Implementing OGC™ GeoPose 1.0

Table of Contents

The OGC GeoPose 1.0 Standard

A frequent question is "How should I implement software that makes or manipulates OGC GeoPoses?" This tutorial is one answer. It may not describe exactly the software you may need to build but I hope it gives you the necessary building blocks to see how you might proceed. This post features a TypeScript implementation, to be followed by posts on testing, C#, C++, Java, Python, Swift, and Kotlin.

OGC GeoPose 1.0 is a draft international standard for a family of JSON data objects representing the

position and orientation of real or virtual entities. There are eight independent forms described in the standard. There are no dependencies between the forms. Each may be implemented independently. There is no requirement to implement any specific form or number of forms.

This tutorial describes one template for developing a software implementation supporting GeoPose 1.0 using any one of many modern object-oriented programming languages. This post focuses on a TypeScript implementation. It is accompanied by a parallel implementation in C#, targeting open-source .NET 6.

This tutorial only touches briefly of the **use** of GeoPose - another important topic.

Use Cases

Here are some examples of use cases where GeoPose can play a role.

The GeoPose use cases involve interactions between information systems or between an information system and a storage medium. The essential role of a GeoPose is to convey the position and orientation of a real or virtual object. The possibility of chained transformational relationships and cross-linkages between chains affords representation of complex pose relationships and a way to bring a collection of related GeoPoses in a common geographic reference frame.

Augmented and Mixed Reality

Augmented Reality (AR) integrates synthetic objects or synthetic representations of real objects with a physical environment. Geospatial AR experiences can use GeoPose to position synthetic objects or their representations in the physical environment. The geospatial connection provides a common reference frame to support integration in AR.

- Stored representation of synthetic objects
- Positioning information to support integration of synthetic object data in a representation or visualization of the physical environment
- Report of position and orientation from a mobile device to an AR network service
- Input to visual occlusion calculations
- Input to ray-casting and line-of-sight calculations
- Input to proximity calculations
- Input and output to and from trajectory projection calculations

Mobile Autonomy

Autonomous vehicles are mobile objects that move through water, across a water surface, in the air, through the solid earth (tunnel boring machine), on the land surface, or in outer space without real-time control by an independent onboard operator. A pose captures the essential information in positioning and orienting a moving object. Sensors attached to mobile elements have their own poses and a chain of reference frame transformations enables common reference frames to be used for data fusion. The possibility of relating the vehicle to other elements of the environment via a

common reference frame is essential.

- Provide accurate visual positioning and guidance based on one or more services based on a 3D representation of the real world combined with real time detection and location of real world objects
- Calculate parameters such as distances and routes
- Record the trajectory of a moving vehicle.

Built Environment

The built environment consists of objects constructed by humans and located in physical space. Buildings, roads, dams, railways, and underground utilities are all part of the built environment. The location and orientation of built objects, especially those whose view is occluded by other objects is essential information needed for human interaction with the built environment. A common reference frame tied to the earth's surface facilitates the integration of these objects when their representations are supplied by different sources.

- Specify the position and orientation of visible objects and objects that are underground or hidden within a construction.
- Compactly and consistently specify or share the location and pose of objects in architecture, design and construction.

Synthetic Environments: Simulations, Replicas, and the Metaverse

Synthetic environments contain collections of moving objects, which themselves may be composed of connected and articulated parts, in an animation or simulation environment that contains a fixed background of air, land, water, vegetation, built objects, and other non-moving elements. The assembly is animated over some time period to provide visualizations or analytical results of the evolving state of the modelled environment. Synthetic environments support training, rehearsal, and archival of activities and events. The location and orientation of the movable elements of a scene are the key data controlling animation of in a synthetic environment. Since there may be multiple possible animations consistent with observations, storage of the sequences of poses of the actors, vehicles, and other objects is a direct and compact way of representing the variable aspects of the event. Access to one or more common reference frames through a graph of frame transformations makes a coherent assembly possible.

- Record pose relationships of all mobile elements in an environment
- Track targets from a moving platform
- Control animation of mobile elements in an environment using stored pose time sequences

Image Understanding

3D image understanding is the segmentation of an image or sequence of images into inferred 3D objects in specific semantic categories, possibly determining or constraining their motion and/or

geometry. One important application of image understanding is the recognition of moving elements in a time series of images. A pose is a compact representation of the key geometric characteristics of a moving element. In addition to moving elements sensed by an imaging device, it is often useful to know the pose of the sensor or imaging device itself. A common geographic reference frame integrates the objects into a single environment.

- Instantaneous and time series locations and orientations of mobile objects
- Instantaneous and time series location and orientation of an optical and/or depth imaging device using Simultaneous Location And Mapping (SLAM)
- Instantaneous and time series estimation of the changes in location and orientation of an object using an optical imaging device (Visual Odometry)
- Instantaneous and time series location and orientation of an optical imaging device used for photogrammetry

Development Goals

The OGC GeoPose 1.0 standard does not specify anything about software design or programming language. This goal of this tutorial is to walk through for design and implementation of software that works well with OGC GeoPose 1.0 and which can be integrated in to applications that create or receive GeoPose 1.0 data objects. The only requirement is that the language offer basic object-oriented programming support.

There are several specific goals:

Add GeoPose to applications

This is an obvious motivation for adding GeoPose software. Using a standard library makes it less difficult to start quickly and have a level of confidence that the operations are performed correctly.

External coordinate transformation libraries

GeoPose is based on an abstraction of transformations linking pairs of spaces or their associated reference frames. Many of the definitions of reference frames are complex and described in terms specific to a particular discipline, such as geodesy, surveying, or astrophysics. Experts in these disciplines have built specialized databases and transformation software and it is highly desirable to be able to use their work.

One very useful example is the PROJ coordinate transformation library eith used by itself or as part of the Geospatial Data Abstraction Library (GDAL) library. This tutorial uses an interface to PROJ to implement a range of more general transformations.

Experiment with possible new features

Having a working implementation of the standardized elements of GeoPose 1.0 makes it easy to experiment wih new features that might be proposed for a new version of the standard. I give two examples of how this can be done. First, I have provided three new properties for the Basic and

Advanced GeoPoses that have proved to be useful in my GeoPose applications. These additional properties serialize as additional JSON properties, which are explicitly allowed by the standard. Second, I have included the "Local" (Geo)Pose. Local is the closest to the usual concept of a pose in computer graphics. It is designed to allow chains and trees in the space of the rortated local tangent plane, east-north-up Cartesian coordinate system associated with the inner frame of Basic GeoPoses. The Local GeoPose can be expressed as an Advanced GeoPose but creating a simplified version with the frame transormation hardwired makes for clearer programming. I have not done so in this tutorial but it would be possible to configure the JSON serialization to output the Advanced equivalent, rather than a non-standard form.

Template Applies to a Wide range of object-oriented languages

The design only relies on a few basic O-O concepts and capabilities. These are supported by a wide range of old and new languages. In this post, I will only describe implementation in **TypeScript 4.9.5** and **C# 11 - .NET 6**. In future posts, I will continue with C++, Java, Swift, Kotlin, and Python.

Architecture

There are many possible implementations. My primary consideration is a simple and completely hierarchical design - patterned to meet the capibilities of common object-oriented languages. I also wanted to make it possible to consider individual parts in isolation and then to assemble them into a GeoPose tree.

I describe the parts in reverse order of dependency. By the time you get to the GeoPose, there are enough elements to start assembling them into the final structures.

Implementation in Eight Steps

The development steps outlined here proceed from independent components to three categories of GeoPoses: Basic, Advanced, and Local. Note that Local GeoPoses are within the scope of the GeoPose 1.0 logical model but must be serialized as Advanced GeoPoses to be compliant data objects.

- Step 1: Supporting Datatypes and Functions
- Step 2: Positions
- Step 3: Frame Transforms
- Step 4: Orientations
- Step 5: Abstract GeoPose
- Step 6: Basic GeoPoses
- Step 7: Advanced GeoPose
- Step 8: Local Pose

Implementing GeoPose in TypeScript

The following is the sequence of steps for a TypeScript implementation:

Step 1: Supporting Datatypes and Functions

Start here.

There are two simple datatypes that encapsulate an identifier and a time instant: PoseID and TimeValue. They are used in several of the classes. They are separated out because their design is dependent on the application domain and the need to interoperate with other systems. The GeoPose 1.0 standard does not specify any identifier and it defines a "valid Time" for only some of the GeoPose forms. Experience with the GeoPose since the initial publication shows the utility of references to GeoPoses and to having times associated with many individual GeoPoses.

Note that additional (private) properties may be added to most otherwise compliant GeoPose elements.

PoseID

```
PoseID has a single property - an id string.
```

UnixTime

```
UnixTime has a single property - a string representation of the number of Unix time
seconds multiplied by 1 000 for millisecond resolution.
```

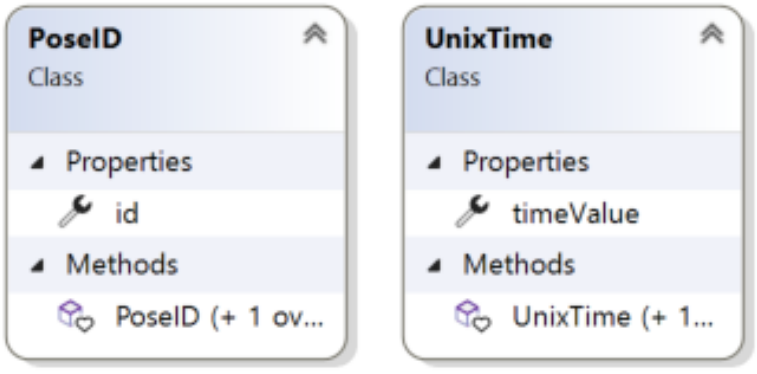

Figure 1. The PoseID and UnixTime Extras Classes

Coordinate conversion

The methodes of the LTP_ENU class are needed to support the Basic and Advanced classes' frame transformations. The GeoPose implementations must implement the actual transformations implied or designated by the class or outer and inner frame definitions. This in contrast to the GeoPose data objects, which carry no explicit information about how the transformations should be

carried out.

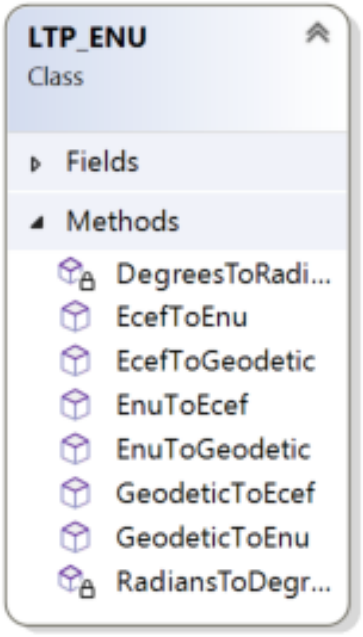

Figure 2. Calculation Support Classes

Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy.

TypeScript implemention:

Datatypes

```
// Implemention order: 1 - start here.
// These classes are non-structural elements.
// These are part of optional elements that are allowed but not standardized.

export class PoseID {
    public constructor(id: string) {
        this.id = id;
    }
    public id: string = "";
}

export class UnixTime {
    // Constructor from long integer count of UNIX Time seconds x 1000
    public constructor(longTime: number) {
        this.timeValue = longTime.toString();
    }
    public timeValue: string = "";
}
```

LTP_ENU coordinate conversion

```
import * as Position from './Position';

export class LTP_ENU {
    // WGS-84 geodetic constants
    readonly a: number = 6378137.0;         // WGS-84 Earth semimajor axis (m)
    readonly b: number = 6356752.314245;     // Derived Earth semiminor axis (m)
    readonly f: number = (this.a - this.b) / this.a;           // Ellipsoid Flatness
    readonly f_inv: number = 1.0 / this.f;       // Inverse flattening
    readonly a_sq: number = this.a * this.a;
    readonly b_sq: number = this.b * this.b;
    readonly e_sq: number = this.f * (2.0 - this.f);    // Square of Eccentricity
    readonly toRadians: number = Math.PI / 180.0;
    readonly toDegrees: number = 180.0 / Math.PI;

    // Convert WGS-84 Geodetic point (lat, lon, h) to the
    // Earth-Centered Earth-Fixed (ECEF) coordinates (x, y, z).
    public GeodeticToEcef(from: Position.GeodeticPosition, to:
Position.CartesianPosition): void {
        // Convert to radians in notation consistent with the paper:
        var lambda = from.lat * this.toRadians;
        var phi = from.lon * this.toDegrees;
        var s = Math.sin(lambda);
        var N = this.a / Math.sqrt(1.0 - this.e_sq * s * s);

        var sin_lambda = Math.sin(lambda);
        var cos_lambda = Math.cos(lambda);
        var cos_phi = Math.cos(phi);
        var sin_phi = Math.sin(phi);
```

```
        to.x = (from.h + N) * cos_lambda * cos_phi;
        to.y = (from.h + N) * cos_lambda * sin_phi;
        to.z = (from.h + (1 - this.e_sq) * N) * sin_lambda;
    }

    // Convert the Earth-Centered Earth-Fixed (ECEF) coordinates (x, y, z) to
    // (WGS-84) Geodetic point (lat, lon, h).
    public EcefToGeodetic(from: Position.CartesianPosition, to:
Position.GeodeticPosition): void {
        var eps = this.e_sq / (1.0 - this.e_sq);
        var p = Math.sqrt(from.x * from.x + from.y * from.y);
        var q = Math.atan2((from.z * this.a), (p * this.b));
        var sin_q = Math.sin(q);
        var cos_q = Math.cos(q);
        var sin_q_3 = sin_q * sin_q * sin_q;
        var cos_q_3 = cos_q * cos_q * cos_q;
        var phi = Math.atan2((from.z + eps * this.b * sin_q_3), (p - this.e_sq *
this.a * cos_q_3));
        var lambda = Math.atan2(from.y, from.x);
        var v = this.a / Math.sqrt(1.0 - this.e_sq * Math.sin(phi) * Math.sin(phi));
        to.h = (p / Math.cos(phi)) - v;

        to.lat = phi * this.toDegrees;
        to.lon = lambda * this.toDegrees;
    }

    // Converts the Earth-Centered Earth-Fixed (ECEF) coordinates (x, y, z) to
    // East-North-Up coordinates in a Local Tangent Plane that is centered at the
    // (WGS-84) Geodetic point (lat0, lon0, h0).
    public EcefToEnu(from: Position.CartesianPosition, origin:
Position.GeodeticPosition, to: Position.CartesianPosition):
        //double x, double y, double z,
        //double lat0, double lon0, double h0,
        //out double xEast, out double yNorth, out double zUp):
        void {
        // Convert to radians in notation consistent with the paper:
        var lambda = origin.lat * this.toRadians;
        var phi = origin.lon * this.toDegrees;
        var s = Math.sin(lambda);
        var N = this.a / Math.sqrt(1.0 - this.e_sq * s * s);

        var sin_lambda = Math.sin(lambda);
        var cos_lambda = Math.cos(lambda);
        var cos_phi = Math.cos(phi);
        var sin_phi = Math.sin(phi);

        var x0: number = (origin.h + N) * cos_lambda * cos_phi;
        var y0: number = (origin.h + N) * cos_lambda * sin_phi;
        var z0: number = (origin.h + (1 - this.e_sq) * N) * sin_lambda;
```

```
        var xd: number = from.x - x0;
        var yd: number = from.y - y0;
        var zd: number = from.z - z0;

        // This is the matrix multiplication
        to.x = -sin_phi * xd + cos_phi * yd;
        to.y = -cos_phi * sin_lambda * xd - sin_lambda * sin_phi * yd + cos_lambda *
zd;
        to.z = cos_lambda * cos_phi * xd + cos_lambda * sin_phi * yd + sin_lambda *
zd;
    }

    // Inverse of EcefToEnu. Converts East-North-Up coordinates (xEast, yNorth, zUp)
in a
    // Local Tangent Plane that is centered at the (WGS-84) Geodetic point (lat0,
lon0, h0)
    // to the Earth-Centered Earth-Fixed (ECEF) coordinates (x, y, z).
    public EnuToEcef(from: Position.CartesianPosition, origin:
Position.GeodeticPosition, to: Position.CartesianPosition): void {
        // Convert to radians in notation consistent with the paper:
        var lambda = origin.lat * this.toRadians;
        var phi = origin.lon * this.toRadians;
        var s = Math.sin(lambda);
        var N = this.a / Math.sqrt(1.0 - this.e_sq * s * s);

        var sin_lambda = Math.sin(lambda);
        var cos_lambda = Math.cos(lambda);
        var cos_phi = Math.cos(phi);
        var sin_phi = Math.sin(phi);

        var x0: number = (origin.h + N) * cos_lambda * cos_phi;
        var y0: number = (origin.h + N) * cos_lambda * sin_phi;
        var z0: number = (origin.h + (1.0 - this.e_sq) * N) * sin_lambda;

        var xd: number = -sin_phi * from.x - cos_phi * sin_lambda * from.y +
cos_lambda * cos_phi * from.z;
        var yd: number = cos_phi * from.x - sin_lambda * sin_phi * from.y + cos_lambda
* sin_phi * from.z;
        var zd: number = cos_lambda * from.y + sin_lambda * from.z;

        to.x = xd + x0;
        to.y = yd + y0;
        to.z = zd + z0;
    }

    // Converts the geodetic WGS-84 coordinated (lat, lon, h) to
    // East-North-Up coordinates in a Local Tangent Plane that is centered at the
    // (WGS-84) Geodetic point (lat0, lon0, h0).
    public GeodeticToEnu(from: Position.GeodeticPosition, origin:
Position.GeodeticPosition, to: Position.CartesianPosition): void
    //double lat0, double lon0, double h0,
```

```
    //out double xEast, out double yNorth, out double zUp)
    {
        let ecef = new Position.CartesianPosition(0, 0, 0);
        this.GeodeticToEcef(from, ecef);
        this.EcefToEnu(ecef, origin, to);
    }
    public EnuToGeodetic(from: Position.CartesianPosition, origin:
Position.GeodeticPosition, to: Position.GeodeticPosition): void
    //double xEast, double yNorth, double zUp,
    //double lat0, double lon0, double h0,
    //out double lat, out double lon, out double h
    {
        let ecef = new Position.CartesianPosition(0, 0, 0);
        this.EnuToEcef(from, origin, ecef);
        this.EcefToGeodetic(ecef, to);
    }
}
```

Step 2: Positions

The Position class and its derivatives represent different styles of using three coordinate values to designate a position in a three-dimensional space.

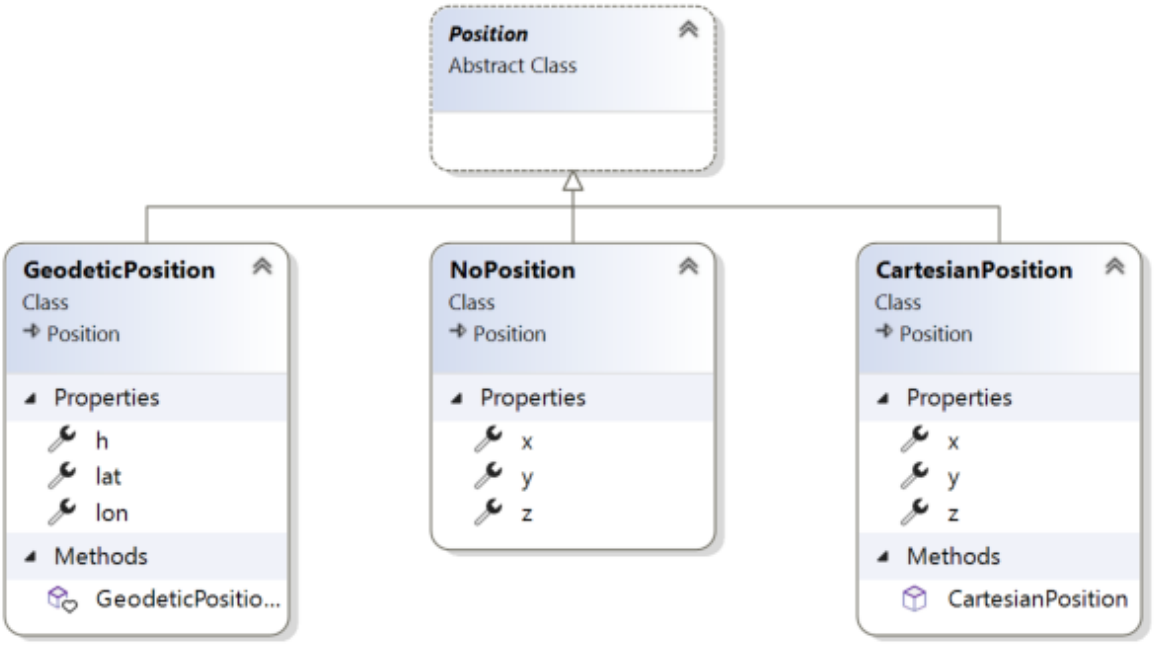

Figure 3. Positions

GeoPose 1.0 supports both a geodetic form and a Cartesian form. These forms are used in both frame transformations and orientation (rotation) transformations, both as quantities to be transformed and, in some cases, as a parameter of a family of transformations. Since some transformations are not possible, due to a mathematical singularity, unavailability of a transformation, or a runtime error in the tranformation calculation, the NoPosition position is used as a "null" value. Each of the coordinates of the NoPosition are the IEEE 754 value NaN.

TypeScript implemention:

```
// Implemention order: 2 - follows Extras.
// These classes define positions in a 3D frame using different conventions.

/// <summary>
/// The abstract root of the Position hierarchy.
/// <note>
/// Because these various ways to express Position share no underlying structure,
/// the abstract root class definition is simply an empty shell.
/// </note>
/// </summary>
export abstract class Position {
}

/// <summary>
/// GeodeticPosition is a specialization of Position for using two angles and a height
for geodetic reference systems.
/// </summary>
export class GeodeticPosition extends Position {
    public constructor(lat: number, lon: number, h: number) {
        super();
        this.lat = lat;
        this.lon = lon;
        this.h = h;
    }

    /// <summary>
    /// A latitude in degrees, positive north of equator and negative south of
equator.
    /// The latitude is the angle between the plane of the equator and a plane tangent
to the ellipsoid at the given point.
    /// </summary>
    public lat: number;
    /// <summary>
    /// A longitude in degrees, positive east of the prime meridian and negative west
of prime meridian.
    /// </summary>
    public lon: number;
    /// <summary>
    /// A distance in meters, measured with respect to an implied (Basic) or specified
(Advanced) reference surface,
    /// postive opposite the direction of the force of gravity,
    /// and negative in the direction of the force of gravity.
    /// </summary>
    public h: number
}
/// <summary>
/// CartesianPosition is a specialization of Position for geocentric, topocentric, and
engineering reference systems.
```

```
/// </summary>
export class CartesianPosition extends Position {
    public constructor(x: number, y: number, z: number) {
        super();
        this.x = x;
        this.y = y;
        this.z = z;
    }

    /// <summary>
    /// A coordinate value in meters, along an axis (x-axis) that typically has origin
at
    /// the center of mass, lies in the same plane as the y axis, and perpendicular to
the y axis,
    /// forming a right-hand coordinate system with the z-axis in the up direction.
    /// </summary>
    public x: number;
    /// <summary>
    /// A coordinate value in meters, along an axis (y-axis) that typically has origin
at
    /// the center of mass, lies in the same plane as the x axis, and perpendicular to
the x axis,
    /// forming a right-hand coordinate system with the z-axis in the up direction.
    /// </summary>
    public y: number;
    /// <summary>
    /// A coordinate value in meters, along the z-axis.
    /// </summary>
    public z: number;
}

export class NoPosition extends Position {
    public constructor() {
        super();
        this.x = this.y = this.z = NaN;
    }
    /// <summary>
    /// A coordinate value in meters, along an axis (x-axis) that typically has origin
at
    /// the center of mass, lies in the same plane as the y axis, and perpendicular to
the y axis,
    /// forming a right-hand coordinate system with the z-axis in the up direction.
    /// </summary>
    public x: number;
    /// <summary>
    /// A coordinate value in meters, along an axis (y-axis) that typically has origin
at
    /// the center of mass, lies in the same plane as the x axis, and perpendicular to
the x axis,
    /// forming a right-hand coordinate system with the z-axis in the up direction.
    /// </summary>
```

```
    public y: number;
    /// <summary>
    /// A coordinate value in meters, along the z-axis.
    /// </summary>
    public z: number;
}
```

Step 3: Frame Transforms

Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy.

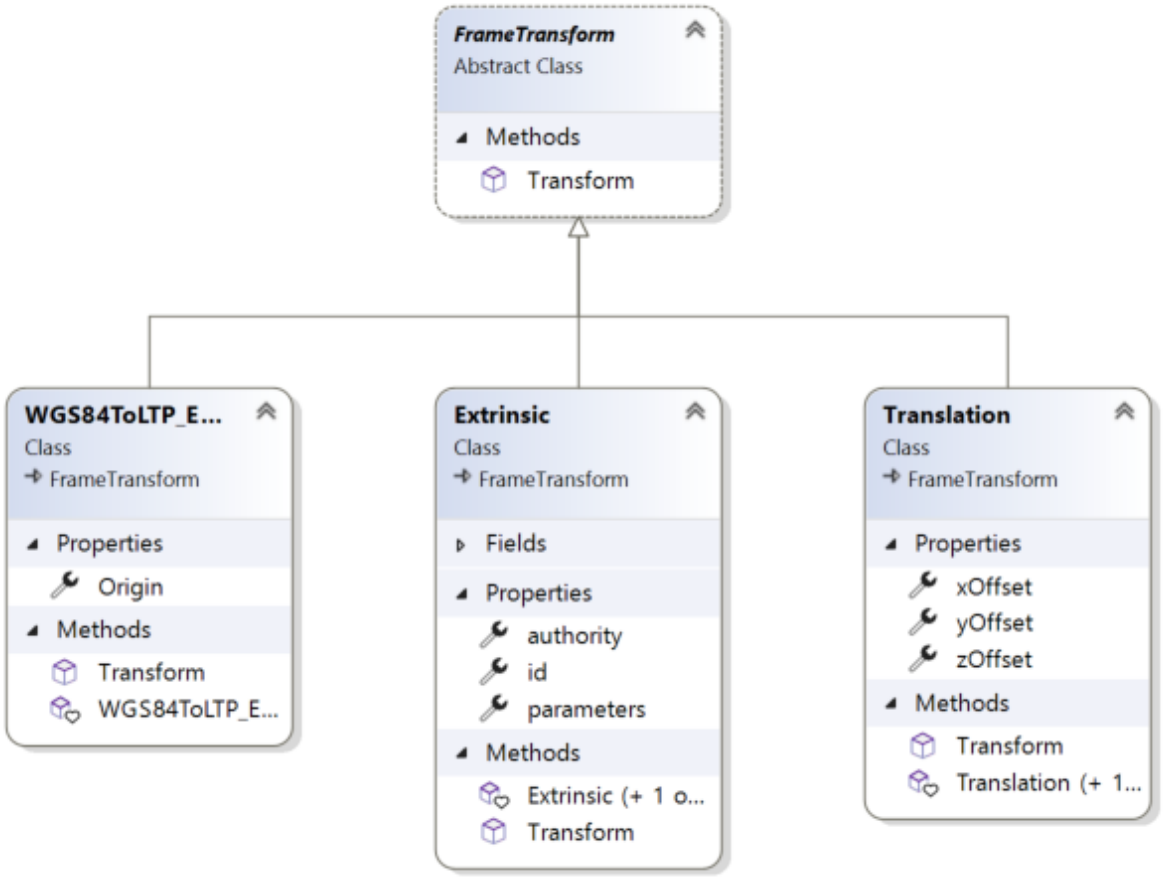

Figure 4. Frame Transform

Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy.

TypeScript implemention:

```
import * as proj4 from 'proj4';
import * as Position from './Position';

// Implemention order: 3 - follows Position.
// These classes define transformations of a Position in one 3D frame to a Position in
another 3D frame.

/// <summary>
/// A FrameTransform is a generic container for information that defines mapping
between reference frames.
/// Most transformation have a context with necessary ancillary information
/// that parameterizes the transformation of a Position in one frame to a
corresponding Position is another.
/// Such context may include, for example, some or all of the information that may be
conveyed in an ISO 19111 CRS specification
/// or a proprietary naming, numbering, or modelling scheme as used by EPSG, NASA
Spice, or SEDRIS SRM.
/// Subclasses of FrameTransform exist precisely to hold this context in conjunction
with code
/// implementing a Transform function.
/// <remark>
/// </remark>
/// </summary>
export abstract class FrameTransform {
    public abstract Transform(point: Position.Position): Position.Position;
}

/// <summary>
/// A FrameSpecification is a generic container for information that defines a
reference frame.
/// <remark>
/// A FrameSpecification can be abstracted as a Position:
/// The origin of the coordinate system associated with the frame is a Position and
serves in that role
/// in the Advanced GeoPose.
/// The origin, is in fact the *only* distinguished Position associated with the
coodinate system.
/// </remark>
/// </summary>
export class Extrinsic extends FrameTransform {
    public constructor(authority: string, id: string, parameters: string) {
        super();
        this.authority = authority;
        this.id = id;
        this.parameters = parameters;
    }
    /// <summary>
    /// The core function of a transformation is to implement a specific frame
```

```
transformation
    /// i.e. the transformation of a triple of point coordinates in the outer frame to
a triple of point coordinates in the inner frame.
    /// When this is not possible due to lack of an appropriate tranformation
procedure,
    /// the triple (NaN, NaN, NaN) [three IEEE 574 not-a-number vales] is returned.
    /// Note that an "authority" is not necessarily a standards organization but
rather an entity that provides
    /// a register of some kind for a category of frame- and/or frame transform
specifications that is useful and stable enough
    /// for someone to implement transformation functions.
    /// An implementation need not implement all possbile transforms.
    /// </summary>
    /// <note>
    /// This would be a good element to implement as a set of plugin.
    /// </note>
    /// <param name="point"></param>
    /// <returns></returns>
    public override Transform(point: Position.Position): Position.Position {
        let uri = this.authority.toLowerCase().replace("//www.", "");
        if (uri == "https://proj.org" || uri == "https://osgeo.org") {
            var outer = proj4.Proj('EPSG:4326');    //source coordinates will be in
Longitude/Latitude, WGS84
            var inner = proj4.Proj('EPSG:3785');     //destination coordinates in
meters, global spherical mercato
            var cp = point as Position.CartesianPosition;
            let p = proj4.Point(cp.x, cp.y, cp.z);
            proj4.transform(outer, inner, p);
            // convert points from one coordinate system to another
            let outP = new Position.CartesianPosition(p.x, p.y, p.z);
            return outP;
        }
        else if (uri == "https://epsg.org") {
            return Position.NoPosition;
        }
        else if (uri == "https://iers.org") {
            return Position.NoPosition;
        }
        else if (uri == "https://naif.jpl.nasa.gov") {
            return Position.NoPosition;
        }
        else if (uri == "https://sedris.org") {
            return Position.NoPosition;
        }
        else if (uri == "https://iau.org") {
            return Position.NoPosition;
        }
        return Position.NoPosition;
    }
    /// <summary>
    /// The name or identification of the definer of the category of frame
```

```
specification.
    /// A Uri that usually but not always points to a valid web address.
    /// </summary>
    public authority: string;
    /// <summary>
    /// A string that uniquely identifies a frame type.
    /// The interpretation of the string is determined by the authority.
    /// </summary>
    public id: string;
    /// <summary>
    /// A string that holds any parameters required by the authority to define a frame
of the given type as specified by the id.
    /// The interpretation of the string is determined by the authority.
    /// </summary>
    public parameters: string;
    public static noTransform: Position.Position = new Position.NoPosition();
}
/// <summary>
/// A specialized specification of the WGS84 (EPSG 4326) geodetic frame to a local
tangent plane East, North, Up frame.
/// <remark>
/// The origin of the coordinate system associated with the frame is a Position - the
origin -
/// which is the *only* distinguished Position associated with the coodinate system
associated with the inner frame (range).
/// </remark>
/// </summary>
export class WGS84ToLTPENU extends FrameTransform {
    public constructor(origin: Position.GeodeticPosition) {
        super();
        this.Origin = origin;
    }
    public override Transform(point: Position.Position): Position.Position {
        let geoPoint = point as Position.GeodeticPosition;
        let outPoint: Position.CartesianPosition;
        GeodeticToEnu(this.Origin, geoPoint, outPoint);
        return outPoint;
    }

    /// <summary>
    /// A single geodetic position defines the tangent point for a transform to LTP-
ENU.
    /// </summary>
    public Origin: Position.GeodeticPosition;
}

export function GeodeticToEnu(origin: Position.GeodeticPosition, geoPoint:
Position.GeodeticPosition, enuPoint: Position.CartesianPosition) {
    let out = new Position.CartesianPosition(0, 0, 0);
    return out;
}
```

```
// A simple translation frame transform.
// The FrameTransform is created with an offset.
// The Transform adds the offset ot an input Cartesian Position and reurns a Cartesian
Position
export class Translation extends FrameTransform {
    public constructor(xOffset: number, yOffset: number, zOffset: number) {
        super();
        this.xOffset = xOffset;
        this.yOffset = yOffset;
        this.zOffset = zOffset;
    }
    public override Transform(point: Position.Position): Position.Position {
        let cp = point as Position.CartesianPosition;
        let p = new Position.CartesianPosition(cp.x + this.xOffset, cp.y +
this.yOffset, cp.z + this.zOffset);
        return p;
    }
    public xOffset: number;
    public yOffset: number;
    public zOffset: number;
}
```

Step 4: Orientations

Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy.

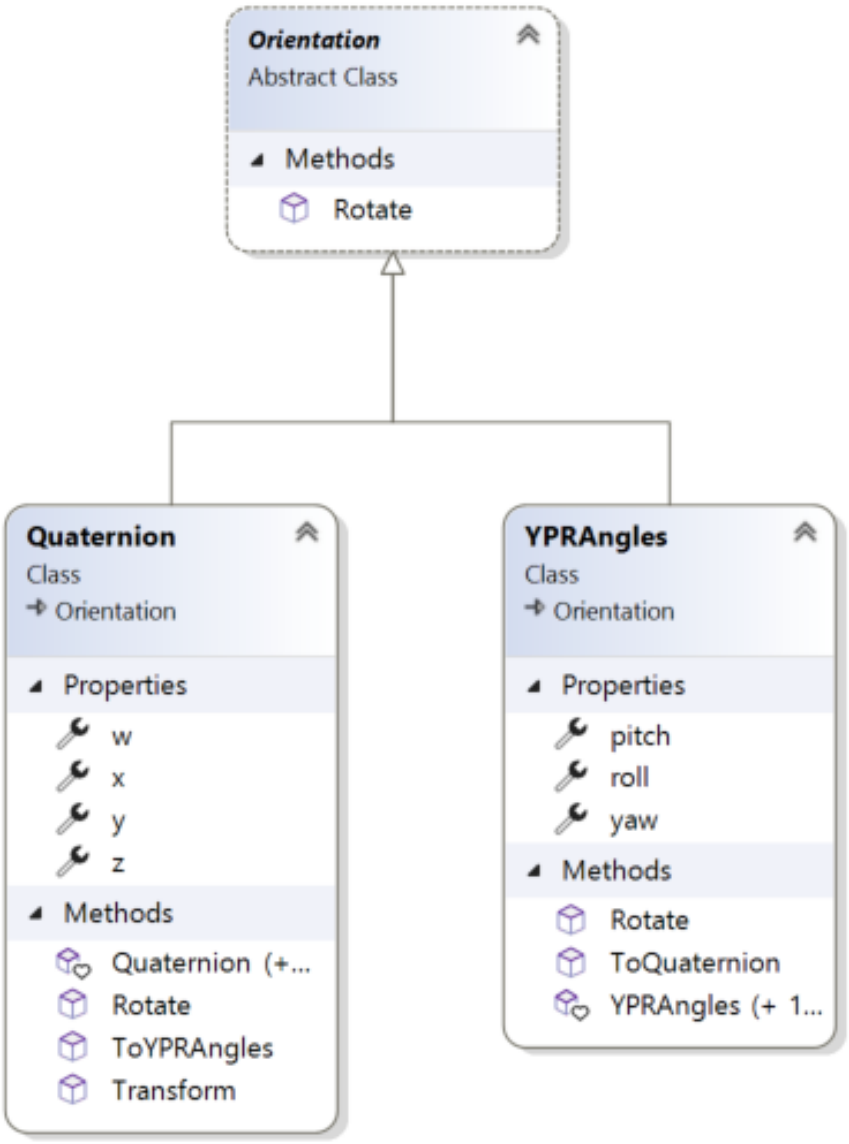

Figure 5. Orientations

Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy.

Yaw, Pitch, Roll

Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy.

Unit Quaternions

Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy.

TypeScript implemention:

```
import * as Position from './Position';

// Implemention order: 4 - follows FrameTransform.
// These classes define rotations of a 3D frame transforming a Position to a rotated
Position.


/// <summary>
/// The abstract root of the Orientation hierarchy.
/// <note>
/// An Orientation is a generic container for information that defines rotation within
a coordinate system associated with a reference frame.
/// An Orientation may have a specialized context with necessary ancillary information
/// that parameterizes the rotation.
/// Such context may include, for example, part of the information that may be
conveyed in an ISO 19111 CRS specification
/// or a proprietary naming, numbering, or modelling scheme as used by EPSG, NASA
Spice, or SEDRIS SRM.
/// Subclasses of Orientation exist precisely to hold this context in conjunction with
code
/// implementing a Rotate function.
/// </note>
/// </summary>
export abstract class Orientation {
    abstract Rotate(point: Position.CartesianPosition): Position.Position;
}

/// <summary>
/// A specialization of Orientation using Yaw, Pitch, and Roll angles measured in
degrees.
/// <remark>
/// This style of Orientation is best for easy human interpretation.
/// It suffers from some computational inefficiencies, awkward interpolation, and
singularities.
/// </remark>
/// </summary>
export class YPRAngles extends Orientation {
    public constructor(yaw: number, pitch: number, roll: number) {
```

```
        super();
        this.yaw = yaw;
        this.pitch = pitch;
        this.roll = roll;
    }

    /// <summary>
    /// The function is to apply a YPR transformation
    /// </summary>
    public override Rotate(point: Position.CartesianPosition): Position.Position {
        // convert to quaternion and use quaternion rotation
        let q = YPRAngles.ToQuaternion(this.yaw, this.pitch, this.roll);
        return Quaternion.Transform(point, q);
    }
    public static ToQuaternion(yaw: number, pitch: number, roll: number): Quaternion {
        // GeoPose angles are measured in degrees for human readability
        // Convert degrees to radians.
        yaw *= (Math.PI / 180.0);
        pitch *= (Math.PI / 180.0);
        roll *= (Math.PI / 180.0);

        let cosRoll = Math.cos(roll * 0.5);
        let sinRoll = Math.sin(roll * 0.5);
        let cosPitch = Math.cos(pitch * 0.5);
        let sinPitch = Math.sin(pitch * 0.5);
        let cosYaw = Math.cos(yaw * 0.5);
        let sinYaw = Math.sin(yaw * 0.5);

        let w = cosRoll * cosPitch * cosYaw + sinRoll * sinPitch * sinYaw;
        let x = sinRoll * cosPitch * cosYaw - cosRoll * sinPitch * sinYaw;
        let y = cosRoll * sinPitch * cosYaw + sinRoll * cosPitch * sinYaw;
        let z = cosRoll * cosPitch * sinYaw - sinRoll * sinPitch * cosYaw;

        let norm = Math.sqrt(x * x + y * y + z * z + w * w);
        let q = new Quaternion(x, y, z, w);
        if (norm > 0.0) {
            q.x = q.x / norm;
            q.y = q.y / norm;
            q.z = q.z / norm;
            q.w = q.w / norm;
        }
        return q;
    }
    /// <summary>
    /// A left-right angle in degrees.
    /// </summary>
    public yaw: number;
    /// <summary>
    /// A forward-looking up-down angle in degrees.
    /// </summary>
    public pitch: number;
```

```
    /// <summary>
    /// A side-to-side angle in degrees.
    /// </summary>
    public roll: number;
}
/// <summary>
/// Quaternion is a specialization of Orientation using a unit quaternion.
/// </summary>
/// <remark>
/// This style of Orientation is best for computation.
/// It is not easily interpreted or visualized by humans.
/// </remark>
export class Quaternion extends Orientation {
    public constructor(x: number, y: number, z: number, w: number) {
        super();
        this.x = x;
        this.y = y;
        this.z = z;
        this.w = w;
    }
    public override Rotate(point: Position.CartesianPosition): Position.Position {
        return Quaternion.Transform(point, this);
    }
    public ToYPRAngles(q: Quaternion): YPRAngles {

        // roll (x-axis rotation)
        let sinRollCosPitch = 2.0 * (q.w * q.x + q.y * q.z);
        let cosRollCosPitch = 1.0 - 2.0 * (q.x * q.x + q.y * q.y);
        let roll = Math.atan2(sinRollCosPitch, cosRollCosPitch) * (180.0 / Math.PI);
// in degrees

        // pitch (y-axis rotation)
        let sinPitch = Math.sqrt(1.0 + 2.0 * (q.w * q.y - q.x * q.z));
        let cosPitch = Math.sqrt(1.0 - 2.0 * (q.w * q.y - q.x * q.z));
        let pitch = (2.0 * Math.atan2(sinPitch, cosPitch) - Math.PI / 2.0) * (180.0 /
Math.PI); // in degrees

        // yaw (z-axis rotation)
        let sinYawCosPitch = 2.0 * (q.w * q.z + q.x * q.y);
        let cosYawCosPitch = 1.0 - 2.0 * (q.y * q.y + q.z * q.z);
        let yaw = Math.atan2(sinYawCosPitch, cosYawCosPitch) * (180.0 / Math.PI); //
in degrees
        let yprAngles = new YPRAngles(yaw, pitch, roll);
        return yprAngles;
    }
    public static Transform(inPoint: Position.CartesianPosition, rotation:
Quaternion): Position.CartesianPosition {
        let point = new Position.CartesianPosition(inPoint.x, inPoint.y, inPoint.z);
        let x2 = rotation.x + rotation.x;
        let y2 = rotation.y + rotation.y;
        let z2 = rotation.z + rotation.z;
```

```
        let wx2 = rotation.w * x2;
        let wy2 = rotation.w * y2;
        let wz2 = rotation.w * z2;
        let xx2 = rotation.x * x2;
        let xy2 = rotation.x * y2;
        let xz2 = rotation.x * z2;
        let yy2 = rotation.y * y2;
        let yz2 = rotation.y * z2;
        let zz2 = rotation.z * z2;

        let p = new Position.CartesianPosition(
            point.x * (1.0 - yy2 - zz2) + point.y * (xy2 - wz2) + point.z * (xz2 +
wy2),
            point.x * (xy2 + wz2) + point.y * (1.0 - xx2 - zz2) + point.z * (yz2 -
wx2),
            point.x * (xz2 - wy2) + point.y * (yz2 + wx2) + point.z * (1.0 - xx2 -
yy2));
        return p;
    }
    /// <summary>
    /// The x component.
    /// </summary>
    public x: number;
    /// <summary>
    /// The y component.
    /// </summary>
    public y: number;
    /// <summary>
    /// The z component.
    /// </summary>
    public z: number;
    /// <summary>
    /// The w component.
    /// </summary>
    public w: number;
}
```

Step 5: Abstract GeoPose

Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy.

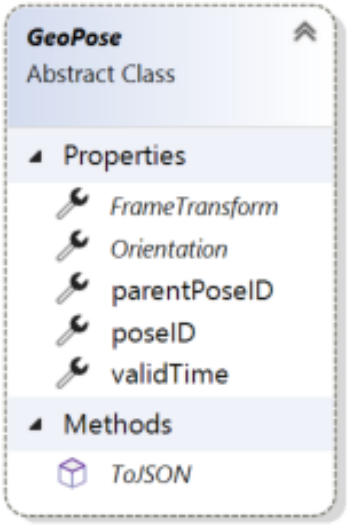

Figure 6. GeoPose

Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy.

TypeScript implemention:

```
import * as Extras from './Extras';
import * as FrameTransform from './FrameTransform';
import * as Orientation from './Orientation';

// Implemention order: 5 - follows Orientation.
// This is the root of the GeoPose inheritance hierarchy.

/// <summary>
/// A GeoPose has a position and an orientation.
/// The position is abstracted as a transformation between one reference frame (outer
frame)
/// and another (inner frame).
/// The position is the origin of the coordinate system of the inner frame.
/// The orientation is applied to the coordinate system of the inner frame.
/// <remark>
/// See the OGS GeoPose 1.0 standard for a full description.
/// </remark>
/// <remark>
/// This implementation includes some optional properties not define in the 1.0
standard
/// but allowed by JSON serializations of all but the Basic-Quaternion(Strict)
standardization target.
/// The optional properties are identifiers and time values that are useful in
practice.
/// They may be part of a future version of the standard but, as of February 2023,
they are optianl add-ons.
/// </remark>
/// </summary>
export abstract class GeoPose {
    // Optional and non-standard but conforming added property:
    // an identifier unique within an application.
    public poseID: Extras.PoseID;

    // Optional and non-standard but conforming added property:
    // a PoseID type identifier of another GeoPose in the direction of the root of a
pose tree.
    public parentPoseID: Extras.PoseID;

    // Optional and non-standard (except in Advanced) but conforming added property:
    // a validTime with milliseconds of Unix time.
    public validTime: number;
    abstract FrameTransform: FrameTransform.FrameTransform;
    abstract Orientation: Orientation.Orientation;
}
```

Step 6: Basic GeoPoses

Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o

text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy.

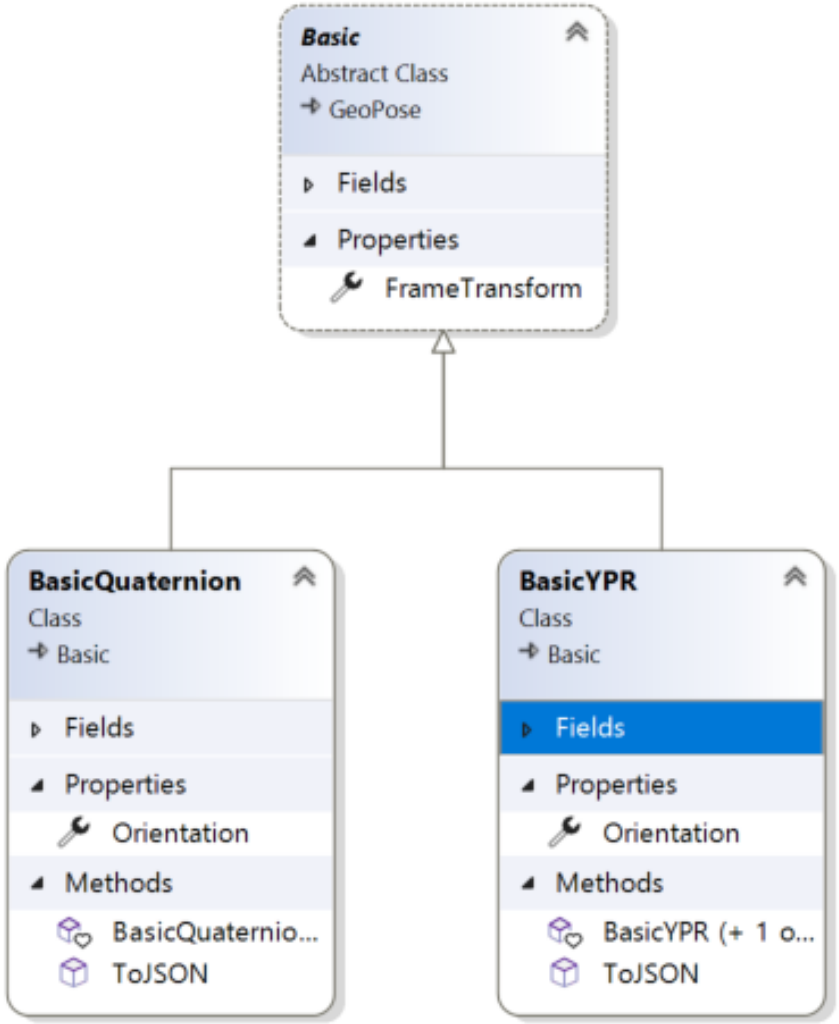

Figure 7. Basic

Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy.

YPR

Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile

o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy.

Quaternion

Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy.

TypeScript implemention:

```
// Implemention order: 6 - follows GeoPose.
// This is the simplest family of GeoPoses - the 80% part of a 80/20 solution.

/// <summary>
/// The Basic GeoPoses share the use of a local tangent plane, east-north-up frame
transform.
/// The types of Basic GeoPose are distinguished by the method used to specify
orientation of the inner frame.
/// </summary>
export abstract class Basic extends GeoPose.GeoPose {
    /// <summary>
    /// A Position specified in geographic coordinates with height above a reference
surface -
    /// usually an ellipsoid of revolution or a gravitational equipotential surface is
    /// transformed to a local Cartesian frame, suitable for use over an extent of a
few km.
    /// </summary>
    public override FrameTransform: FrameTransform.WGS84ToLTPENU;
}

/// <summary>
/// A Basic-YPR GeoPose uses yaw, pitch, and roll angles measured in degrees to define
the orientation of the inner frame..
/// </summary>
export class BasicYPR extends Basic {
    public constructor(id: string, tangentPoint: Position.GeodeticPosition, yprAngles:
Orientation.YPRAngles) {
        super();
        this.poseID = new Extras.PoseID(id);
        this.FrameTransform = new FrameTransform.WGS84ToLTPENU(tangentPoint);
        this.Orientation = yprAngles;
    }
    /// <summary>
```

```
    /// An Orientation specified as three successive rotations about the local Z, Y,
and X axes, in that order..
    /// </summary>
    public override Orientation: Orientation.YPRAngles;

    /// <summary>
    /// This function returns a Json encoding of a Basic-YPR GeoPose
    /// </summary>
    public toJSON(): string {
        let indent: string = "";
        let sb:string[] = [''];
        if (FrameTransform != null && Orientation != null) {
            sb.push("{\r\n  " + indent);
            if (this.validTime != null ) {
                sb.push("\"validTime\": " + this.validTime.toString() + ",\r\n" +
indent + "  ");
            }
            if (this.poseID != null && this.poseID.id != "") {
                sb.push("\"poseID\": \"" + this.poseID.id + "\",\r\n" + indent + "
");
            }
            if (this.parentPoseID != null && this.parentPoseID.id != "") {
                sb.push("\"parentPoseID\": \"" + this.parentPoseID.id + "\",\r\n" +
indent + "  ");
            }
            sb.push("\"position\": \r\n  {\r\n    " + indent + "\"lat\": " +
                (this.FrameTransform as FrameTransform.WGS84ToLTPENU).Origin.lat +
",\r\n    " + indent +
                "\"lon\": " + (this.FrameTransform as
FrameTransform.WGS84ToLTPENU).Origin.lon + ",\r\n    " + indent +
                "\"h\":   " + (this.FrameTransform as
FrameTransform.WGS84ToLTPENU).Origin.h);
            sb.push("\r\n  " + indent + "},");
            sb.push("\r\n  " + indent);
            sb.push("\"angles\": \r\n  {\r\n    " + indent + "\"yaw\":   " +
                (this.Orientation as Orientation.YPRAngles).yaw + ",\r\n    " + indent
+
                "\"pitch\": " + (this.Orientation as Orientation.YPRAngles).pitch +
",\r\n    " + indent +
                "\"roll\":  " + (this.Orientation as Orientation.YPRAngles).roll);
            sb.push("\r\n  " + indent + "}");
            sb.push("\r\n" + indent + "}");
        }
        return sb.join('');
    }

}

/// <summary>
/// A Basic-Quaternion GeoPose uses a unit quaternions to define the orientation of
the inner frame..
```

```
/// <remark>
/// See the OGS GeoPose 1.0 standard for a full description.
/// </remark>
/// </summary>
export class BasicQuaternion extends Basic {
    public constructor(id: string, tangentPoint: Position.GeodeticPosition,
quaternion: Orientation.Quaternion) {
        super();
        this.poseID = new Extras.PoseID(id);
        this.FrameTransform = new FrameTransform.WGS84ToLTPENU(tangentPoint);
        this.Orientation = quaternion;
    }

    /// <summary>
    /// An Orientation specified as a unit quaternion.
    /// </summary>
    public override Orientation: Orientation.Quaternion;

    /// <summary>
    /// This function returns a Json encoding of a Basic-Quaternion GeoPose
    /// </summary>
    public toJSON(): string {
        let indent: string = "";
        let sb: string[] = [''];
        if ((this.FrameTransform as FrameTransform.WGS84ToLTPENU).Origin != null &&
(this.Orientation as Orientation.Quaternion) != null) {
            sb.push("{\r\n  " + indent);
            if (this.validTime != null) {
                sb.push("\"validTime\": " + this.validTime.toString() + ",\r\n" +
indent + "  ");
            }
            if (this.poseID != null && this.poseID.id != "") {
                sb.push("\"poseID\": \"" + this.poseID.id + "\",\r\n" + indent + "
");
            }
            if (this.parentPoseID != null && this.parentPoseID.id != "") {
                sb.push("\"parentPoseID\": \"" + this.parentPoseID.id + "\",\r\n" +
indent + "  ");
            }
            sb.push("\"position\": \r\n  {\r\n    " + indent + "\"lat\": " +
                (this.FrameTransform as FrameTransform.WGS84ToLTPENU).Origin.lat +
",\r\n    " + indent +
                "\"lon\": " + (this.FrameTransform as
FrameTransform.WGS84ToLTPENU).Origin.lon +
                ",\r\n    " + indent +
                "\"h\":   " + (this.FrameTransform as
FrameTransform.WGS84ToLTPENU).Origin.h);
            sb.push("\r\n  " + indent + "},");
            sb.push("\r\n  " + indent);
            sb.push("\"quaternion\": \r\n  {\r\n    " + indent + "\"x\":   " +
                (this.Orientation as Orientation.Quaternion).x + ",\r\n      " +
```

```
indent +
                "\"y\": " + (this.Orientation as Orientation.Quaternion).y + ",\r\n
" + indent +
                "\"z\": " + (this.Orientation as Orientation.Quaternion).z + ",\r\n
" + indent +
                "\"w\":  " + (this.Orientation as Orientation.Quaternion).w);
            sb.push("\r\n  " + indent + "}");
            sb.push("\r\n" + indent + "}");
            return sb.join('');
        }
    }
}
```

Step 7: Advanced GeoPose

Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy.

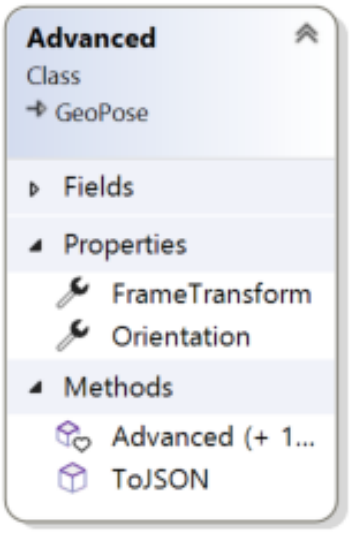

Figure 8. The Advanced Class

Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy.

TypeScript implemention:

```
import * as Extras from './Extras';
import * as FrameTransform from './FrameTransform';
import * as Orientation from './Orientation';
import * as GeoPose from './GeoPose';

// Implemention order: 7 - follows Basic GeoPose.
// This is the most general GeoPose - the largest part of the 20% part of a 80/20
solution.
// The difficult implementation is creating the interface layer between the
// Extrinsic specification and external authorities and data sources.

/// <summary>
/// Advanced GeoPose.
/// </summary>
export class Advanced extends GeoPose.GeoPose {
    public constructor(id: string, frameTransform: FrameTransform.Extrinsic,
orientation: Orientation.Quaternion) {
        super();
        this.poseID = new Extras.PoseID(id);
        this.FrameTransform = frameTransform;
        this.Orientation = orientation;
    }

    /// <summary>
    /// A Frame Specification defining a frame with associated coordinate system whose
Position is the origin.
    /// </summary>
    public override FrameTransform: FrameTransform.Extrinsic;

    /// <summary>
    /// An Orientation specified as a unit quaternion.
    /// </summary>
    public override Orientation: Orientation.Quaternion;

    /// <summary>
    /// This function returns a Json encoding of an Advanced GeoPose
    /// </summary>
    public toJSON(): string {
        let indent: string = "";
        let sb: string[] = [''];
        {
            sb.push("{\r\n" + indent + "  ");
            if (this.validTime != null) {
                sb.push("\"validTime\": " + this.validTime.toString() + ",\r\n" +
indent + "  ");
            }
            if (this.poseID != null && this.poseID.id != "") {
                sb.push("\"poseID\": \"" + this.poseID.id + "\",\r\n" + indent + "
```

```
");
            }
            if (this.parentPoseID != null && this.parentPoseID.id != "") {
                sb.push("\"parentPoseID\": \"" + this.parentPoseID.id + "\",\r\n" +
indent + "  ");
            }
            sb.push("\"frameSpecification\":\r\n" + indent + "  " + "{\r\n" + indent +
"    \"authority\": \"" +
                (this.FrameTransform as
FrameTransform.Extrinsic).authority.replace("\"", "\\\"") + "\",\r\n" + indent + "
\"id\": \"" +
                (this.FrameTransform as FrameTransform.Extrinsic).id.replace("\"",
"\\\"") + "\",\r\n" + indent + "    \"parameters\": \"" +
                (this.FrameTransform as
FrameTransform.Extrinsic).parameters.replace("\"", "\\\"") + "\"\r\n" + indent + "
},\r\n" + indent + "  ");
            sb.push("\"quaternion\":\r\n" + indent + "  {\r\n" + indent + "    \"x\":"
+ (this.Orientation as Orientation.Quaternion).x + ",\"y\":" +
                (this.Orientation as Orientation.Quaternion).y + ",\"z\":" +
                (this.Orientation as Orientation.Quaternion).z + ",\"w\":" +
                (this.Orientation as Orientation.Quaternion).w);
            sb.push("\r\n" + indent + "  }\r\n" + indent + "}\r\n");
            return sb.join('');
        }
    }
}
```

Step 8: Local [Geo]Pose

The Local GeoPose is in essence, an implementation of the pose concept from computer graphics. It can be implemented as an Advanced geoPose but a lightweight implementation for operations within a local Carsian coordinate system is often useful. I hope that a future version of OGC GeoPose has something like the Local form. Until then, there are three alternatives

- use a non-standard serialized form,
- serialize this class structure as an Advanced and compliant data object, or
- rely on the Advanced form.

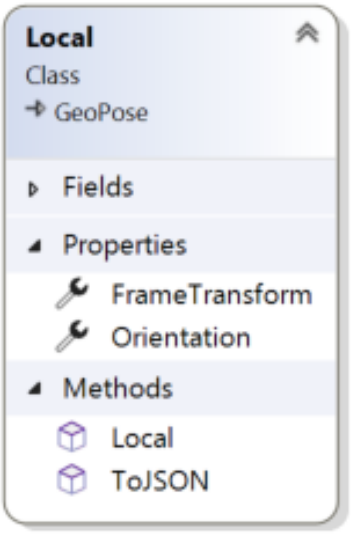

Figure 9. Local

Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy.

TypeScript implemention:

```
// Implemention order: 8 -a useful GeoPose for working within a local Cartesian (i.e.
engineering) frame.
// Local can be expressed as an Advanced form, but the Advanced form is more complex
and this implementation is a shortcut.

/// <summary>
/// Local GeoPose is a derived pose within an engineering CRS with a Cartesian
coordinate system.
/// This form is the closest to the classical computer graphics pose concept.
/// <remark>
/// WARNING: Local is not (yet) part of the OGC GeoPose standard and not backwards-
compatible.
/// Useful when operating within a local Cartesian frame defined by a Basic (or other)
GeoPose.
/// It is possible to define Local via the Advanced GeoPose with
///   "authority": "steve@opensiteplan.org-experimental", "id": "translation",
"parameters": {<dx>, <dy>, <dz> }
/// </remark>
/// </summary>
export class Local extends GeoPose.GeoPose {
    public constructor(id: string, frameTransform: FrameTransform.Translation,
orientation: Orientation.YPRAngles) {
        super();
```

```
        this.poseID = new Extras.PoseID(id);
        this.FrameTransform = frameTransform;
        this.Orientation = orientation;
    }
    /// <summary>
    /// The xOffset, yOffset, zOffset from the origin of the rotated inner frame of a
"parent" GeoPose.
    /// </summary>
    public override FrameTransform: FrameTransform.Translation;

    /// <summary>
    /// An Orientation specified as three rotations.
    /// </summary>
    public override Orientation: Orientation.YPRAngles;


    /// <summary>
    /// This function returns a Json encoding of an Advanced GeoPose
    /// </summary>
    public toJSON(): string {
        let indent: string = "";
        let sb: string[] = [''];
        {
            sb.push("{\r\n  ");
            if (this.validTime != null) {
                sb.push("\"validTime\": " + this.validTime.toString() + ",\r\n" +
indent + "  ");
            }
            if (this.poseID != null && this.poseID.id != "") {
                sb.push("\"poseID\": \"" + this.poseID.id + "\",\r\n" + indent + "
");
            }
            if (this.parentPoseID != null && this.parentPoseID.id != "") {
                sb.push("\"parentPoseID\": \"" + this.parentPoseID.id + "\",\r\n" +
indent + "  ");
            }
            sb.push("\"position\": \r\n  {\r\n    " + "\"x\": " + (this.FrameTransform
as FrameTransform.Translation).xOffset + ",\r\n    " +
                "\"y\": " + (this.FrameTransform as
FrameTransform.Translation).yOffset + ",\r\n    " +
                "\"z\":   " + (this.FrameTransform as
FrameTransform.Translation).zOffset);
            sb.push("\r\n  " + "},");
            sb.push("\r\n  ");
            sb.push("\"angles\": \r\n  {\r\n    " + "\"yaw\":   " + (this.Orientation
as Orientation.YPRAngles).yaw + ",\r\n    " +
                "\"pitch\": " + (this.Orientation as Orientation.YPRAngles).pitch +
",\r\n    " +
                "\"roll\":  " + (this.Orientation as Orientation.YPRAngles).roll);
            sb.push("\r\n  " + "}");
            sb.push("\r\n" + "}\r\n");
```

```
            return sb.join('');
        }
    }
}
```

Examples

Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy.

TypeScript

Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy.

C#

Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy.

References

Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy. Pile o text pile o text I am filler and I feel puffy.

TypeScript https://www.typescriptlang.org

C#

NET 6

C++

IEEE 754 Floating Point

Java

Kotlin

Python

SfM

SLAM

Swift