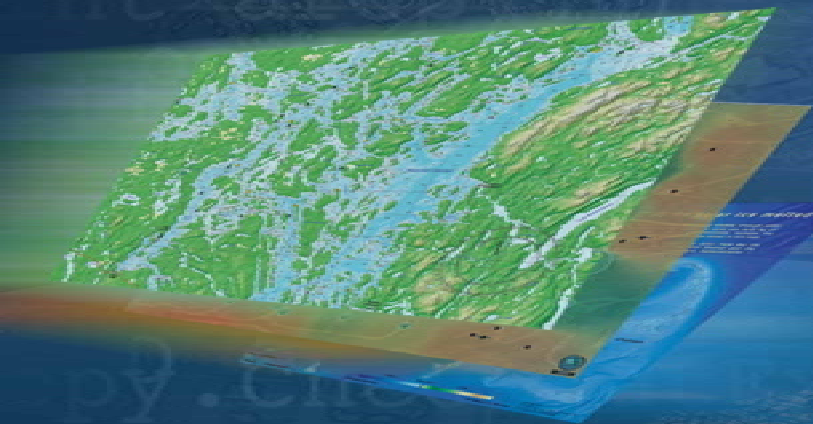


# ESRI Developer Summit

March 22–25, 2010  
Palm Springs, CA

## Understanding and Using Geometry, Projections, and Spatial Reference Systems in ArcGIS

*Jim TenBrink, Melita Kennedy, David Raleigh*



# Introduction

- **Spatial references**
- **The geometry types**
- **Relational operators**
- **Using geometry efficiently**
- **What's new at 10?**

# Spatial references

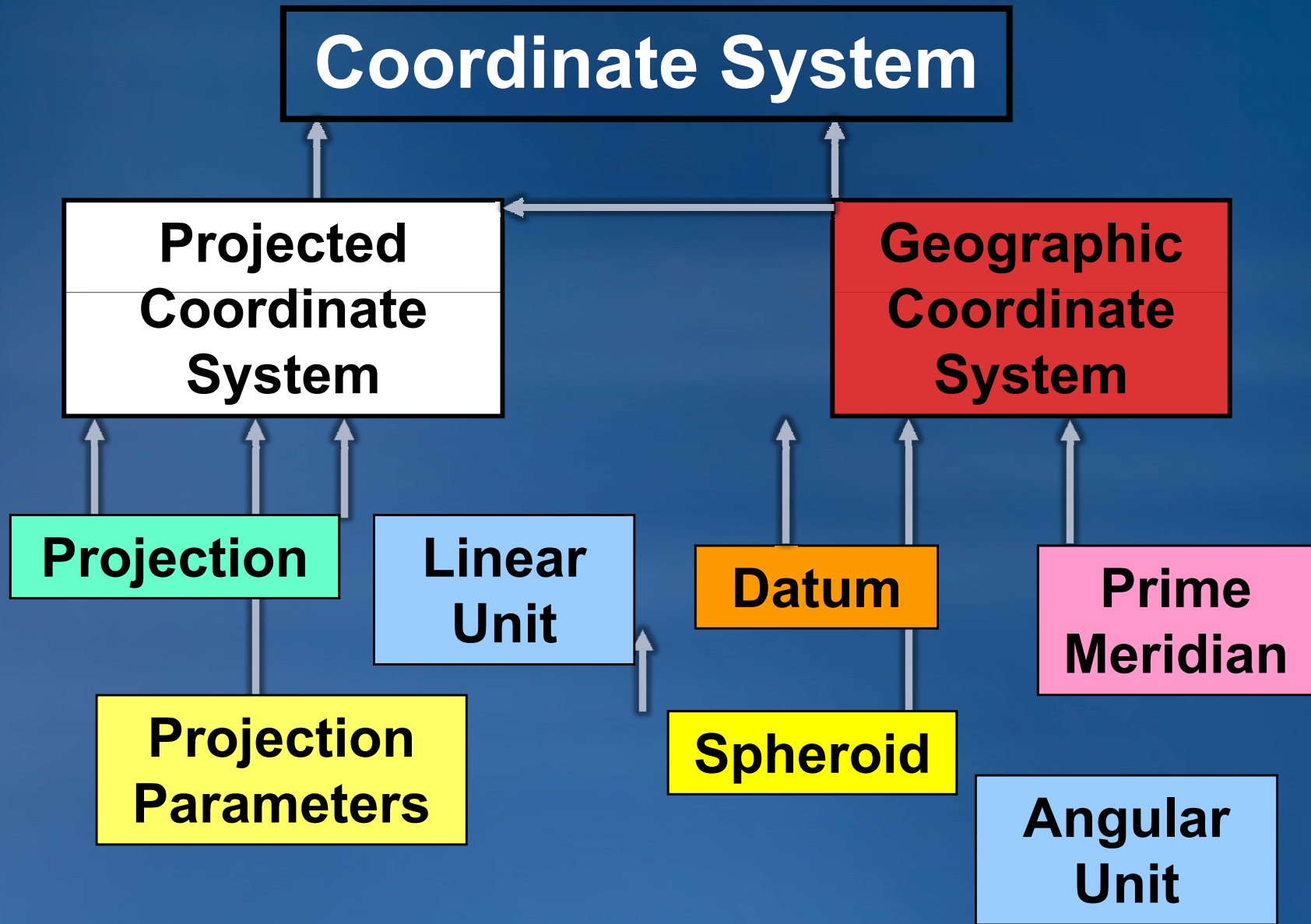
**Spatial references in ArcGIS define these key properties that affect your coordinates:**

- **Round Earth model**
- **Coordinate system (Projection)**
- **Coordinate Resolution Grid**
- **Cluster Tolerances**

**In addition, geographic transformations move coordinates between different earth models**

# Spatial Reference Objects

- **Spatial References**
  - Coordinate systems: projected, geographic, vertical, unknown
  - PCS, GCS, VCS, UCS
- **Units**
  - Linear and Angular
- **GeoTransformations**
  - Geographic (datum) transformations

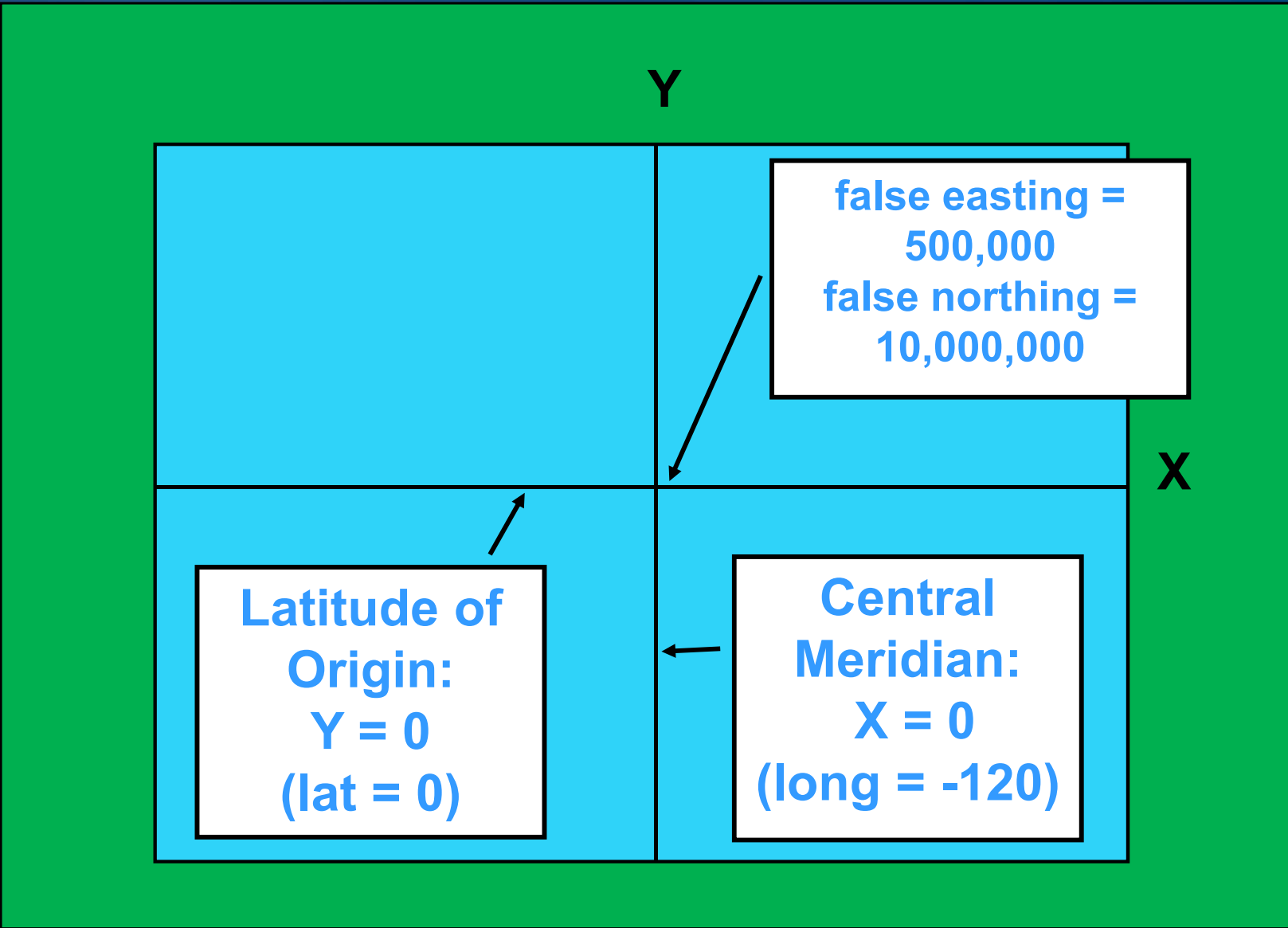


## Well-Known Text String

```
GEOGCS[ "GCS_WGS_1984",  
  DATUM[ "D_WGS_1984",  
    SPHEROID[ "WGS_1984", 6378137.0, 298.257223563] ],  
  PRIMEM[ "Greenwich", 0.0],  
  UNIT[ "Degree", 0.0174532925199433] ],
```

## Well-Known Text String

```
PROJCS[ "Test",  
  GEOGCS[ "GCS_WGS_1984",  
    DATUM[ "D_WGS_1984",  
      SPHEROID[ "WGS_1984", 6378137.0, 298.257223563] ],  
    PRIMEM[ "Greenwich", 0.0],  
    UNIT[ "Degree", 0.0174532925199433] ],  
  PROJECTION[ "Mercator " ],  
  PARAMETER[ "Central_Meridian", -120.0],  
  PARAMETER[ "Standard_Parallel_1", 0.0],  
  PARAMETER[ "False_Easting", 500000.0],  
  PARAMETER[ "False_Northing", 10000000.0],  
  UNIT[ "Foot", 0.3048] ]
```



**Y**

false easting =  
500,000  
false northing =  
10,000,000

**X**

Latitude of  
Origin:  
Y = 0  
(lat = 0)

Central  
Meridian:  
X = 0  
(long = -120)



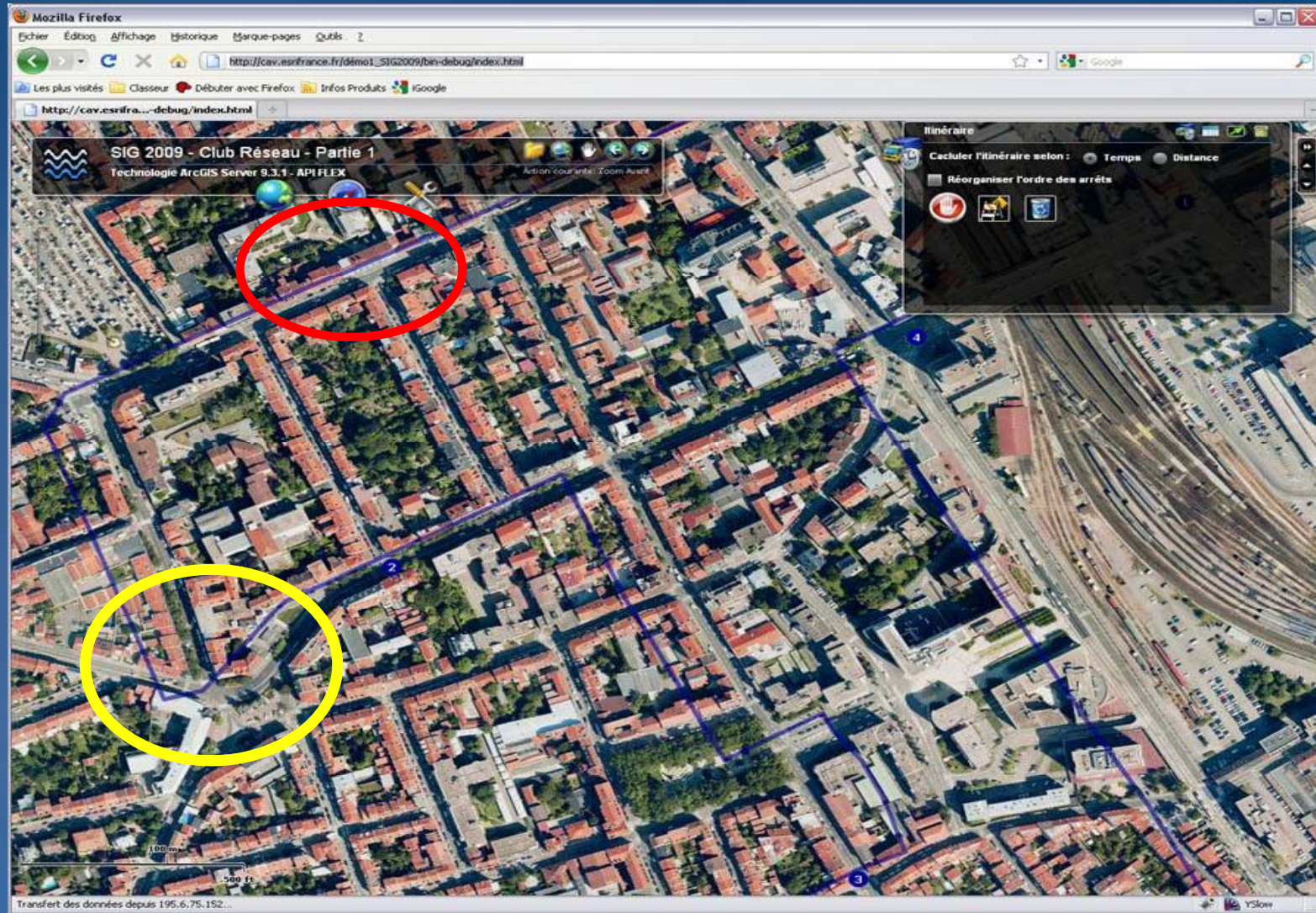
# Well-known IDs

- Each object has an ID and a macro
  - 4326
  - PE\_GCS\_WGS\_1984
  - esriSRGeoCS\_WGS1984
- IDs < 32766 are EPSG-assigned
  - EPSG Geodetic parameter Dataset, <http://www.epsg.org>
- IDs > 32767 are ESRI-assigned
- IDs may change
  - ESRI → EPSG
  - EPSG → EPSG
  - Old IDs will still work
- Macros more stable

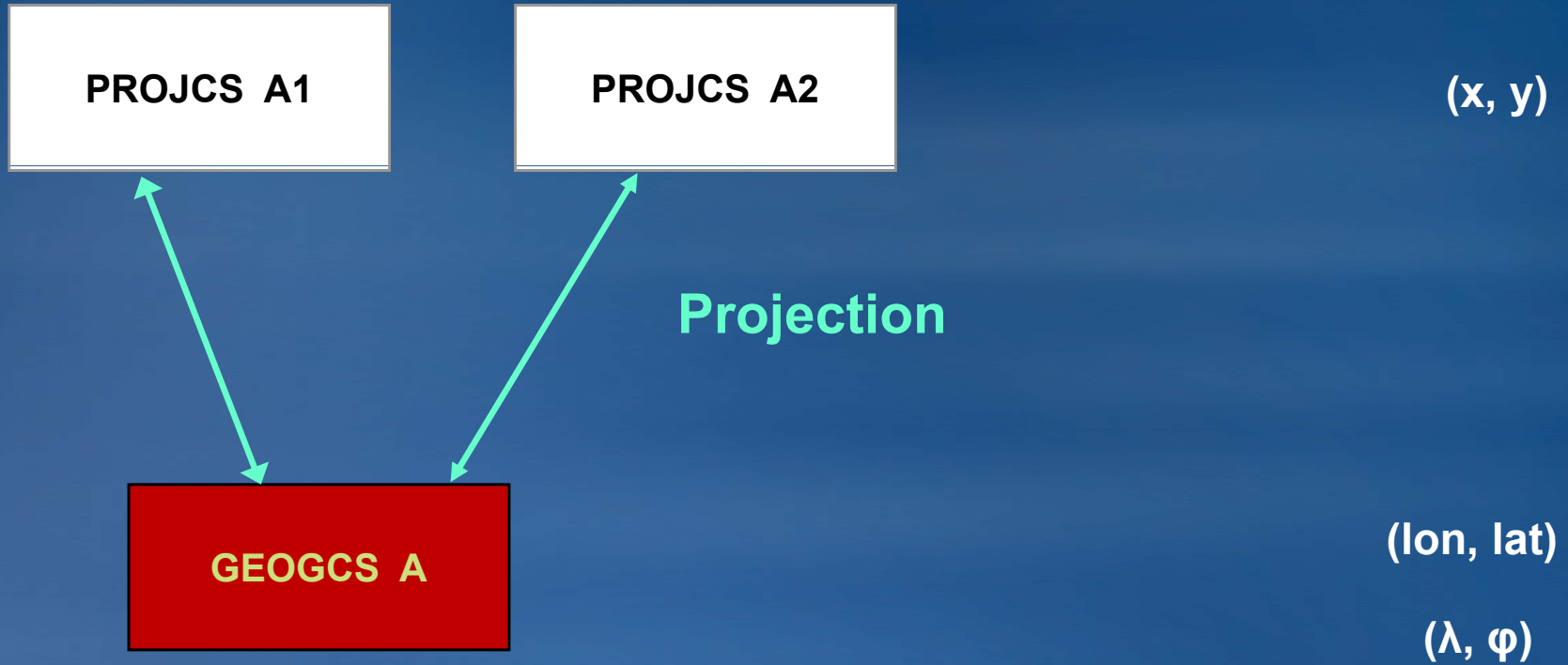
# Geographic transformations

- “datum” transformations, geotransformations
- Convert between GCS
- Includes unit, prime meridian, and spheroid changes
- Defined in a particular direction
- All are reversible
  
- Not all web APIs support geotransformations!

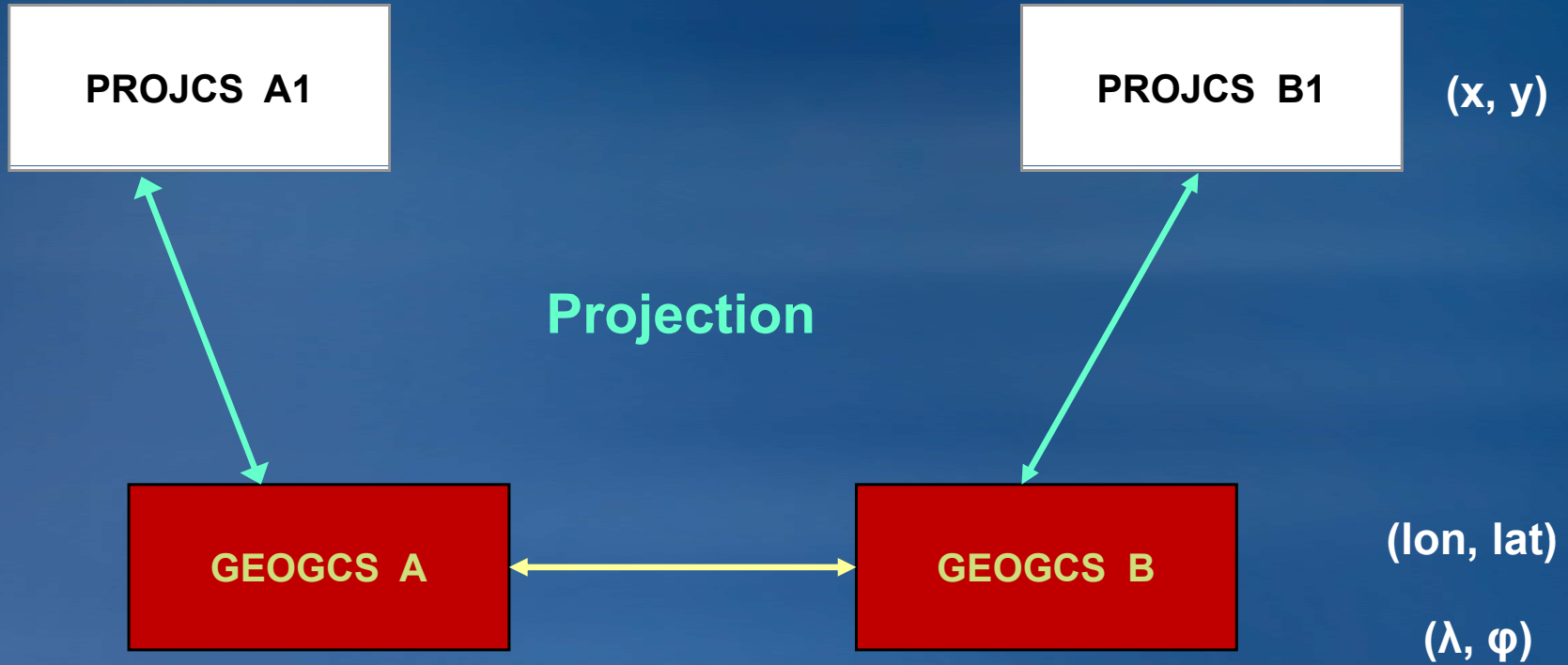
# ED50 versus WGS84



# Conversion Flow



# Conversion Flow



**Geographic Transformation**

# Web Mercator

- **Online mapping services use a sphere-only Mercator**
- **Two ways to emulate it**
  - Sphere-based GCS
  - Projection that can force sphere equations
- **Mathematically EQUAL**

```
PROJCS["WGS_1984_Web_Mercator",  
  GEOGCS["GCS_WGS_1984_Major_Auxiliary_Sphere",  
    DATUM["D_WGS_1984_Major_Auxiliary_Sphere",  
      SPHEROID["WGS_1984_Major_Auxiliary_Sphere",  
        6378137.0,0.0]],  
    PRIMEM["Greenwich", 0.0],  
    UNIT["Degree", 0.0174532925199433]],  
  PROJECTION["Mercator"],  
  PARAMETER["False_Easting", 0.0]  
  PARAMETER["False_Northing", 0.0],  
  PARAMETER["Central_Meridian", 0.0],  
  PARAMETER["Standard_Parallel_1", 0.0],  
  UNIT["Meter", 1.0]]
```

```
# 102113
```

```
PROJCS["WGS_1984_Web_Mercator_Auxiliary_Sphere",  
  GEOGCS["GCS_WGS_1984",  
    DATUM["D_WGS_1984",  
      SPHEROID["WGS_1984",6378137.0,  
        298.257223563]],  
    PRIMEM["Greenwich", 0.0],  
    UNIT["Degree", 0.0174532925199433]],  
  PROJECTION["Mercator_Auxiliary_Sphere"],  
  PARAMETER["False_Easting", 0.0],  
  PARAMETER["False_Northing", 0.0],  
  PARAMETER["Central_Meridian", 0.0],  
  PARAMETER["Standard_Parallel_1", 0.0],  
  PARAMETER["Auxiliary_Sphere_Type", 0.0],  
  UNIT["Meter", 1.0]]
```

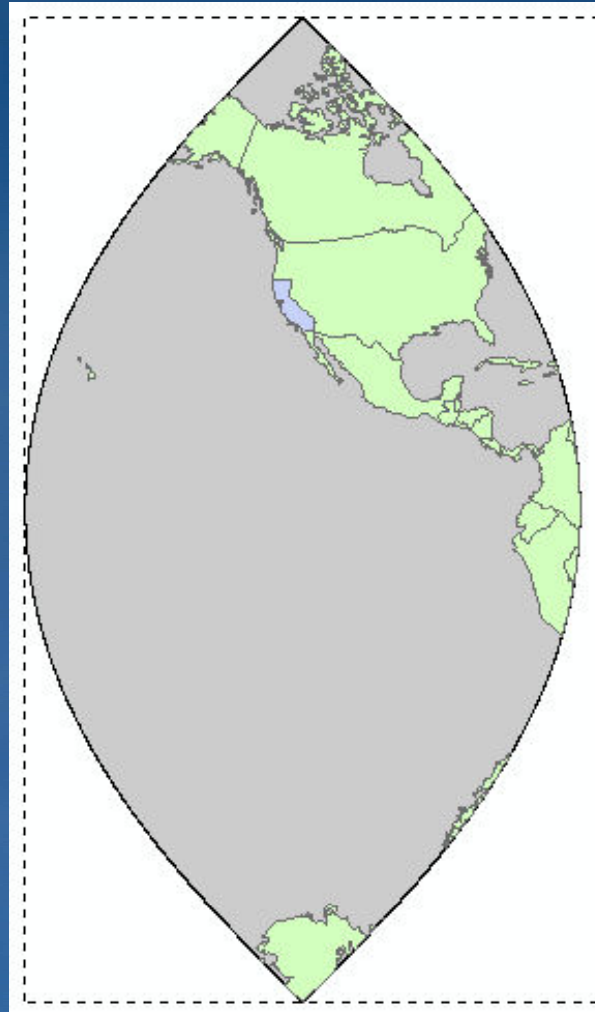
```
#3857    (old #102100)
```



# Coordinate resolution grids

- **What is a grid?**
  - It's where your points can be placed
  - Start with a horizon polygon
  - Put a square over it
  - Slice it horizontally and vertically
  - High precision (lots of slices, today) vs. low precision (not so many slices, legacy)
- **Majority of ArcGIS operations snap to this grid**
- **Datasources except shapefiles snap to this grid when storing features**

# Example: Horizon for NAD 1983 UTM Zone 11

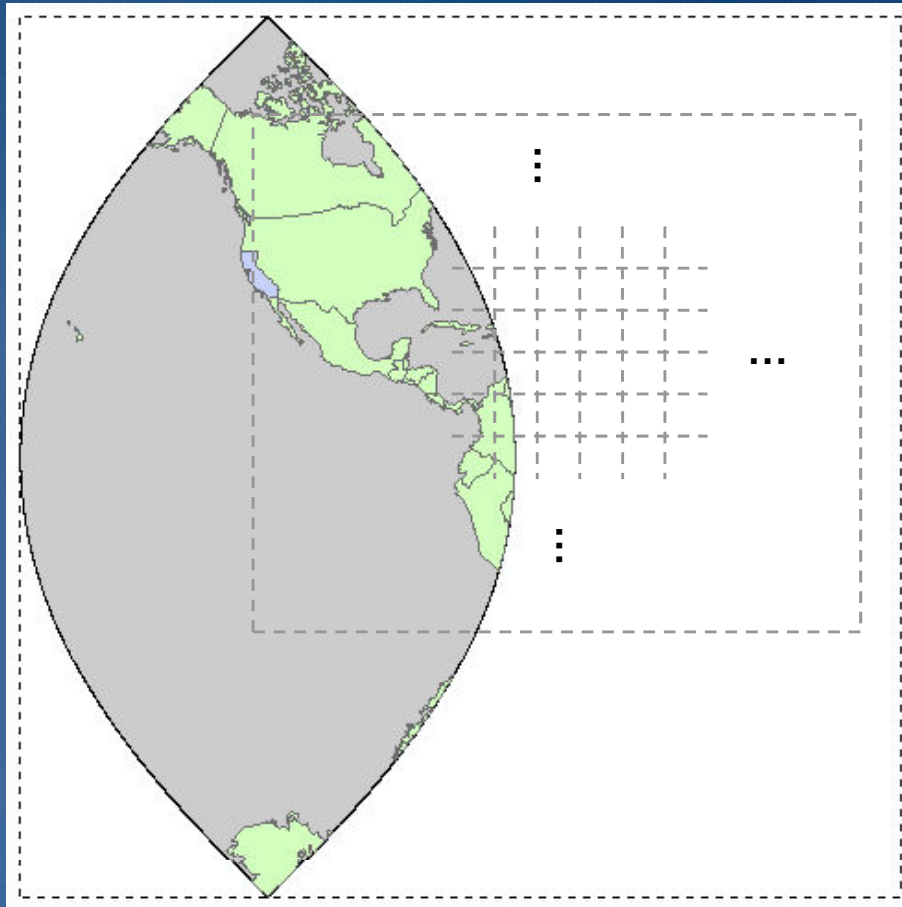


False Origin:

Xmin = -5,136,135 m

Ymin = -9,950,490 m

# Impose a grid



$X_{max} = 14,890,535 \text{ m}$

$Y_{max} = 9,992,387 \text{ m}$

$XY \text{ resolution} = 2.2 \times 10^{-9} \text{ m}$

**False Origin:**

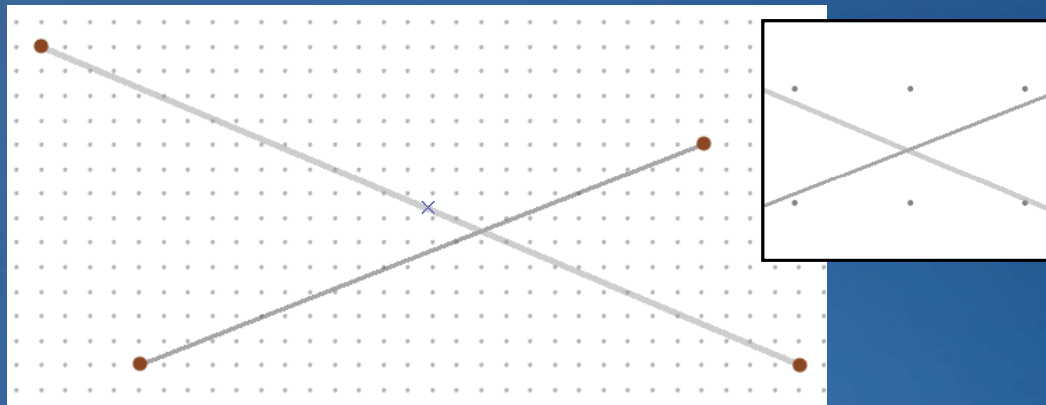
$X_{min} = -5,136,135 \text{ m}$

$Y_{min} = -9,950,490 \text{ m}$

# Demo 1: Visualize the grid, snapping

# The cluster tolerance

- Topo and rel-ops snap input coordinates to the grid, but...
- A larger *cluster tolerance*, and an iterative procedure called *integration* (aka cracking/clustering), are used to make consistent spatial decisions with vector data
- We recommend the ct to be  $1/10^{\text{th}}$  of the min separation between geometries and the resolution to be  $1/10^{\text{th}}$  of the ct
- The default cluster tolerance is 1mm and the default resolution is 1/10 mm



# The core geometry types

- **“Top Level Types” can be stored in a geodatabase or shapefile**
  - Points
  - Multipoints
  - Polylines
  - Polygons
  - Multipatches
- **Polylines are sequences of Paths**
- **Polygons are sequences of Rings**
- **Paths and rings are sequences of Segments**
  - Lines
  - Circular arcs
  - Elliptical arcs
  - Bezier curves

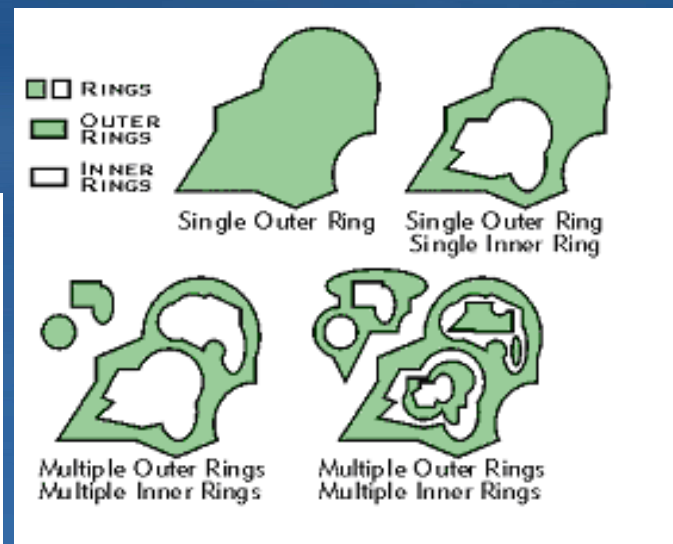
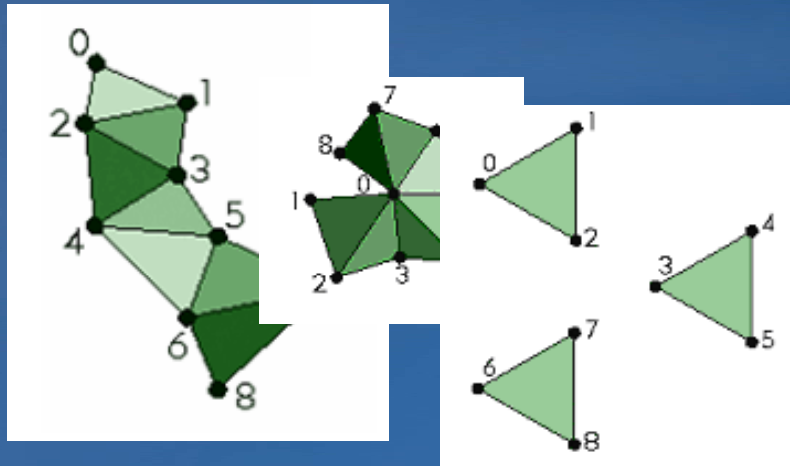
# Other objects

- **Envelopes**
- **Triangle sets, strips and fans can be stored in multipatches, in addition to rings**
- **Geometry Service**
- **Buffer Construction service**
- **Geometry Environment**
- **Geometry Bags**
- **WKSPoint (structure)**
- **Transformation objects**



## Polygons and their associated envelopes

Triangle strips, fans, sets;  
organizing rings in a multipatch





# Simple vs. non-Simple geometries

- **Multipoints** – each point must be unique. Only uses resolution grid
- **Polylines** – segment length  $>$  resolution, verts permitted when z-aware
- **Polygons (same as Arc/Info regions)** – uses cluster tolerance
  - inside-outside-boundary
  - Finite area
  - multiple rings, outer clockwise, inner counterclockwise
  - Rings can touch at points
  - Note difference between connected components and outer components

# Enforcing simplicity

- **Low level**

- **Polylines**

- **IPolyline6::SimplifyNonPlanar (10.0)**
    - **IPolyline::SimplifyNetwork (merges valence two parts)**

- **Polygons**

- **ITopologicalOperator::Simplify**
    - **SimplifyPreserveFromTo, less efficient**

- **Higher level**

- **IFeatureSimplify::SimplifyGeometry**

- **If Polygon, simplify (preserve from/to)**
    - **Project to dataset (doesn't clear isSimple bit)**
    - **Simplify again**
      - **Preserve from/to for polygons**
      - **non-planar for polylines**
      - **snap to grid for points/multipoints**

## Demo 2: simple geometries

# Relational Operators

- **5 core operators, defined in terms of interior, boundary, exterior**
- **A point has an interior only**
- **A polyline's boundary is its set of path endpoints, its interior is everything except the endpoints**
  - we do not planarize polylines before applying this definition
- **A polygon's boundary is its rings; its interior is the inside of its rings**

## Relational Operators (cont.)

- **Disjoint** – boundary and interiors disjoint
- **In** – the usual, but interior intersection can't be empty
  - same for contains, we have **Contains/WithinEx** to avoid this extra constraint
- **Touch** – shapes are not disjoint but interiors are
- **Cross** – interior intersection dimension is  $<$  max dimension of inputs and there is no containment
- **Overlap** – interior intersection dimension = dimension of inputs and there is no containment

## Relational Operators (cont.)

- In ArcGIS server/desktop/engine these operations are evaluated relative to the cluster tolerance. In the ESRI geometry implementations accessed via SQL they are evaluated relative to the resolution.
- The definitions of these operations have been extended to work with vertical segments in polylines: “2D hybrid polyline used to evaluate 2D relations”
- Examples:
  - a vertical polyline is “in” a 2D polygon if the point projection (footprint) of the polyline is in the polygon.
  - Two vertical polylines intersect (not disjoint) if they project to the same point

# Demo 3: Relational Operators + Map Spatial Reference

## Correctness and Efficiency Tips

- Always tag geometry with a spatial reference (sr)
- Tag geometry bag with an sr before adding geometries to it
- Know when you're sharing references to rings/paths/segments
- Shapefiles can't store curves, vertex IDs, or some properties of Multipatches (e.g. textures)
- Use point and segment enumerators to access existing polylines and polygons (recycling)
- Use array-based methods (IGeometryBridge/2) to add data to geometries
- Use WKSPoint structures when possible
- Watch your object burn rate



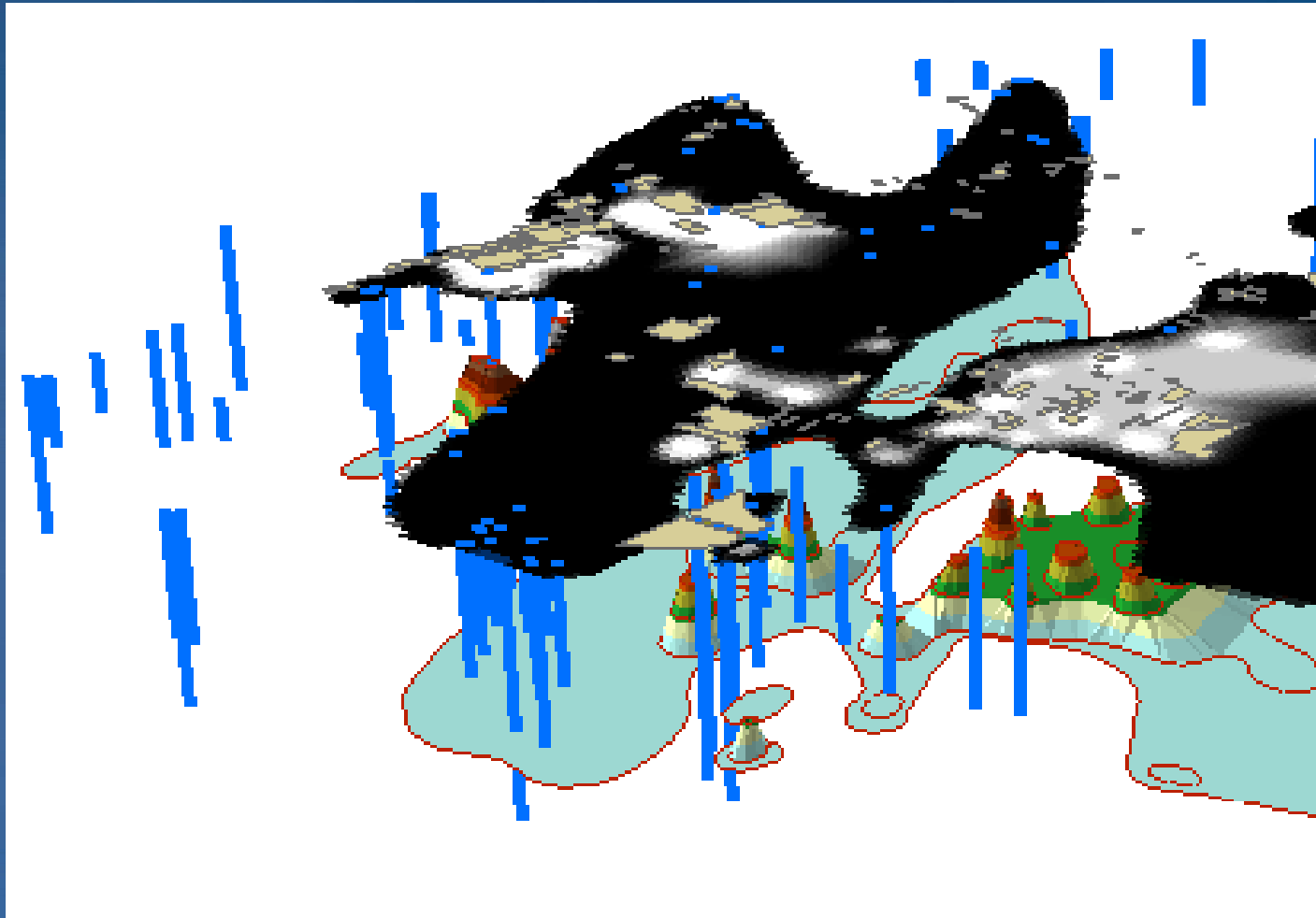
# What's New at 10.0?

- Vertical segments in polylines
- Geodetic constructions (IConstructGeodetic, IPolycurveGeodetic): editor UI, measurement tool UI
- Clothoid (transition) spirals: SDK only
- Methods for string $\leftrightarrow$ double coordinate conversion (Point::IConversionNotation)
- New datum transformation methods
- Arcpy python site package exposes some geometry methods (relational operators, length, area)
- 3D awareness in select by location (near 3D, 3D hit testing, some 3D relational operators)

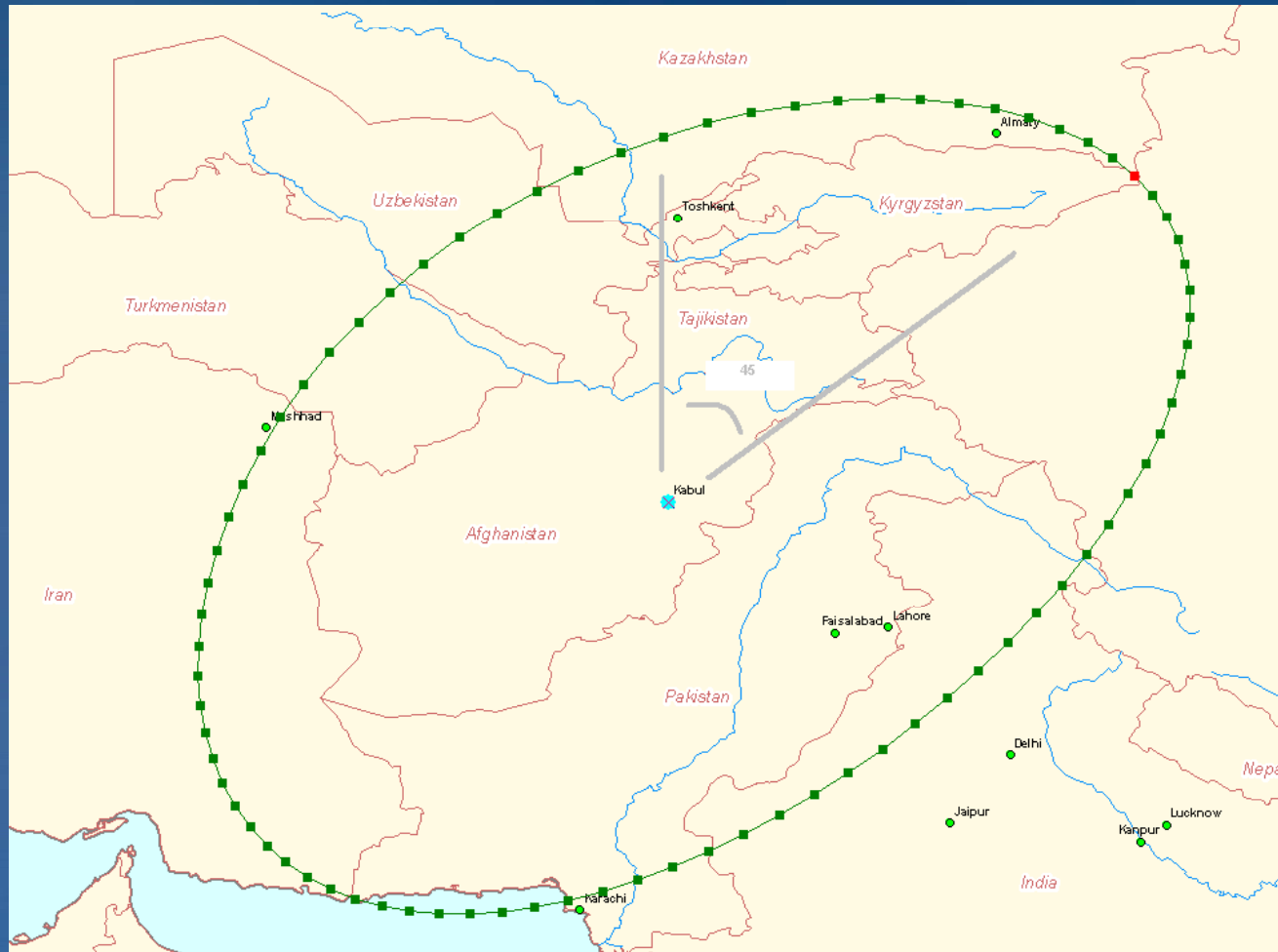
## What's New (cont.)

- Projection optimizations (precompute when horizon line clipping will be needed)
- Geometry server methods to support web editing and some geodesic operations:  
<http://servicesbeta.esri.com/arcgis/sdk/soap/>,  
<http://servicesbeta.esri.com/ArcGIS/rest/services/Geometry/GeometryServer>

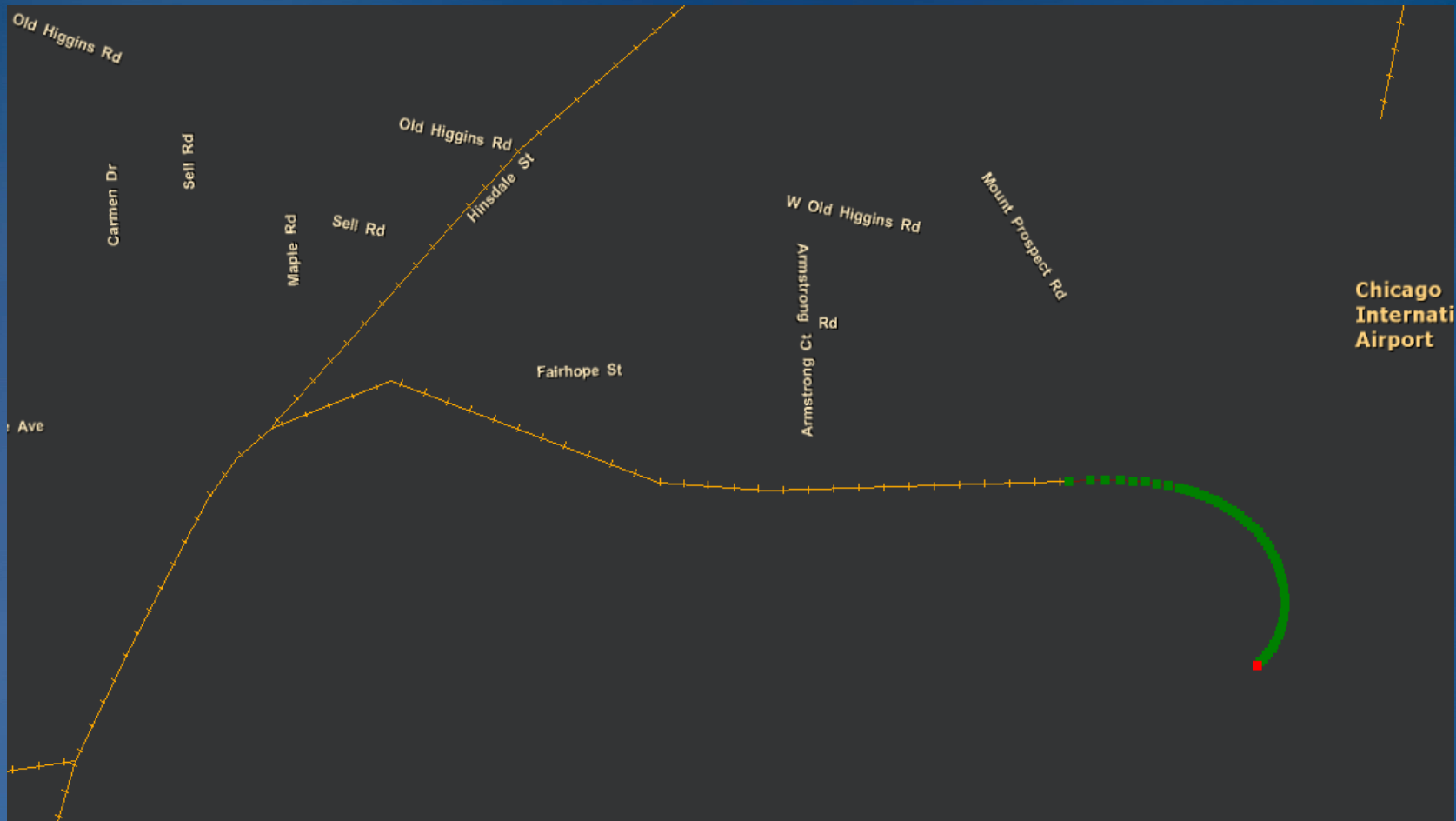
# Vertical polylines in geodatabase



# What's new example: Geodesic ellipses



# What's new example: Spirals



# Notation transformation

- **Supported notations**

- **MGRS**      - **Military Grid Reference System**
- **USNG**      - **US National Grid**
- **UTM**        - **Universal Transverse Mercator**
- **GEOREF**   - **Geographic Reference System**
- **GARS**      - **Global Area Reference System**
- **DMS**        - **Degree Minute Second**
- **DDM**        - **Degree Decimal-Minute**
- **DD**         - **Decimal Degree**

- **Conversions**

- **Notation to coordinates**
- **Coordinates to notation**

# Exposing Projection Engine C-API in .Net

- **User Request:**

- Provide a method for sorting geographic transformations by usefulness

- **Projection Engine Solution:**

- Analyze user input and EPSG metadata to return a sorted list of geographic transformations
  - Use extent of data being transformed (user input)
  - Use area of use of each GCS (EPSG metadata)
  - Use area of use of the available geographic transformations (EPSG metadata)
  - Use the accuracy of the available geographic transformation (EPSG metadata)

# **Demo 4: Areas of use for coordinate systems and transformations**



# Resources

- ESRI White paper: “[Understanding Coordinate Management in the Geodatabase](#)”
- The scripts in the demo map document for this presentation
- ArcGIS Desktop help, Professional Library
  - Map Projections Guide Book
  - Data Management → Editing Data → Fundamentals of editing → *About editing data in a different projection (projecting on the fly).*
- [Resolution and tolerance default values](#)
- ArcObjects SDK help
  - Geometry library overview topic
  - *Working with vertical polyline segments* (accessible from geometry library overview topic)
- *A small set of formal topological relationships suitable for end-user interaction, Clementini, et al.*

**Questions?**

# Session Evaluations Reminder

**Session Attendees:**

**Please turn in your session evaluations.**

***... Thank you***