AutoCAD® Map 3D 2013 Platform API Training

Partha Sarkar

Sr. Developer Consultant

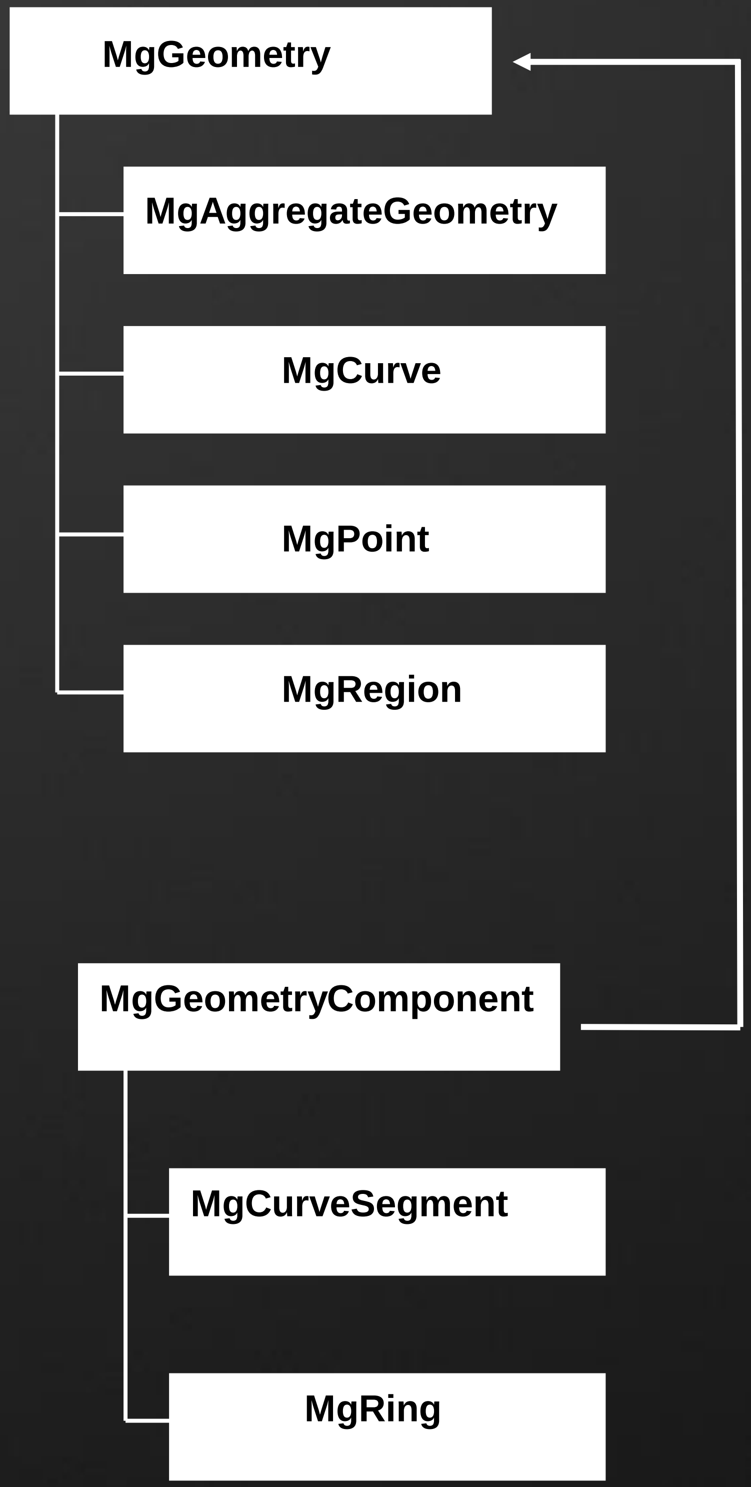
Geometry

Contents

- Overview of Geometry Objects
- Geometry Format Conversion
- Coordinate systems
- Buffer objects

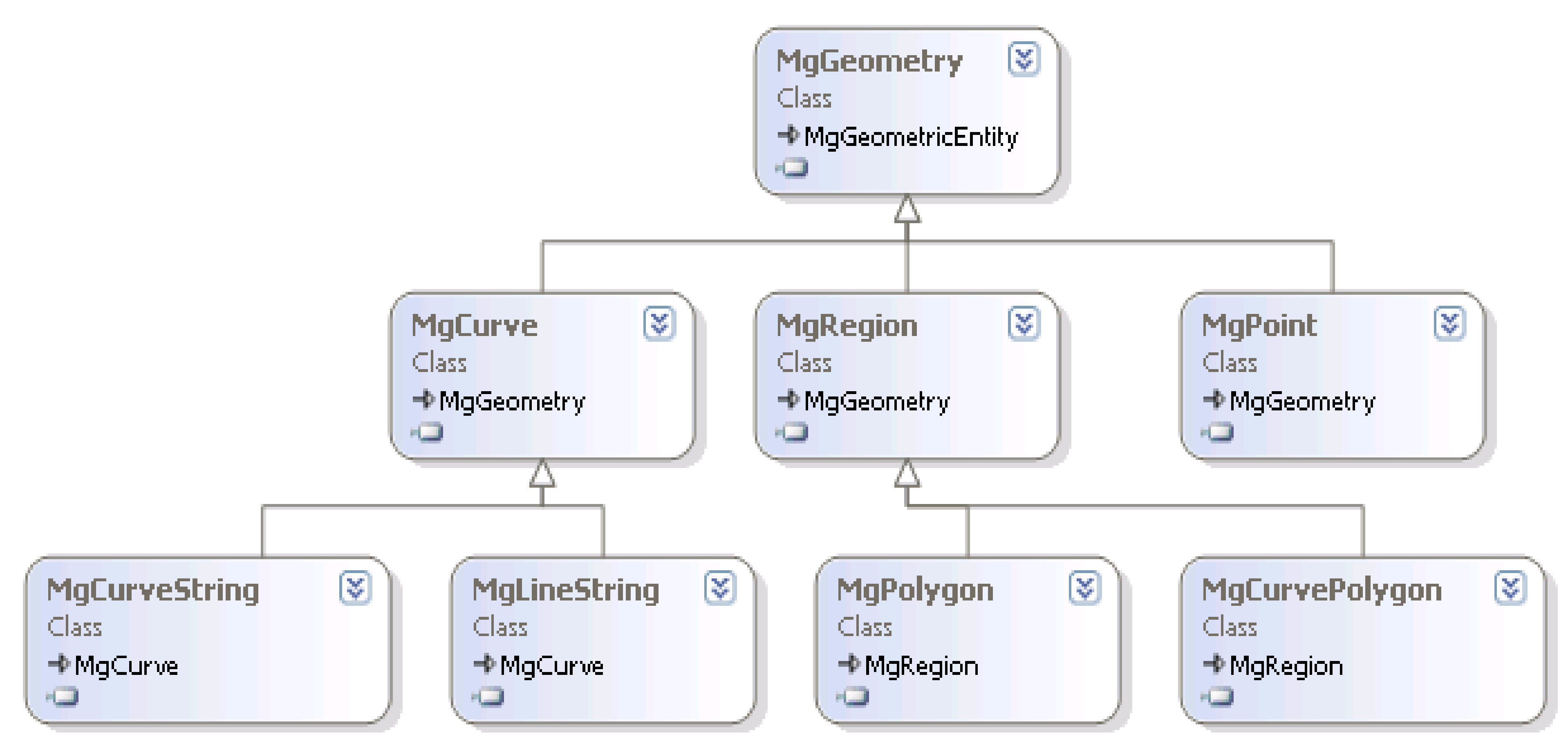
Geometry Objects

- Geometry types have MgGeometry as their base class
- Geometry objects are constructed using geometry component types
  - Base class of geometry component types is MgGeometryComponent
- Geometry types
  - MgAggregateGeometry
  - MgCurve
  - MgPoint
  - MgRegion
- Geometry component types
  - MgCurveSegment
  - MgRing



Geometry Objects

Simple Geometry Types



- MgPoint – a single location in coordinate space

- MgCurveString – Series of connected curve segments

- MgLineString – Series of connected line segments

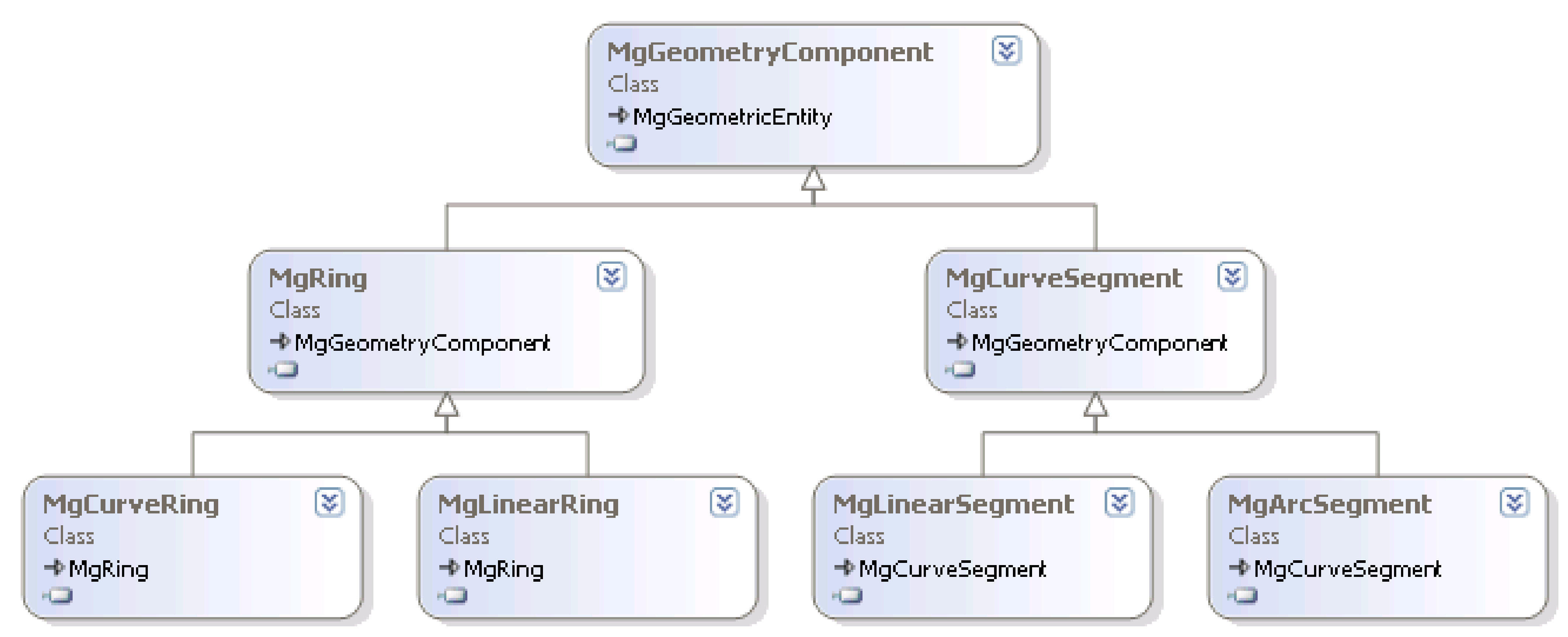
- MgPolygon – A polygon with sides formed from line segments

- MgCurvePolygon – A polygon with sides formed from curve segments

Autodesk

Geometry Objects

Geometry component types

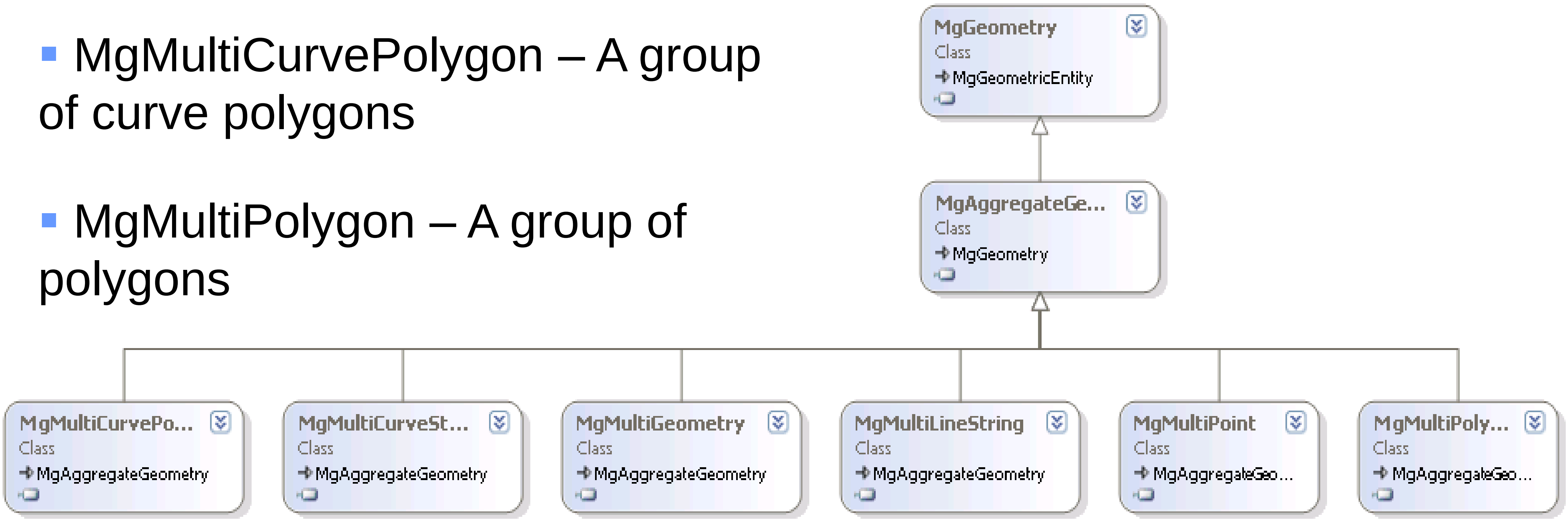


- MgCurveRing - Used in the construction of MgCurvePolygon
- MgArcSegment - Used in the construction of MgCurveString and MgCurveRing
- MgLinearRing - Used in the construction of MgPolygon object
- MgLinearSegment - Used in the construction of MgCurveString and MgCurveRing

Geometry Objects

Aggregate Geometry Types

- MgMultiCurvePolygon – A group of curve polygons
- MgMultiPolygon – A group of polygons



- MgMultiPoint – A group of points
- MgMultiLineString – a group of line strings
- MgMultiCurveString– A group of curve strings
- MgMultiGeometry – a group of simple geometry objects of any type

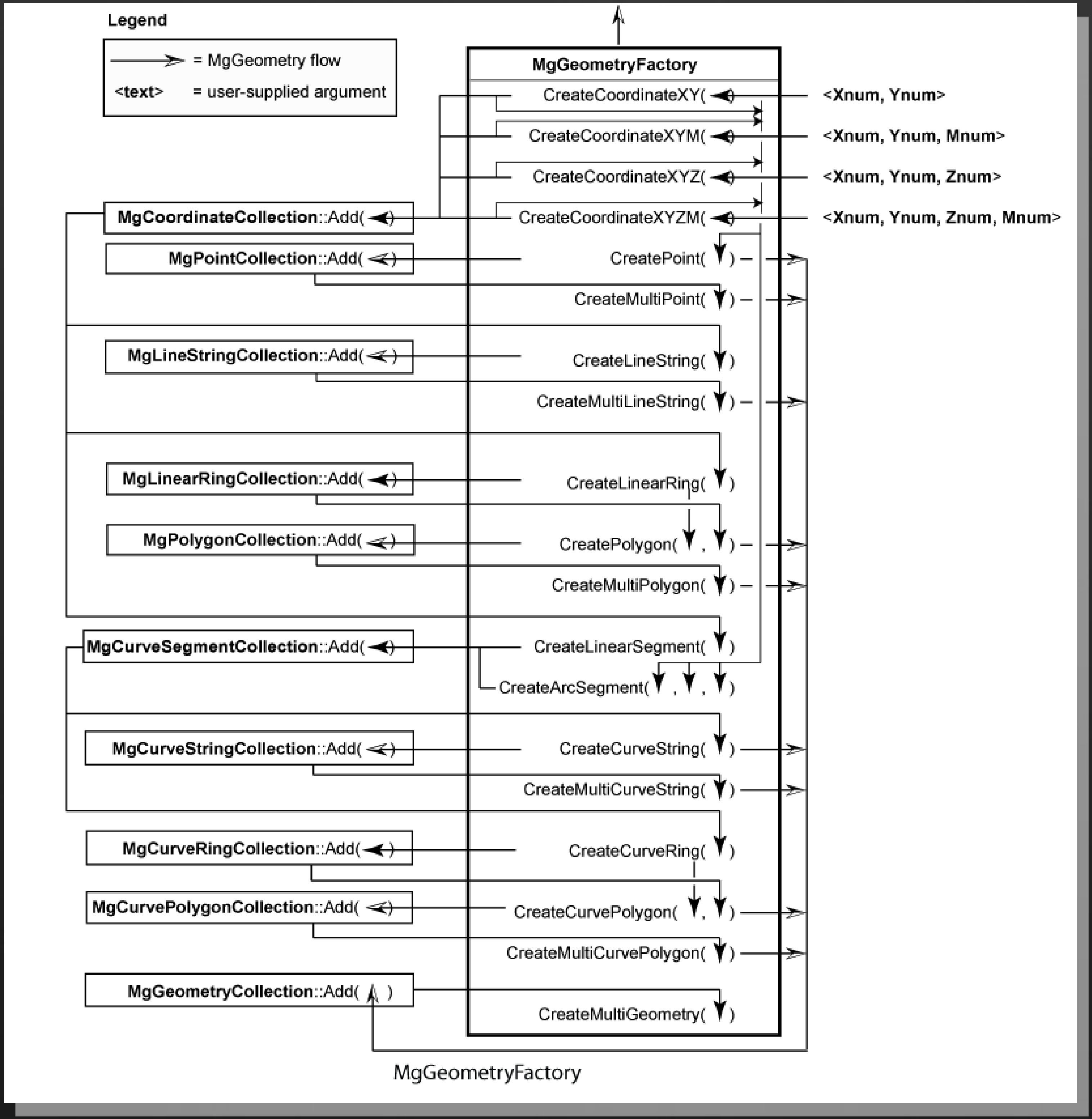
Geometry Objects

Constructing Geometry Objects

- Use MgGeometryFactory for constructing MgGeometry objects

```
void CreateGeometryFromClickedPoint(out MgPoint Pt, double X, double Y)
      {
                 Pt = null;
                 MgGeometryFactory geometryFactory = new MgGeometryFactory();
                 // create a coordinate
                MgCoordinate coordinate = geometryFactory.CreateCoordinateXY(X,Y);
                // create a point
                Pt = geometryFactory CreatePoint(coordinate);
      }
```

Geometry Objects



http://mapguide.osgeo.org/files/mapguide/docs/webapi/index.htm

Geometry Objects

Constructing Geometry Objects

- Use MgGeometryFactory for constructing MgGeometry objects
- Some General Rules
  - First and last coordinates in MgLinearRing coordinates collection must be identical
  - When adding MgCurveSegment objects (MgArcSegment objects or MgLinearSegment) to MgCurveSegmentCollection, last coordinate in an MgCurveSegment object must be identical to the first coordinate of the next MgCurveSegment object

Geometry Objects

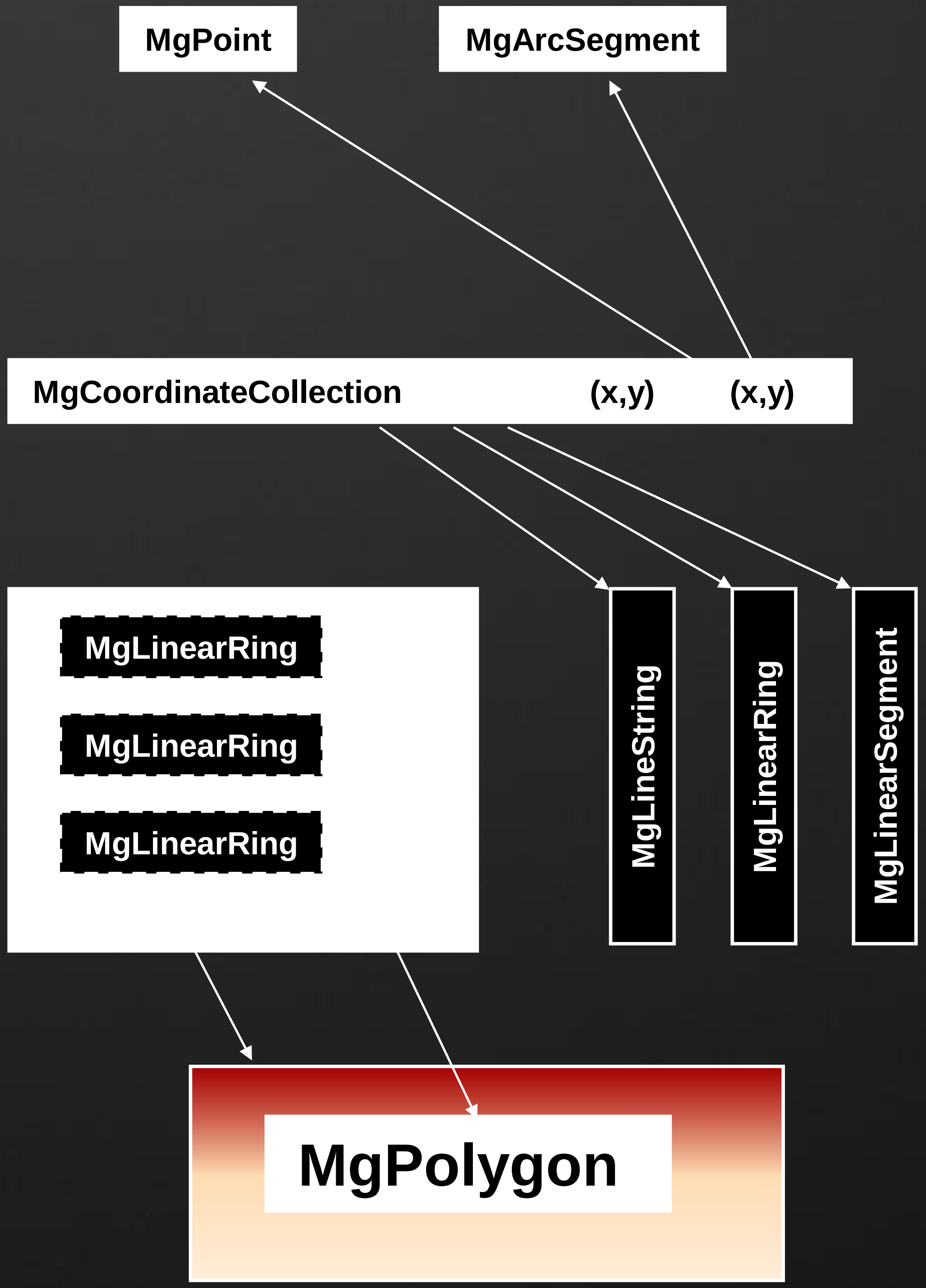
Constructing Geometry Objects

- Some General Rules
  - When adding coordinates to MgLinearRing coordinates collection to be used as an **exterior** ring in an MgPolygon, direction of traversal must be counterclockwise, for **interior** ring, direction of traversal must be clockwise
  - When adding MgCurveSegment objects to MgCurveSegmentCollection that defines an MgCurveRing to be used as an **exterior** ring in an MgCurvePolygon, direction of traversal must be counterclockwise, for **interior** ring, direction of traversal must be clockwise

Geometry Objects

Constructing Geometry Objects

- Use MgCoordinate objects to create MgPoint geometry and MgArcSegment geometry component
- Use MgCoordinateCollection to create MgLineString, MgLinearRing, and MgLinearSegment
- Use MgLinearRing to construct an MgPolygon object’s external boundary and an optional MgLinearRingCollection of MgLinearRing geometry components to define "holes' in the containing ring



Geometry Objects

Constructing a polygon object

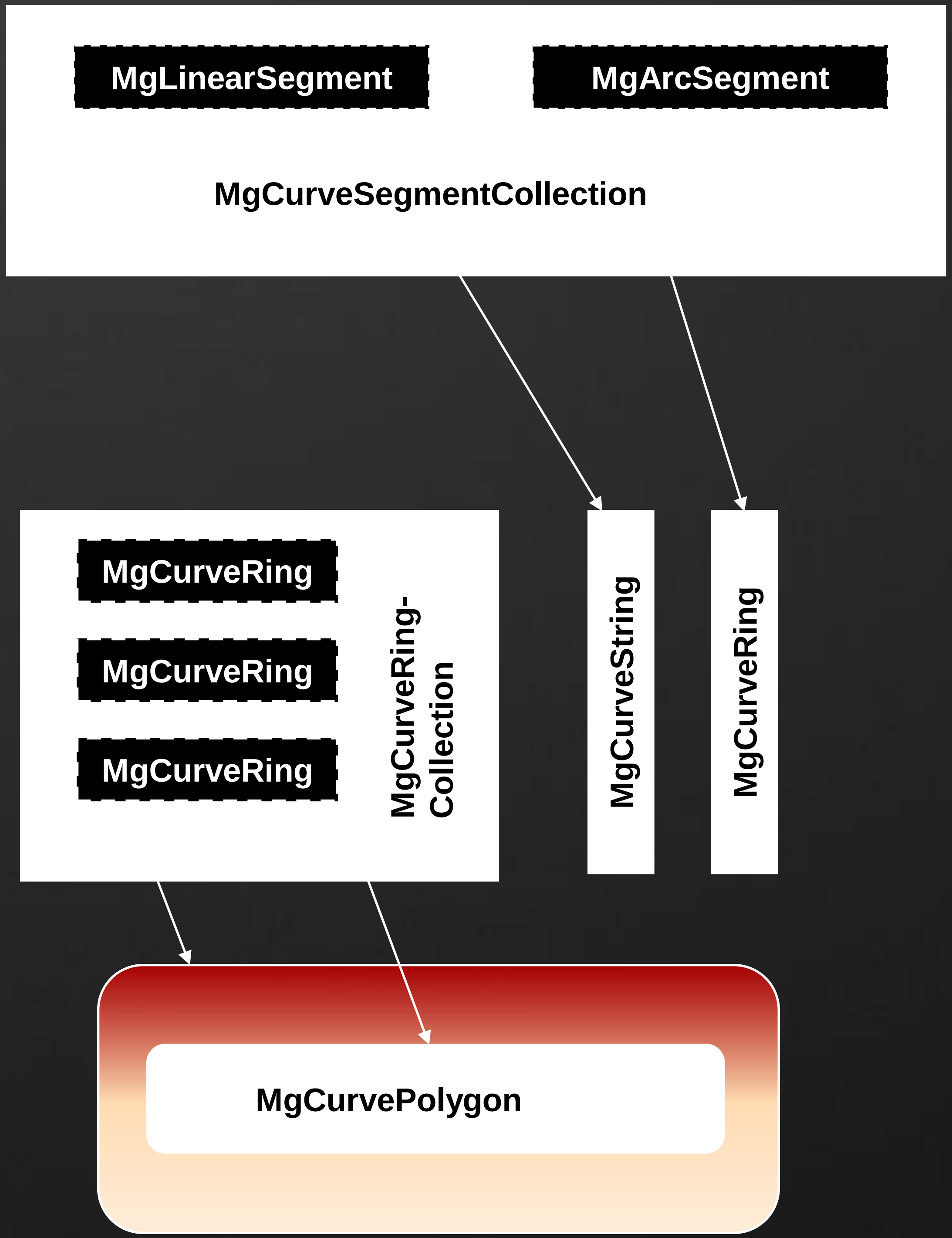
- Create a coordinate collection object
- Create coordinates representing the polygon vertices and add to the coordinates collection
- Create a LinearRing object from the coordinates collection
- Create polygon from LinearRing object

```
// Creating a polygon geometry

MgGeometryFactory geomFactory = new
MgGeometryFactory();

// create coordinate   and add to coordinates collection
MgCoordinateCollection coordCollection = new
MgCoordinateCollection();
MgCoordinate coord = geomFactory.CreateCoordinateXY(0,0);
coordCollection.Add(coord );
coord = geomFactory.CreateCoordinateXY(2,0);
coordCollection.Add(coord );
coord = geomFactory.CreateCoordinateXY(2,2);
coordCollection.Add(coord );
coord = geomFactory.CreateCoordinateXY(0,0);
coordCollection.Add(coord );

 //create LinearRing
 MgLinearRing ring= geomFactory.CreateLinearRing
(coordCollection );

// Create Polygon
MgPolygon polygon = geomFactory.CreatePolygon(ring);
```

Geometry Objects

Constructing Geometry Objects

- Use MgCurveSegmentCollection of MgLinearSegment objects and MgArcSegment
  to create MgCurveString geometries and MgCurveRing geometry components
- Use MgCurveRing to construct MgCurvePolygon object's external boundary, and an optional MgCurveRingCollection of MgCurveRing geometries to define "holes" in the containing ring



Geometry Objects

Constructing Geometry Objects

- Construct aggregate geometry objects (MgMultiXXX) by adding simple geometries to appropriate helper collection class object and passing that collection object to a constructor

```
// Creating a multiline string geometry
MgGeometryFactory geomFactory = new MgGeometryFactory();

// create coordinate   and add to coordinates collection
MgCoordinateCollection coordCollection = new MgCoordinateCollection();
MgCoordinate coord = geomFactory.CreateCoordinateXY(0,2); coordCollection.Add(coord );
coord = geomFactory.CreateCoordinateXY(2,2); coordCollection.Add(coord );

 //create LineString
 MgLineString lineString= geomFactory.CreateLineString (coordCollection );

MgLineStringCollection lineStringColl = new MgLineStringCollection();
 // After each MgLineString geometry is constructed, it is added to an MgLineStringCollection.
 int index =  lineStringColl.Add(lineString);

// construct the MgMultiLineString geometry
 MgMultiLineString multiLineString = geomFactory.CreateMultiLineString(lineStringColl);
```

Geometry Representation

- Geometry data formats
  - AGF text format represents geometry as a character string
    - “LINESTRING XY (0 0 , 1 -1)”
    - “POLYGON XY ((1 -3, 4 -3, 4 -6, 1 -6, 1 -3), (2 -4, 3 -4, 3 -5, 2 -5, 2 -4))”
    - “MULTIPOLYGON XY (((5 -3, 8 -3, 8 -6, 5 -6, 5 -3), (6 -4, 7 -4, 7 -5, 6 -5, 6 -4)), ((9 -3, 12 -3, 12 -6, 9 -6, 9 -3), (10 -4, 11 -4, 11 -5, 10 -5, 10 -4)))’’
  - Binary AGF format (in MgByteReader)
  - Internal representation, using MgGeometry or subclasses

Geometry Format Conversion

- Use MgAgfReaderWriter to translate between binary AGF and MgGeometry
- Use MgWktReaderWriter to translate between AGF Text and MgGeometry

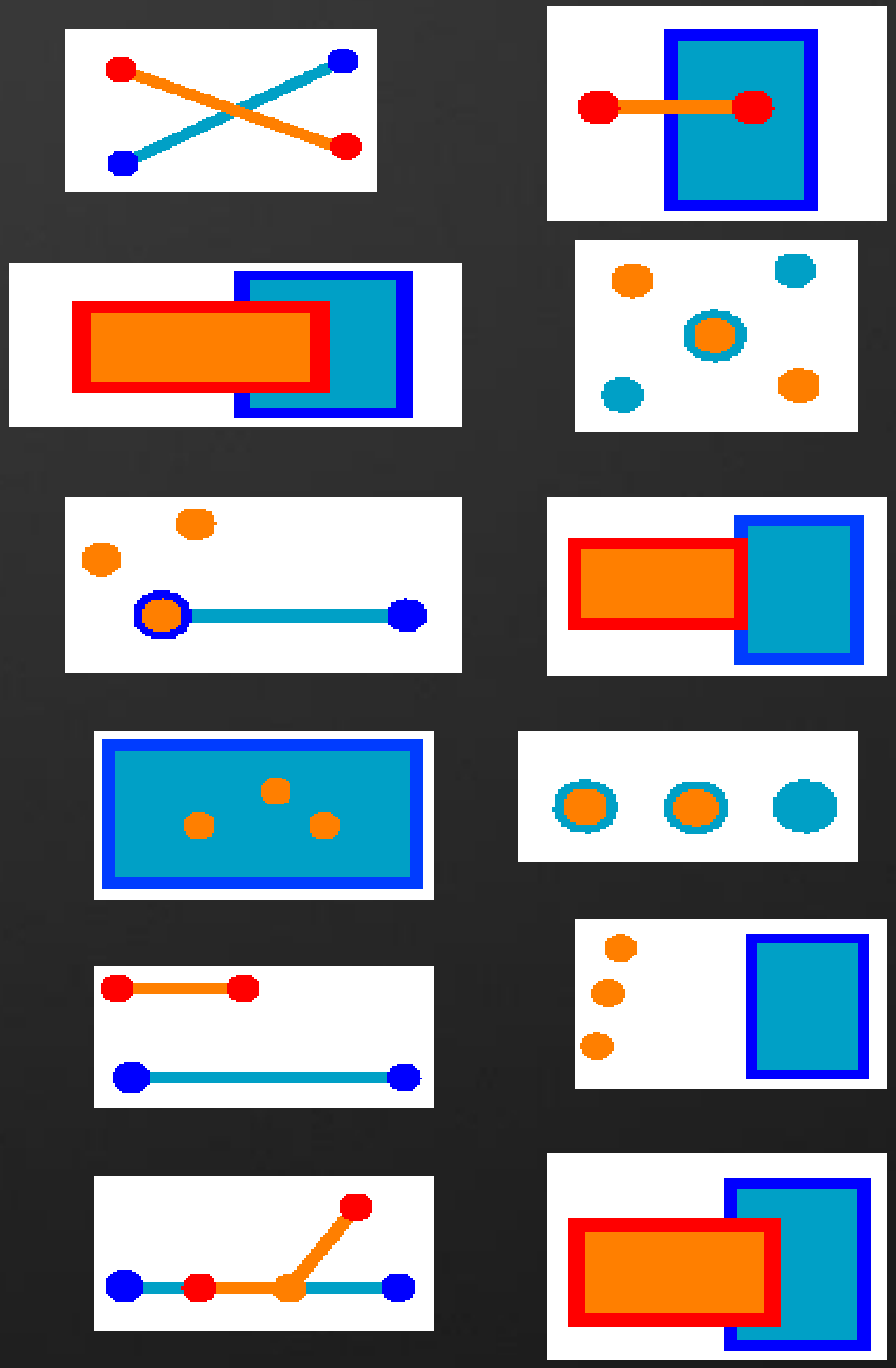
```
// Binary AGF -> MgGeometry
MgAgfReaderWriter  agfRW = new MgAgfReaderWriter();
// assume byteRdr, an MgByteReader object, contains a (binary) geometry
MgGeometry geom = agfRW.Read(byteRdr);

// MgGeometry -> Binary AGF
MgAgfReaderWriter  agfRW = new MgAgfReaderWriter();
// assume geom, an MgGeometry is already defined
MgByteReader byteRdr= agfRW.Write(geom);

// AGF Text -> MgGeometry
MgWktReaderWriter  wktRW = new MgWktReaderWriter();
String  wktStr = “POINT XY (1 -1)”;
MgGeometry geom = wktRW .Read(wktStr);

// MgGeometry -> AGF Text
MgWktReaderWriter  wktRW = new MgWktReaderWriter();
// assume geom, an MgGeometry is already defined
String  wktStr = wktRW.Write(geom);
```

Geometry Format Conversion

```
// Binary AGF -> MgGeometry
MgAgfReaderWriter  agfRW = new MgAgfReaderWriter();
// assume byteRdr, an MgByteReader object, contains a (binary) geometry
MgGeometry geom = agfRW.Read(byteRdr);

// MgGeometry -> Binary AGF
MgAgfReaderWriter  agfRW = new MgAgfReaderWriter();
// assume geom, an MgGeometry is already defined
MgByteReader byteRdr= agfRW.Write(geom);

// AGF Text -> MgGeometry
MgWktReaderWriter  wktRW = new MgWktReaderWriter();
String  wktStr = “POINT XY (1 -1)”;
MgGeometry geom = wktRW .Read(wktStr);

// MgGeometry -> AGF Text
MgWktReaderWriter  wktRW = new MgWktReaderWriter();
// assume geom, an MgGeometry is already defined
String  wktStr = wktRW.Write(geom);
```

Spatial Relationships between Objects

- Contains
- Crosses
- Disjoint
- Equals
- Intersects
- Overlaps
- Touches
- Within
- Inside



Crosses

Intersects

Touches

Within

Disjoint

Overlaps

Spatial Relationships between Objects

Predicates that create new geometries

- Union
- Intersection
- Difference
- Symmetric Difference

Coordinate Systems

Supported Coordinate Systems

- Arbitrary X-Y (MgCoordinateSystemType::Arbitrary)
- Geographic (MgCoordinateSystemType::Geographic)
- Projected (MgCoordinateSystemType::Projected)

Coordinate System Representation

- WKT string

```
PROJCS["SPAIN-TM30-I”,GEOGCS["",DATUM["",SPHEROID["INTNL",6378388.000,297.00000000]],PRIMEM["Greenwich",0],UNIT["Degree",0.017453292519943295]],PROJECTION["Transverse_Mercator"],PARAMETER["false_easting",500000.000],PARAMETER["false_northing",0.000],PARAMETER["scale_factor",0.999600000000],PARAMETER["central_meridian",-3.00000000000000],PARAMETER["latitude_of_origin",0.00000000000000],UNIT["Meter",1.00000000000000]]
```

- System Code – “SPAIN-TM30-I”

Coordinate Systems

Creation of Coordinate System

- Use MgCoordinateSystemFactory::Create()

```
//Creating a coordinate system object
 AcMapMap map = AcMapMap.GetCurrentMap();
 string csWkt = map.GetMapSRS();
 if (!string.IsNullOrEmpty(csWkt))
   {
      MgCoordinateSystemFactory coordSysFact = new MgCoordinateSystemFactory();
      MgCoordinateSystem coordSys = coordSysFact.Create(csWkt);
  }
```

Coordinates conversion functions

- MgCoordinateSystem:: ConvertCoordinateFromLonLat()
- MgCoordinateSystem:: ConvertCoordinateToLonLat()
- MgCoordinateSystem:: ConvertCoordinateSystemUnitsToMeters()
- MgCoordinateSystem:: ConvertMetersToCoordinateSystemUnits
- MgCoordinateSystem:: ConvertFromLonLat()
- MgCoordinateSystem:: ConvertToLonLat()

Creating Buffers

Creating Buffer Geometry

- Get the feature to create buffer around by using a selection filter
- Get the geometry of the feature
- Create buffer using MgGeometry::Buffer()
- Display buffer object

Displaying the Buffer object

- Create/Get feature source for the buffer, may require creating a new file
- Create a layer that references the feature source, add it to the map and make it visible.
- Create a new feature using the buffer geometry and insert it into the feature source

Autodesk®