

Article

Representing Time-Dynamic Geospatial Objects on Virtual Globes Using CZML—Part I: Overview and Key Issues

Liangfeng Zhu ^{1,2,3,*}, Zhongliang Wang ^{1,2} and Zhiwen Li ^{1,2}

¹ Key Laboratory of Geographic Information Science (Ministry of Education), East China Normal University, Shanghai 200241, China; wzlh100@126.com (Z.W.); hi@zhiwenli.com (Z.L.)

² School of Geography Science, East China Normal University, Shanghai 200241, China

³ Shanghai Key Lab for Urban Ecological Processes and Eco-Restoration, East China Normal University, Shanghai 200241, China

* Correspondence: lzfzhu@geo.ecnu.edu.cn; Tel.: +86-136-7172-1009

Received: 24 January 2018; Accepted: 12 March 2018; Published: 13 March 2018

Abstract: Cesium Markup Language (CZML) is an emerging specification for the representation and exchange of time-dynamic geospatial objects on virtual globes. The principal focus of CZML is on the definition of time-varying characteristics that are important for applications of geospatial objects, such as changeable positions/extents, graphical appearances, and other geospatial properties. Due to its unique ability to stream massive geospatial datasets, CZML is ideally suited for efficient, incremental streaming to the client in the network environment. Our goal is to explore and outline the overall perspective of CZML as an efficient schema for representing time-dynamic geospatial objects on virtual globes. Such a perspective is the topic of the two present companion papers. Here, in the first part, we provide an overview of CZML and explore two key issues, and their associated solutions, for representing time-dynamic geospatial objects using CZML: one is how to use CZML properties to describe time-varying characteristics of geospatial objects, and the other is how to use CZML to support streaming data. These innovative improvements provide highly-efficient and more reliable supports for representing time-dynamic geospatial objects. The relevant applications, academic influence, and future developments of CZML are explored in a second paper.

Keywords: virtual globe; CZML; time-dynamic visualization; geospatial object; specification

1. Introduction

Over the past two decades, virtual globes have become a standard way to integrate, visualize, and analyze multidisciplinary geospatial information at both local and planetary scales [1–6]. A number of sophisticated and powerful virtual globes, such as Google Earth [7], NASA World Wind [8], Cesium [9], and other online Earth browsers, have been developed and, subsequently, accepted by Earth scientists from different subject fields as convenient platforms to conduct studies, exchange ideas, present results, and share knowledge with a global perspective in a natural and intuitive way [10–23]. The primary reason for virtual globes' success is that they provide more than just a digital replica of the Earth's surface with high-resolution images and terrain data. They also offer users an open framework for efficiently describing, displaying, and understanding their own geospatial objects in exquisite detail [24]. At present, there are plenty of universal standards/specifications for the representation and exchange of geospatial objects, such as GML [25], CityGML [26], and 3D Portrayal Service [27]. Among them, Keyhole Markup Language (KML) [28,29], an XML-based encoding schema, was adopted by nearly all major virtual globes as the dominant data format for describing custom geospatial objects [30–33]. As an open standard authorized by the Open Geospatial Consortium (OGC),

KML allows users to define how, where, and when custom objects are displayed or animated [34]. Using KML, scientific users can easily create dynamic, interactive displays on virtual globes [24]. Due to the authority and broad scope of applications, KML has become widely used for representing geospatial objects in the geoscientific environment, allowing spatially-based objects to be distributed, visualized, and analyzed in one seamless virtual globe interface [35].

In the few years since the approval of the OpenGIS KML 2.2 Encoding Standard (OGC KML 2.2) in 2008 [29], KML has found numerous applications in several branches of geoscience, including geology [32,36–42], geophysics [43–45], geochemistry [46], climate change [47–49], weather forecasting [50], natural disaster management [51,52], and online education games [53–55]. These applications primarily use KML to generate geographic annotations and visualizations on 3D virtual globes. While KML has proven to be quite useful for representing geographical maps and spatially-based geographical information, there remain several serious limitations when using KML to interact with geospatial objects in a virtual environment. One of the current main limitations of KML is the lack of advanced capability and necessary flexibility in representing time-dynamic geospatial objects. It is well known that the Earth is three-dimensional and rapidly changing over time. With the recent technological advances in geospatial information acquisition, such as global navigation satellite systems (GNSS), remote sensing (RS), mobile computing, and sensor networks, the number of time-dynamic geospatial objects and their applications in various disciplines are steadily increasing [56–60]. Prominent examples of time-dynamic geospatial objects are satellites, aircraft, cars, ships, wild animals, mobile subscribers, meteorological disasters, forest fires, land cover, snow coverage in mountains, and tectonic plates [61–64]. Potential applications of time-dynamic geospatial objects include aerospace, defense, sports, tourism, real-time observations of environmental change, tracking the path and behavior of moving objects, situation awareness in disaster management, tectonic reconstructions, and time-based modelling of geologic processes.

The importance of time-dynamic geospatial objects and their applications is sharply increasing. For most of the applications of time-dynamic geospatial objects, not only the geometrical (including spatial positions and extents) and graphical (visual appearance) aspects are changing over time, but also their semantic properties (such as the function of objects) are dynamically updated [65]. Representing geospatial objects within virtual globes requires the implementation of both the geometrical and graphical aspects, as well as the time-varying geospatial properties. Unfortunately, up to now, KML's support for time-dynamic objects has been limited. Two critical problems that users of virtual globes may encounter when trying to use KML to represent time-dynamic objects are listed below.

One problem with KML is its deficiency to represent dynamically-updated properties of time-dynamic geospatial objects, especially when property values can be determined by interpolating over provided time-tagged samples. In the KML specification prior to version 2.3, a time element (`kml:TimeStamp` or `kml:TimeSpan`) can be associated with a KML Feature (such as a place mark, a ground overlay, etc.) to restrict the visibility of the data set to a given time [33]. However, this method is only suited for controlling the geometry and appearance of time-dynamic geospatial objects. It is not appropriate in cases where time-varying semantics are required. Moreover, even to describe the time-varying geometry or appearance of a given object, KML still has some prominent issues affecting the representation and visualization performance. For example, in order to represent a moving vehicle over a specified period of time, users are required to duplicate the object many times to create dozens of vehicles, each containing a time element. This mechanism produces a great data redundancy, and reduces the performance of the virtual globe client. The recently-released OGC KML 2.3 [28] provides two new geometry elements, `kml:Track` and `kml:MultiTrack`, to capture and display the path of moving objects. Those two geometries offer a compact encoding of time series data by associating a single object with multiple time elements, and also provide the interpolation functionality for calculating the position of a time-varying object along the track. However, those new functions relate only to point icons (`kml:Point`) and 3D models (`kml:Model`). They cannot be exported to other frequently-used KML geometry elements, like `kml:LineString`, `kml:LinearRing`, or `kml:Polygon`.

A second problem faced by KML is that of real-time data transmission for time-dynamic geospatial information. Due to the limitation of its grammatical rule, KML does not provide the capability for streaming time-varying data in the network environment. When the virtual globe client processes a KML-formatted time-dynamic object, the complete dataset associated with that object, often in a large volume, should be fetched in advance [33]. This limitation makes it extremely difficult to process and display large-scaled time-dynamic geospatial objects in a real-time manner.

The above-mentioned shortcomings of KML have become more obvious as Earth scientists struggle to describe and analyze large-scaled, real-time, dynamic objects on virtual globes. There is a pressing need to provide a more modern, universal, and web-friendly schema to describe time-dynamic geospatial objects. The recent development of the Cesium Markup Language (CZML) has produced a great potential for earth scientists to solve this problem. CZML is a JSON-based schema for describing time-dynamic objects on 3D virtual globes and 2D maps. It represents the geometry, appearance, and properties of geospatial objects, and accurately specifies how they change with time. While CZML can also handle traditional static data sets, the principal focus of CZML is on the definition of time-varying characteristics that are important for applications of geospatial objects: changeable positions/extents/appearances or other geospatial properties. Furthermore, CZML is also easily packetized and streamable, making it ideal for real-time, data-driven visualization and analysis on the web. Hence, CZML is a suitable standard or specification for getting time-dynamic geospatial objects into web-based virtual globes.

CZML was initiated in 2011 by the Cesium team, an active community started by Analytical Graphics, Inc. (AGI) (Exton, PA, USA), as a new data format specifically for its open-source, cross-platform Earth browser named Cesium [9,66]. Since that time, CZML has developed alongside Cesium, but it can also be implemented in other sophisticated virtual-globe-like clients [66]. The currently-available CZML specification is version 1.0. Up to now CZML is not an OGC specification, and the development of CZML is still a work in progress. However, because of its treatment of time-dynamic objects as a first-class citizen, its popularity is growing rapidly so that it is now widely used in a diverse array of fields for the geospatial industry [9]. Nowadays, plenty of CZML-based data, tools, and applications are shared on the World Wide Web (cf. the accompanying second paper of our series). From those perspectives, CZML is complementary to most of the key existing OGC standards including GML, CityGML and 3D Portrayal Service, even though they are not necessarily designed for virtual globes. The developers of CZML intend to propose CZML as an international standard of OGC for the representation and exchange of time-dynamic geospatial objects.

In this paper, we provide an overview of CZML, including its overall structure, underlying concepts, and main characteristics. For a complete presentation on details of CZML, including the CZML specification, tools and services, reference documents, examples, applications, and other materials, we refer the reader to the CZML Guide website [66,67], or to the Cesium Home page [9].

The remainder of this paper is structured as follows: In Section 2, an overview of CZML is presented. The principal focus is on the overall structure, as well as a basic concept, the CZML property for describing geospatial objects. The details of CZML's functionality are discussed in two sections: Section 3 introduces how to use CZML properties to describe time-varying characteristics of geospatial objects, while Section 4 concentrates on CZML's unique ability to support streaming data. We believe these key issues and their innovative improvements will provide highly-efficient and more reliable supports for representing time-dynamic geospatial objects on virtual globes.

2. CZML: Overview

In this section, an overview of CZML (version 1.0) is given. The focus is set on the two most important aspects: the overall structure of CZML, and the CZML properties for describing geospatial objects.

2.1. CZML Overall Structure

CZML is a JSON-based markup language for encoding representations of time-dynamic geospatial objects. JSON (JavaScript Object Notation) is a lightweight, text-based, language-independent data interchange format [68], derived from the ECMAScript programming language [69]. It defines a small set of structuring rules for the portable representation of structured data, which makes it have a self-descriptive, but clear, framework. JSON is a text format that is completely programming language independent. It is not only suitable for humans to read and write, but also suitable for machines to parse and generate. CZML is a subset of the JSON specification, meaning that a valid CZML document is also a valid JSON document [66,70].

As with JSON, CZML uses a simple, human-readable notation to record data objects consisting of a collection of name/value pairs. All the details of geospatial objects, including the geometrical aspect and the graphical appearance, as well as the semantic property and the time, can be depicted by CZML. The ultimate result of describing geospatial objects is a CZML document, represented as a string or a plaintext file with the *.czml* extension.

As shown in Figure 1, a CZML document is a single array which consists of a set of JSON objects, and each object-literal element in the array is a CZML packet [67,70]. A CZML packet is a collection of name/value pairs, which describe the geometrical/graphical/semantic/temporal aspects for a single geospatial object. Each packet has an *id* property and several other CZML properties for defining the object described by the packet. Some of the CZML properties may have other additional sub-properties to refine the characteristics of the object.

```
[
  // Packet 1: The first packet in CZML is a document object, serving as the document declaration.
  {"id": "document", // The id property of the first packet must be "document".
   "version": "1.0" // The CZML version being written. Only valid on the document object.
  },

  // Packet 2: The second packet.
  {"id": "myObject1", //The ID of the object described by this packet. Any unique identifier.
   "Property 1": property value 1,
   "Property 2": {"Sub-property 1": sub-property 1,
                 "Sub-property 2": sub-property 2,
                 // ... Other sub-properties.
                },
   // ... Other properties.
  },

  // Packet 3: The third packet.
  {"id": "myObject2",
   "Property 1": property value 1,
   "Property 2": property value 2
  },

  // Packet 4
  {"id": "myObject2",
   "Property 3": property value 3,
   // ... Other properties.
  },

  // ... Other packets.
]
```

Figure 1. General structure of a CZML document. The first packet (Packet 1) in a CZML document must be a document object. A CZML property may have its own sub-properties, just like “Property 2” in the second packet (Packet 2). In this example, Packet 3 and Packet 4 have the same *id*, indicating they are used to describe different aspects of a same geospatial object.

The CZML packet is the basic data unit for processing and analyzing objects, aiming to represent geospatial objects which may have plenty of dynamically updated characteristics. In a CZML document,

multiple packets are placed sequentially, and the relationship between different packets is parallel. In the ordinary course of events, a single geospatial object is represented using a single packet that describes all of the characteristics associated with the object. However, this is not necessarily the case. If an object involves large quantities of information to be described, putting the entire object in one large packet would make it difficult to load the data incrementally. To avoid this problem, a geospatial object can be divided into multiple CZML packets with the same *id*, describing different properties of the same object, or a same property varying with time. The data volume of individual packets is moderate, fitting for incremental loading and displaying on virtual globes. This is particularly useful when a CZML property is defined over many intervals or when an interval contains many time-tagged samples. By breaking the complete definition of a geospatial object into multiple packets, time-dynamic geospatial data can be transmitted progressively over the network environment.

2.2. CZML Properties for Describing Geospatial Objects

In CZML, a collection of name/value pairs is used to describe geospatial objects by means of explicit enumeration. Each pair of name/value records a specific aspect of a data object, like the location, extent, appearance, or semantics. Such a pair of name/value is called a CZML property. Table 1 groups CZML properties into two main categories: standard and extended properties. The table also includes three additional criteria that CZML users should consider: whether or not a property possesses additional sub-properties, whether to allow changing over time, and the level of support for interpolating an unknown value (see Section 3).

A standard property is one that describes the most frequently used aspect of a geospatial object. As shown in Table 1, the standard properties are classified into three different groups: Group 1, Group 2, and Group 3, depending on their characteristics and functions:

- **Group 1:** This group of standard properties is commonly used to record the general information of a geospatial object or a CZML document object.
- **Group 2:** This group of standard properties is used to define the position/orientation or the custom properties of a geospatial object, or the suggested camera location when viewing the object, or an interval of time associated with a time-varying property.
- **Group 3:** This group of standard properties is used to describe the geometry (positions and extents) and graphical appearance of a geospatial object. Such a property is called a CZML geometry property.

CZML provides 15 types of geometry properties to describe the geometry of geospatial objects. Using geometry properties listed in Table 2, users of CZML not only can efficiently define simple geometric shapes, like points, polylines and polygons, but also can conveniently describe several complicated shapes and volumes, like ellipsoids, cylinders, corridors, and 3D models.

In CZML, individual geometry properties may contain several additional sub-properties. Table 2 lists some of the most frequently used sub-properties defined for each geometry property. A sub-property also can contain its own sub-properties. All those sub-properties are used to control the visual appearance and graphical style of geospatial objects, such as the visibility, color, material, texture, and other exclusive, personalized characteristics. Most of those sub-properties can be combined with the *interval* property to represent dynamically updated graphical styles or visual appearances for geospatial objects.

CZML also has an efficient extension mechanism to allow for additional capability beyond what is available in the standard properties. Based on the requirement of practical applications, users can customize extended properties to represent more complicated geospatial objects. For example, in order to represent and visualize complicated geometrical shapes in the space and defense industries, AGI has provided a series of CZML extensions to support a number of new features such as sensors, vectors, and fans geometries (cf. Table 1). These extensions use the *agi* prefix.

Table 1. Summary of main CZML properties for describing geospatial objects.

Type	Property Name	Description	Sub-Property ¹	Changeable ²	Interpolatable ³	
Standard property	<i>id</i>	The ID of the object described by a packet.	X	X	X	
	<i>delete</i>	Whether the client should delete all existing data for this object.	X	X	X	
	Group 1	<i>name</i>	The name of the object.	X	X	X
		<i>parent</i>	The ID of the parent object, if any.	X	X	X
		<i>version</i>	The CZML version being written.	X	X	X
		<i>clock</i>	The clock settings for the entire data set.	✓	✓	X
		<i>description</i>	An HTML description of the object.	✓	✓	X
		<i>availability</i>	The set of time intervals over which data for an object is available.	X	X	X
		Group 2	<i>position</i>	The position of the object in the world.	✓	⊘
	<i>properties</i>		A set of custom properties for the object.	✓	⊘	⊘
	<i>orientation</i>		The orientation of the object in the world.	✓	⊘	⊘
	<i>viewFrom</i>		A suggested camera location when viewing the object.	✓	⊘	⊘
	<i>interval</i>		An interval of time, associated with a time-varying property to define its value over different time intervals.	X	X	X
	Group 3	<i>point</i>	A point or viewport-aligned circle.	✓	⊘	⊘
		<i>polyline</i>	A line in the scene composed of multiple segments.	✓	⊘	⊘
		<i>rectangle</i>	An equiangular quadrilateral.	✓	⊘	⊘
		<i>polygon</i>	A closed figure on the surface of the Earth.	✓	⊘	⊘
		<i>ellipse</i>	A closed curve on the surface of the Earth.	✓	⊘	⊘
		<i>box</i>	A closed rectangular cuboid.	✓	⊘	⊘
		<i>corridor</i>	A shape defined by a centerline and width that conforms to the curvature of the globe.	✓	⊘	⊘
		<i>cylinder</i>	A cylinder, truncated cone, or cone.	✓	⊘	⊘
		<i>polylineVolume</i>	A polyline with a volume, or a 2D shape extruded along a polyline.	✓	⊘	⊘
		<i>ellipsoid</i>	A closed quadric surface that is a three dimensional analogue of an ellipse.	✓	⊘	⊘
		<i>wall</i>	A series of points, which extrude down to the ground.	✓	⊘	⊘
		<i>model</i>	A 3D model.	✓	⊘	⊘
		<i>path</i>	A polyline defined by the motion of an object over time.	✓	⊘	⊘
		<i>label</i>	A string of text.	✓	⊘	⊘
Extended property	<i>billboard</i>	A viewport-aligned image positioned in the 3D scene, sometimes called a marker.	✓	⊘	⊘	
	<i>agi_conicSensor</i>	A conical sensor volume taking into account occlusion of an ellipsoid.	✓	⊘	⊘	
	<i>agi_customPatternSensor</i>	A custom sensor volume taking into account occlusion of an ellipsoid.	✓	⊘	⊘	
	<i>agi_rectangularSensor</i>	A rectangular pyramid sensor volume taking into account occlusion of an ellipsoid.	✓	⊘	⊘	
	<i>agi_fan</i>	A fan that starts at a point or apex and extends in a specified list of directions from the apex.	✓	⊘	⊘	
	<i>agi_vector</i>	A graphical vector.	✓	⊘	⊘	

Note: ¹ For **Sub-property**: ✓ marks that this property possesses some additional sub-properties, and X marks that it does not possess any additional sub-property; ² For **Changeable**: ✓ marks that this property allows changing over time, X marks that the property does not allow changing over time, and ⊘ marks that only some of the sub-properties in this property allow changing over time; ³ For **Interpolatable**: X marks that this property does not support for interpolating, and ⊘ marks that only some of the sub-properties in this property are interpolatable.

Table 2. Summary of CZML geometry properties for describing the geometry of geospatial objects.

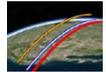
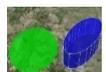
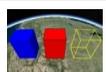
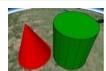
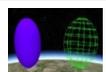
Geometry Property	Location Definition	Additional Sub-Properties	Illustration
<i>point</i>	position-dependent	<i>show, color, pixelSize, scaleByDistance, outlineColor, outlineWidth, heightReference, and translucencyByDistance.</i>	
<i>polyline</i>	position-independent	<i>show, positions, material, width, granularity, shadows, and followSurface.</i>	
<i>rectangle</i>	position-independent	<i>show, coordinates, material, height, extrudedHeight, granularity, rotation, stRotation, fill, outline, outlineColor, outlineWidth, closeBottom, closeTop, and shadows.</i>	
<i>polygon</i>	position-independent	<i>show, positions, height, extrudedHeight, material, outline, outlineColor, outlineWidth, granularity, fill, perPositionHeight, closeTop, closeBottom, stRotation, and shadows.</i>	
<i>ellipse</i>	position-dependent	<i>show, semiMajorAxis, semiMinorAxis, rotation, material, height, extrudedHeight, granularity, stRotation, fill, shadows, numberOfVerticalLines, outline, outlineColor, and outlineWidth.</i>	
<i>box</i>	position-dependent	<i>show, dimensions, fill, material, outline, shadows, outlineColor, and outlineWidth.</i>	
<i>corridor</i>	position-independent	<i>show, positions, width, height, cornerType, extrudedHeight, fill, material, outline, granularity, shadows, outlineColor, and outlineWidth.</i>	
<i>cylinder</i>	position-dependent	<i>show, length, topRadius, bottomRadius, fill, material, numberOfVerticalLines, slices, shadows, outline, outlineColor, and outlineWidth.</i>	
<i>polylineVolume</i>	position-independent	<i>show, positions, shape, cornerType, fill, material, outline, granularity, shadows, outlineColor, and outlineWidth.</i>	
<i>ellipsoid</i>	position-dependent	<i>show, radii, fill, material, outline, outlineColor, outlineWidth, stackPartitions, slicePartitions, shadows, and subdivisions.</i>	
<i>wall</i>	position-independent	<i>show, positions, material, minimumHeights, maximumHeights, granularity, fill, shadows, outline, outlineColor, and outlineWidth.</i>	

Table 2. Cont.

Geometry Property	Location Definition	Additional Sub-Properties	Illustration
<i>model</i>	position-dependent	<i>show, gltf, scale, runAnimations, incrementallyLoadTextures, minimumPixelSize, maximumScale, shadows, heightReference, silhouetteColor, silhouetteSize, color, colorBlendMode, colorBlendAmount, and nodeTransformations.</i>	
<i>path</i>	position-dependent	<i>show, material, width, resolution, leadTime, and trailTime.</i>	
<i>label</i>	position-dependent	<i>show, text, style, fillColor, outlineColor, outlineWidth, showBackground, backgroundColor, backgroundPadding, font, scale, translucencyByDistance, eyeOffset, pixelOffset, pixelOffsetScaleByDistance, horizontalOrigin, heightReference, and verticalOrigin.</i>	
<i>billboard</i>	position-dependent	<i>show, image, color, eyeOffset, horizontalOrigin, verticalOrigin, pixelOffset, scale, rotation, width, height, scaleByDistance, sizeInMeters, alignedAxis, pixelOffsetScaleByDistance, translucencyByDistance, heightReference, and imageSubRegion.</i>	

3. CZML Properties Varying with Time

3.1. Static Versus Dynamic Properties

One of the prime merits of CZML is the ability to accurately describe time-varying characteristics of geospatial objects, including their geometrical aspects, graphical appearances, and semantic properties. As listed in Table 1, not all CZML properties are time-varying, though many of them are, particularly most of the sub-properties defined for CZML geometry properties. According to their changeability, CZML properties can be divided into two categories: static and dynamic properties. The value of a static property is constant across the entire life cycle of the described object. Examples of static properties are the *id*, *delete*, *name*, *parent*, *availability*, and *version* properties. On the contrary, the value of a dynamic property is optionally changing over different time intervals or moments.

Furthermore, CZML properties can be classified into two different types: interpolatable and uninterpolatable, depending on the level of support for interpolating an unknown property value at any given time. All the static properties are uninterpolatable since their values are constant across time. Dynamic properties are more complex than static properties. Some of the dynamic properties, such as the *clock* and *description* properties, and the *show* sub-property, are uninterpolatable because they represent the semantics of the described object, and their data types are string or boolean. Other dynamic properties, notably properties that represent the positions of objects, are interpolatable as they are number-type properties. Prominent examples of interpolatable properties are the *position* property in a packet, as well as the *positions/coordinates/color/scale* sub-properties for a CZML geometry property. The interpolatable properties also can be called as the sampled properties because their values can be determined by interpolating over provided time-tagged samples.

For defining time-varying dynamic properties, there are two cases to be considered: one is to specify properties using time intervals, and one is to specify properties using time tags [33]. The time interval represents an extent in time bounded by a begin time and an end time. The time tag represents a single moment in time. Sections 3.2 and 3.3 will describe how to use time intervals or time tags to specify a time-varying dynamic property to a series of time spans or a collection of moments, respectively.

3.2. Specifying Time-Varying Properties Using Time Intervals

In CZML, the value of the *interval* property is specified using an ISO8601 date and time string [71]. It represents a period of time, which is associated with a time-varying property to specify different property values over different time intervals. The time-varying property possesses a fixed value over a time interval, and may possess other constants over other intervals. Any dynamic property, whether it is represented with one of the simple JSON data types (string, number, or boolean) or is represented by a complicated composite array (such as a Cartesian position or a color), can be associated with the *interval* property to define its different values over different time intervals.

Figure 2 illustrates how to use the *interval* property to define the values of a time-varying property over a series of time intervals. In general, the value of the described CZML property (here, named “*someIntervalProperty*”) is a JSON array, where each element in the array is an object literally defining the value of the property for a different interval of time [70]. The time intervals defined by multiple *interval* properties for a same property, such as *TimeInterval_1* and *TimeInterval_2* in Figure 2, may be contiguous or discrete. If two time intervals are contiguous, the transition of the property value from one interval to the next is an instant change. If there is a time gap between two discrete time intervals, the property value associated with this time gap is unknown, and it cannot be calculated by interpolating along the timeline.

Here we illustrate how to use the *interval* property to animate a time-varying polygon that transitions instantly from one time interval to the next (the structural outline of this CZML file is shown in Appendix A). As shown in Figure 3 and the online application [72], the polygon has different geometries, appearances, and descriptions over four different intervals of time, the first from 16:00 to 16:20 where the polygon is a pink quadrilateral (Figure 3a), the second from 16:20 to 16:30 where

the polygon is a pink triangle (Figure 3b), the third from 16:30 to 16:40 where the polygon is a cyan triangle (Figure 3c), and the fourth from 16:40 to 17:00 where the polygon is a cyan quadrilateral (Figure 3d). This example also demonstrates that the *interval* property not only can be used to control the time-varying change of the simple number-type/string-type CZML properties, but also can be used to deal with complicated composite property values, such as the *positions* and *color* sub-properties for a *polygon* property.

```

{
  "id": Object_ID,
  ... //Other properties
  "someIntervalProperty":
  [
    {
      "interval": TimeInterval_1,
      "propertyValue": PropertyValue_1
    },
    {
      "interval": TimeInterval_2,
      "propertyValue": PropertyValue_2
    },
    ... //Other time intervals and property values
  ],
  ... //Other properties
}

```

Figure 2. General structure of specifying a time-varying property over different time intervals using the *interval* property.

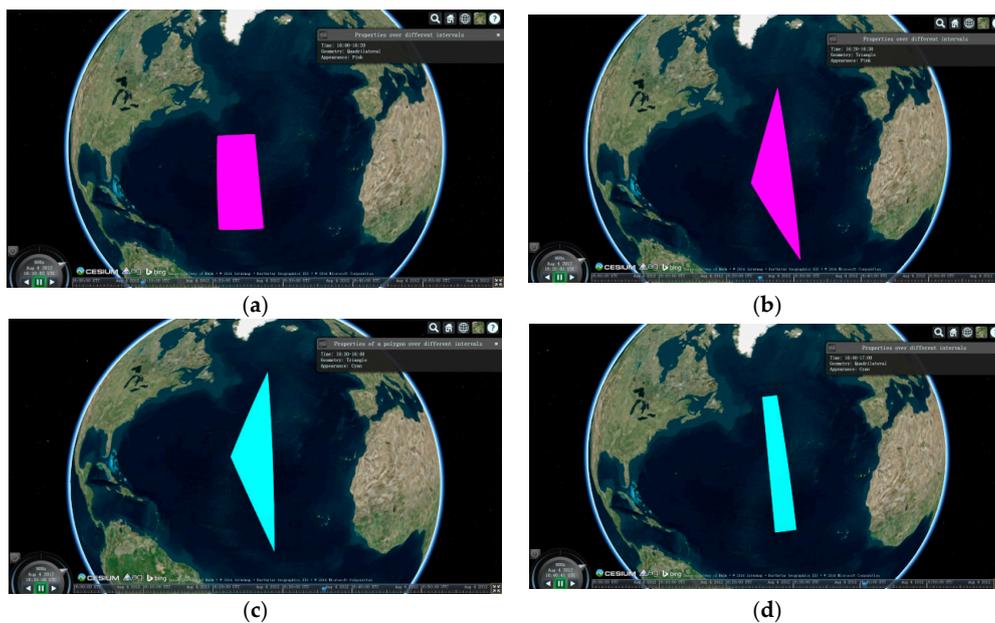


Figure 3. An animated polygon varying over a series of time intervals: (a) TimeInterval_1 (from 16:00 to 16:20); (b) TimeInterval_2 (from 16:20 to 16:30); (c) TimeInterval_3 (from 16:30 to 16:40); and (d) TimeInterval_4 (from 16:40 to 17:00).

3.3. Specifying Time-Varying Properties Using Time Tags

CZML offers a relatively easy way to specify and interpolate time-tagged dynamic properties on virtual globes. Time tags are specified using ISO8601 date and time strings [71] or seconds since an epoch. Property values in between two time tags can be automatically interpolated in the CZML clients. Therefore, the time-varying property can change its value over every second. At the same time, the CZML client will produce a continuous and smooth visible animation through the automatic interpolation.

It should be pointed out that not all dynamic properties are allowed to change their values by interpolating time-tagged samples. Only those who are explicitly declared as interpolatable in the CZML specification, generally number-typed properties, are allowed to specify their values using time tags.

As shown in Figure 4, CZML provides a compact encoding for time series data by associating a single property with multiple time tags. For a given time-varying property (like “*someTaggedProperty*”), a collection of time-tagged samples is listed in a property value array (like “*propertyValue*”), arranged as [TimeTag_1, PropertyValue_1, TimeTag_2, PropertyValue_2, . . .]. CZML provides seven optional sub-properties that can be associated with the property value array to control the interpolation:

- ***epoch***. In CZML, times can be specified in seconds since an epoch. The *epoch* sub-property, specified as an ISO 8601 date and time string [71], defines the time 0.0 for sampling a time-varying property and associated sub-properties.
- ***interpolationAlgorithm***. This specifies the algorithm to be used for interpolating a value at a different time from the provided time-tagged data. Possible values are “LINEAR”, “LAGRANGE”, or “HERMITE” [73–75].
- ***interpolationDegree***. This specifies the degree of the polynomial to use for interpolation.
- ***forwardExtrapolationType***. This specifies the type of extrapolation to perform when a value is requested at a time after any available samples. Valid values are “NONE”, “HOLD”, and “EXTRAPOLATE”.
- ***forwardExtrapolationDuration***. This defines the amount of time to extrapolate forward before the property becomes undefined. It can be used to determine if there is a time gap between samples specified in different packets.
- ***backwardExtrapolationType***. This specifies the type of extrapolation to perform when a value is requested at a time before any available samples.
- ***backwardExtrapolationDuration***. This defines the amount of time to extrapolate backward before the property becomes undefined. This sub-property can be used to determine if there is a time gap between samples specified in different packets.

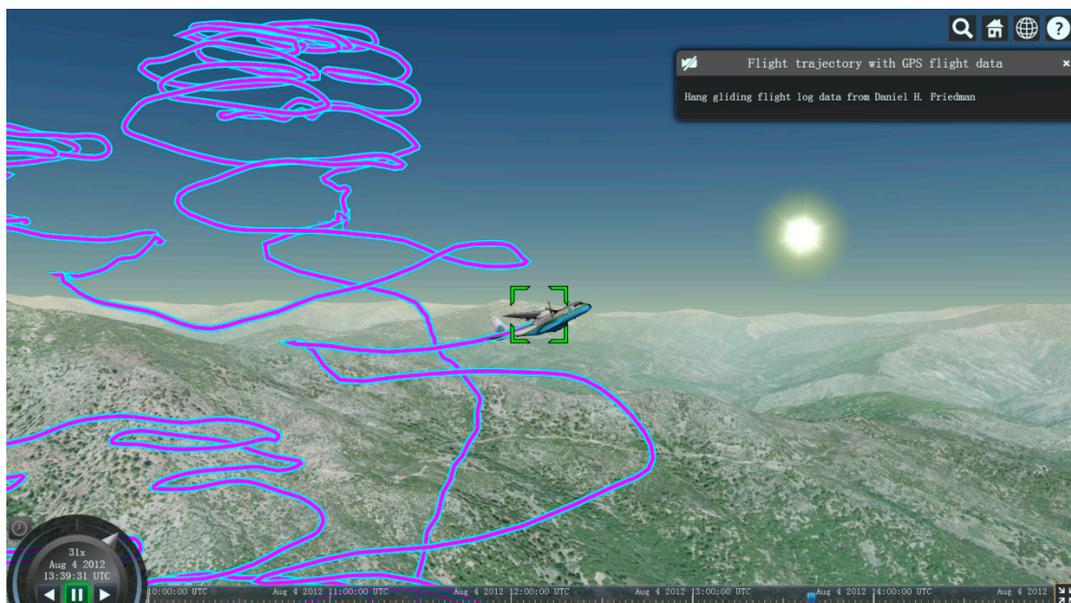
We offer an example of using time tags to animate a time-varying flight trajectory with GPS flight log data. For simplicity, the flight trajectory data has been omitted, and the detailed CZML-formatted document can be obtained from [73]. In this example, we animate a 3D aircraft model whose *id* is “myPlaneObject”. As shown in Figure 5 and the online application [74], the position and the orientation of the aircraft are changing over time. Therefore, the flight trajectory of the moving aircraft can be vividly represented using time-tagged positions.

```

{
  "id": Object_ID,
  ... //Other properties
  "someTaggedProperty":
  {
    "epoch": Epoch,
    "interpolationAlgorithm": InterpolationAlgorithm,
    "interpolationDegree": InterpolationDegree,
    "forwardExtrapolationType": ForwardExtrapolationType,
    "forwardExtrapolationDuration": Next_sample_time_tag,
    "backwardExtrapolationType": BackwardExtrapolationType,
    "backwardExtrapolationDuration": Previous_sample_time_tag,
    "propertyValue":[
      TimeTag_1, PropertyValue_1,
      TimeTag_2, PropertyValue_2,
      ... //Other time tags and property values
    ]
  },
  ... //Other properties
}

```

Figure 4. General structure of specifying an interpolatable time-varying property using time tags.



(a)

Figure 5. Cont.

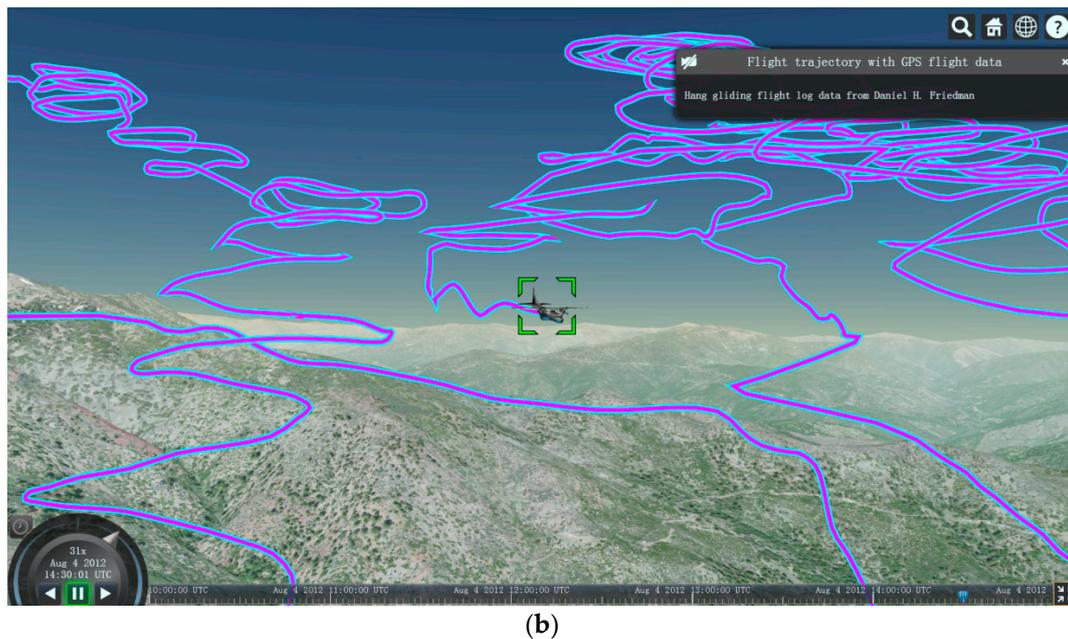


Figure 5. Example of using time tags to animate a flight trajectory with time-varying positions. Two figures compare the flight trajectory and the position of the aircraft at the moments of 13:39:31 (a) and 14:30:01 (b).

4. Streaming CZML

For time-dynamic geospatial objects, their positions/extents and/or other geospatial characteristics are changing continuously over time. Correspondingly, the amount of geospatial data would be increased rapidly as time goes on. Then, how to structure those increasing geospatial data in order to incrementally transmit them to the virtual globe client in the network environment, especially when the datasets are in large volume, is always a challenging problem faced by developers and users of all online virtual globes [76].

CZML is a JSON-based schema for describing geospatial objects along with their time-varying properties. It preserves the main advantage of JSON by providing a streamable scene description for data-driven visualization, making it ideal for efficient, incremental streaming to the client over the Internet. In CZML, the CZML properties of a single geospatial object can be split into one or more CZML packets. Each packet can contain as much, or as little, data according to the actual needs. Multiple packets are placed side by side with a relatively independent relationship, correlated by the *id* property. Therefore, users of CZML can freely adjust the data size of an individual packet to control it not too large, making it well-suited for incremental loading and display on the virtual globe client. Using the server-sent events (EventSource) API that is widely supported by modern browsers [77], CZML packets can be progressively transmitted to the client to facilitate high-performance streaming. When using the EventSource API, each CZML packet is streamed to the client as a separate event. As a result, the client raises an event when each packet is received, containing the data for just that one packet. The entire CZML document need not be present on the client before the scene can be displayed [70]. This approach allows the CZML client to incrementally process CZML packets with excellent performance, drastically enhancing the capabilities of CZML by arranging the size of data transfers, and achieves near-real-time transmission, visualization, and analysis of a large volume of real-time geospatial objects on limited network bandwidth and transferring speed.

As mentioned earlier, in order to stream a large volume of geospatial data efficiently, CZML allows users to distribute the properties of a geospatial object into several CZML packets. Those packets possess the same *id*, and describe different aspects of the same object. This means that each packet

only describes some partial characteristics of the object, and multiple packets can be linked together via the same *id* property to reflect the panorama.

In fact, in some cases, two or more packets with the same *id* property can even describe the same CZML property of a geospatial object. This is particularly useful when the property is defined over many time intervals or when an interval contains many time-tagged samples. When CZML users describe a time-varying property, putting the entire property data into one large packet would make it difficult to load the packet incrementally. Loading a very large block of data all at once may lock up the client until it finishes. In addition, although today's web browsers allow some access to a stream before it is complete, parsing and interpreting the incomplete data requires slow and cumbersome string manipulations [70]. In order to solve this problem, CZML allows users to break the complete definition of a time-varying property into multiple packets with appropriate sizes. Each packet can be efficiently transmitted and visualized in the network environment. The CZML client can obtain the relevant data sooner to minimize the time that the user must wait before the client starts rendering the scene. This trick allocates the size of data transfers properly while keeping the client responsive, and allows visualization with almost zero latency on a decent broadband connection.

For example, we capture and describe the position of a moving point object over a long period of time such that it creates a large number of time-tagged samples. In this situation, the *position* property of the moving object can be distributed into multiple CZML packets to support streaming. As shown in Figure 6, several packets (like Packet_1 and Packet_2) are defined to describe CZML properties of a single object whose *id* is "Point_1". The first packet (Packet_1) includes the *id*, *name* and *color* properties, as well as four time-tagged positions, indicating the starting point and the following three samples. The second packet (Packet_2) only contains the *id* property and four time-tagged positions following the previous samples. In this manner, a large quantity of time-varying positions can be divided into numerous packets to transmit incrementally and display synchronously.

```
[
  { // Packet_1
    "id": "Point_1",
    "name": "A time-dynamic point",
    "point":
      { "color": { "rgba": [255, 255, 0, 255] } },
    "position":
      { "epoch": "2012-08-04T16:00:00Z",
        "cartographicDegrees":
          [ 0, -70, 20, 100000,
            100, -80, 44, 120000,
            200, -90, 18, 130000,
            300, -98, 52, 140000 ] }
  },

  { // Packet_2
    "id": "Point_1",
    "position":
      { "cartographicDegrees":
          [ 400, -120, 20, 150000,
            450, -100, 34, 160000,
            480, -105, 40, 170000,
            540, -130, 52, 180000 ] }
  },

  ... //More packets recording the time-varying positions of Point_1, omitted
]
```

Figure 6. Example of distributing a CZML property into multiple packets to support streaming.

When splitting a time-varying property into multiple CZML packets, the orders of the times and the property values must be arranged in light of the following rules:

- If the time-varying property is specified using time intervals, it is not necessary to arrange the time intervals and corresponding property values in order of increasing time intervals, regardless of whether the property values are placed within a packet or across multiple packets. When the client receives a CZML packet, it walks through each property contained in the packet. For each property, it walks through each time interval over which the property is defined. For each time interval, it determines if the specified time interval has already been defined for the property. If the time interval has already been defined, the existing interval is updated; otherwise, a new one is created. When a new time interval overlaps existing intervals, the new interval takes precedence and the existing intervals are truncated or removed entirely.
- If the time-varying property is specified using time tags, the time tags and corresponding property values must be ordered by increasing time within a single CZML packet. Across two or more packets, however, it is not necessary for the time-tagged property values to be presented in any particular order. However, if adjacent packets are not arranged in order of increasing time, two additional sub-properties, *backwardExtrapolationDuration* and *forwardExtrapolationDuration* (cf. Section 3.3), need to be employed with the time-tagged property values to ensure reasonable interpolation when streaming non-contiguous samples. It is not necessary to specify both *backwardExtrapolationDuration* and *forwardExtrapolationDuration*, though one or the other might be more convenient in different situations. If either is specified, adjacent samples are checked before interpolation.

Appendix B provides an example of using the *backwardExtrapolationDuration* and *forwardExtrapolationDuration* sub-properties to control the streaming transmission and automatic interpolation of time-tagged samples.

5. Summary

We have introduced CZML, an alternative markup language that is well-suited to represent time-dynamic geospatial objects on virtual globes. CZML is an open, web-friendly data format, specifically designed for the purpose of describing time-dynamic 3D scenes. It completely describes the geometrical, graphical, and semantic aspects of geospatial objects, and accurately specifies how they change with time. In addition to representing dynamic positions/extents of geospatial objects, CZML is also suitable for describing geospatial properties that vary over time, including visual appearances and semantic properties. A key capability of CZML is the unique ability to stream massive time-dynamic geospatial datasets, which meets the demand for progressive transmission of geospatial information on virtual globes. CZML is optimized for streaming and data-driven visualization. A single geospatial object can be packetized into multiple data units, making it ideal for incrementally loading and displaying real-time data on virtual globes. These innovative improvements are helpful to promote the representation, exchange, and visualization of time-dynamic geospatial objects on virtual globes. CZML is implemented in the Cesium virtual globe designed for the modeling and visualization of dynamic-data in the geospatial industry, and Part II of this work [78] focuses on relevant applications, academic influence, and future developments of CZML.

Acknowledgments: The research leading to this paper was supported by the National Natural Science Foundation of China (Grant No. 41672327), the Natural Science Foundation of Shanghai Municipality (Grant No. 16ZR1408900), the Project of Shanghai Science and Technology Committee (Grant No. 17DZ1202804 & 17DZ1202805), and the Social Science Foundation of Shanghai (Grant No. 2014BCK002). We would like to thank the members of the Cesium team and community for their persistent engagement in developing CZML. Furthermore, we gratefully thank the editor and four anonymous reviewers for their useful comments that improved the quality of the manuscript.

Author Contributions: Liangfeng Zhu was in charge of the research work and wrote the paper. Zhongliang Wang and Zhiwen Li supervised the research work and the manuscript conception. All authors participated to the improvement of the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Using the *Interval* Property to Animate a Time-Varying Polygon

Here is the CZML code of using the *interval* property to animate a time-varying polygon that transitions instantly from one time interval to the next:

```
{
  "id": "myPolygonObject",
  "name": "Properties of a polygon over different intervals",
  "availability": "2012-08-04T16:00:00Z/2012-08-04T17:00:00Z",
  "description": [{
    "interval": "2012-08-04T16:00:00Z/2012-08-04T16:20:00Z", //TimeInterval_1
    "string": "Time: 16:00-16:20<br/>Geometry: Quadrilateral<br/>Appearance: Pink"
  },
  {
    "interval": "2012-08-04T16:20:00Z/2012-08-04T16:30:00Z", //TimeInterval_2
    "string": "Time: 16:20-16:30<br/>Geometry: Triangle<br/>Appearance: Pink"
  },
  {
    "interval": "2012-08-04T16:30:00Z/2012-08-04T16:40:00Z", //TimeInterval_3
    "string": "Time: 16:30-16:40<br/>Geometry: Triangle<br/>Appearance: Cyan"
  },
  {
    "interval": "2012-08-04T16:40:00Z/2012-08-04T17:00:00Z", //TimeInterval_4
    "string": "Time: 16:40-17:00<br/>Geometry: Quadrilateral<br/>Appearance: Cyan"
  }
  ],
  "polygon": {
    "positions": [{
      "interval": "2012-08-04T16:00:00Z/2012-08-04T16:20:00Z", //TimeInterval_1
      "cartographicDegrees": [-50,20,0,-50,40,0,-40,40,0,-40,20,0]
    },
    {
      "interval": "2012-08-04T16:20:00Z/2012-08-04T16:40:00Z", //TimeInterval_2 &
      TimeInterval_3
      "cartographicDegrees": [-35,50,0,-35,10,0,-45,30,0]
    },
    {
      "interval": "2012-08-04T16:40:00Z/2012-08-04T17:00:00Z", //TimeInterval_4
      "cartographicDegrees": [-35,50,0,-40,50,0,-40,20,0,-35,20,0]
    }
  ]
  },
  "material": {
    "solidColor": {
      "color": [{
        "interval": "2012-08-04T16:00:00Z/2012-08-04T16:30:00Z", //TimeInterval_1 &
        TimeInterval_2
        "rgbaf": [1,0,1,1]
      },
      {
        "interval": "2012-08-04T16:30:00Z/2012-08-04T17:00:00Z", //TimeInterval_3 &
        TimeInterval_4
        "rgbaf": [0,1,1,1]
      }
    ]
  }
}
```

```

    }}
  }
}
}
}

```

Appendix B. Using CZML Properties to Control the Streaming Transmission and Automatic Interpolation of Time-Tagged Samples

Here is an example of using the *backwardExtrapolationDuration* and *forwardExtrapolationDuration* sub-properties to control the streaming transmission and automatic interpolation of time-tagged samples:

```

[[
  // Packet_1
  "id": "myObject1",
  "someInterpolatableProperty": {
    "epoch": "2012-04-30T12:00:00Z",
    "cartesian": [
      0.0,1.0,2.0,3.0,
      1.0,4.0,5.0,6.0,
      2.0,7.0,8.0,9.0,
      3.0,10.0,11.0,12.0
    ],
    "backwardExtrapolationDuration": 0,
    "forwardExtrapolationDuration": 4.0
  }
},
{
  // Packet_2
  "id": "myObject1",
  "someInterpolatableProperty": {
    "cartesian": [
      8.0,25.0,26.0,27.0,
      9.0,28.0,29.0,30.0,
      10.0,31.0,32.0,33.0
    ],
    "backwardExtrapolationDuration": 7.0,
    "forwardExtrapolationDuration": 11.0
  }
},
{
  // Packet_3
  "id": "myObject1",
  "someInterpolatableProperty": {
    "cartesian": [
      4.0,13.0,14.0,15.0,
      5.0,16.0,17.0,18.0,
      6.0,19.0,20.0,21.0,
      7.0,22.0,23.0,24.0
    ],
    "backwardExtrapolationDuration": 3.0,

```

```

        "forwardExtrapolationDuration": 8.0
    }
},
... // More packets
]

```

In this example, the time-varying property ("*someInterpolatableProperty*") has sampled values at times 0.0 through 10.0, inclusive, at 1.0 s intervals, and the time-tagged samples are distributed into three packets. The first packet (Packet_1) contains times 0.0 through 3.0, the second packet (Packet_2) contains times 8.0 through 10.0, and the third packet (Packet_3) contains times 4.0 through 7.0. Obviously, those packets are not presented in order of increasing time.

When receiving the first two packets, the CZML client would find a time gap between times 3.0 and 8.0 because it has not yet received the third packet containing times 4.0 through 7.0. If the *backwardExtrapolationDuration* and *forwardExtrapolationDuration* sub-properties do not appear in the first two packets, the CZML client would fill the gap preferentially by interpolating across those two packets. This probably produces a significant difference with the actual data as provided in Packet_3, because interpolating across this gap can result in a very wrong-looking scene, especially with higher-degree interpolation.

However, this problem can be efficiently resolved using the *backwardExtrapolationDuration* and *forwardExtrapolationDuration* sub-properties, as in our example above. In Packet_1, the value of the *forwardExtrapolationDuration* property is specified as 4.0. This property explicitly tells the CZML client that the next sample time after 3.0 is 4.0. If the next sample it has after 3.0 is 8.0, as in our code above, the client knows that there is a gap, and it will wait for more data before interpolating at a time within that gap. Similarly, the *backwardExtrapolationDuration* and *forwardExtrapolationDuration* sub-properties specified in Packet_2 and Packet_3 also play the same role.

References

1. Cozzi, P.; Ring, K. *3D Engine Design for Virtual Globes*; CRC Press: Boca Raton, FL, USA, 2011.
2. Blaschke, T.; Donert, K.; Gossette, F.; Kienberger, S.; Marani, M.; Qureshi, S.; Tiede, D. Virtual Globes: Serving Science and Society. *Information* **2012**, *3*, 372–390. [[CrossRef](#)]
3. De Paor, D.G.; Dordevic, M.M.; Karabinos, P.; Burgin, S.; Coba, F.; Whitmeyer, S.J. Exploring the reasons for the seasons using Google Earth, 3D models, and plots. *Int. J. Digit. Earth* **2017**, *10*, 582–603. [[CrossRef](#)]
4. De Paor, D.G.; Dordevic, M.M.; Karabinos, P.; Tewksbury, B.J.; Whitmeyer, S.J. The Fold Analysis Challenge: A virtual globe-based educational resource. *J. Struct. Geol.* **2016**, *85*, 85–94. [[CrossRef](#)]
5. Zhu, L.; Zhang, C.; Li, M.; Pan, X.; Sun, J. Building 3D solid models of sedimentary stratigraphic systems from borehole data: An automatic method and case studies. *Eng. Geol.* **2012**, *127*, 1–13. [[CrossRef](#)]
6. Zhu, L.F.; Li, M.J.; Li, C.L.; Shang, J.G.; Chen, G.L.; Zhang, B.; Wang, X.F. Coupled modeling between geological structure fields and property parameter fields in 3D engineering geological space. *Eng. Geol.* **2013**, *167*, 105–116. [[CrossRef](#)]
7. Google Earth. Available online: <https://www.google.com/earth/> (accessed on 30 January 2018).
8. NASA World Wind. Available online: <https://worldwind.arc.nasa.gov/> (accessed on 30 January 2018).
9. Cesium—WebGL Virtual Globe and Map Engine. Available online: <http://cesiumjs.org> (accessed on 4 November 2017).
10. Goodchild, M.F.; Guo, H.; Annoni, A.; Bian, L.; De, B.K.; Campbell, F.; Craglia, M.; Ehlers, M.; Van, G.J.; Jackson, D. Next-generation Digital Earth. *Proc. Natl. Acad. Sci. USA* **2012**, *109*, 11088–11094. [[CrossRef](#)] [[PubMed](#)]
11. Keysers, J. *Review of Digital Globes 2015*; The Australia and New Zealand Cooperative Research Centre for Spatial Information: Melbourne, Australia, 2015; p. 24.
12. Liang, J.; Gong, J.; Li, W.; Ibrahim, A.N. Visualizing 3D atmospheric data with spherical volume texture on virtual globes. *Comput. Geosci.* **2014**, *68*, 81–91. [[CrossRef](#)]

13. Mahdavi-Amiri, A.; Alderson, T.; Samavati, F. A Survey of Digital Earth. *Comput. Graph.* **2015**, *53*, 95–117. [[CrossRef](#)]
14. Muller, R.D.; Qin, X.D.; Sandwell, D.T.; Dutkiewicz, A.; Williams, S.E.; Flament, N.; Maus, S.; Seton, M. The GPlates Portal: Cloud-Based Interactive 3D Visualization of Global Geophysical and Geological Data in a Web Browser. *PLoS ONE* **2016**, *11*, e0150883. [[CrossRef](#)] [[PubMed](#)]
15. Schroth, O.; Pond, E.; Campbell, C.; Cizek, P.; Bohus, S.; Sheppard, S.R.J. Tool or Toy? Virtual Globes in Landscape Planning. *Future Int.* **2011**, *3*, 204–227. [[CrossRef](#)]
16. Tian, Y.; Zheng, Y.; Zheng, C. Development of a visualization tool for integrated surface water-groundwater modeling. *Comput. Geosci.* **2016**, *86*, 1–14. [[CrossRef](#)]
17. Tiede, D.; Lang, S. Analytical 3D views and virtual globes—Scientific results in a familiar spatial context. *ISPRS J. Photogramm. Remote Sens.* **2010**, *65*, 300–307. [[CrossRef](#)]
18. Bernardin, T.; Cowgill, E.; Kreylos, O.; Bowles, C.; Gold, P.; Hamann, B.; Kellogg, L. Crusta: A new virtual globe for real-time visualization of sub-meter digital topography at planetary scales. *Comput. Geosci.* **2011**, *37*, 75–85. [[CrossRef](#)]
19. Wu, H.; He, Z.; Gong, J. A virtual globe-based 3D visualization and interactive framework for public participation in urban planning processes. *Comput. Environ. Urban Syst.* **2010**, *34*, 291–298. [[CrossRef](#)]
20. Yu, L.; Gong, P. Google Earth as a virtual globe tool for Earth science applications at the global scale: Progress and perspectives. *Int. J. Remote Sens.* **2012**, *33*, 3966–3986. [[CrossRef](#)]
21. Du, X.; Guo, H.; Fan, X.; Zhu, J.; Yan, Z.; Zhan, Q. Vertical accuracy assessment of freely available digital elevation models over low-lying coastal plains. *Int. J. Digit. Earth* **2016**, *9*, 252–271. [[CrossRef](#)]
22. Heavner, M.J.; Fatland, D.R.; Hood, E.; Connor, C. SEAMONSTER: A demonstration sensor web operating in virtual globes. *Comput. Geosci.* **2011**, *37*, 93–99. [[CrossRef](#)]
23. Liu, P.; Gong, J.H.; Yu, M. Visualizing and analyzing dynamic meteorological data with virtual globes: A case study of tropical cyclones. *Environ. Model. Softw.* **2015**, *64*, 80–93. [[CrossRef](#)]
24. Bailey, J.E.; Chen, A.J. The role of Virtual Globes in geoscience. *Comput. Geosci.* **2011**, *37*, 1–2. [[CrossRef](#)]
25. OGC Geography Markup Language (GML)—Extended Schemas and Encoding Rules. OGC10-129r1. Available online: https://portal.opengeospatial.org/files/?artifact_id=46568 (accessed on 23 January 2018).
26. OGC City Geography Markup Language (CityGML) Encoding Standard. Available online: https://portal.opengeospatial.org/files/?artifact_id=47842 (accessed on 23 January 2018).
27. OGC® 3D Portrayal Service 1.0. Available online: <http://docs.opengeospatial.org/is/15-001r4/15-001r4.html> (accessed on 30 January 2018).
28. OGC KML 2.3. OGC12-007r2. Available online: <http://docs.opengeospatial.org/is/12-007r2/12-007r2.html> (accessed on 28 October 2017).
29. OGC KML. OGC07-147r2. Available online: http://portal.opengeospatial.org/files/?artifact_id=27810 (accessed on 28 October 2017).
30. Chiang, G.T.; White, T.O.H.; Dove, M.T.; Bovolo, C.I.; Ewen, J. Geovisualization Fortran library. *Comput. Geosci.* **2011**, *37*, 65–74. [[CrossRef](#)]
31. Chien, N.Q.; Tan, S.K. Google Earth as a tool in 2-D hydrodynamic modeling. *Comput. Geosci.* **2011**, *37*, 38–46. [[CrossRef](#)]
32. De Paor, D.G.; Whitmeyer, S.J. Geological and geophysical modeling on virtual globes using KML, COLLADA, and Javascript. *Comput. Geosci.* **2011**, *37*, 100–110. [[CrossRef](#)]
33. Wernecke, J. *The KML Handbook: Geographic Visualization for the Web*; Addison-Wesley: Boston, MA, USA, 2009.
34. KML. Available online: <http://www.opengeospatial.org/standards/kml> (accessed on 28 October 2017).
35. Ballagh, L.M.; Raup, B.H.; Duerr, R.E.; Khalsa, S.J.S.; Helm, C.; Fowler, D.; Gupte, A. Representing scientific data sets in KML: Methods and challenges. *Comput. Geosci.* **2011**, *37*, 57–64. [[CrossRef](#)]
36. Zhu, L.; Pan, X.; Sun, J. Visualization and dissemination of global crustal models on virtual globes. *Comput. Geosci.* **2016**, *90*, 34–40. [[CrossRef](#)]
37. De Paor, D.G. Virtual Rocks. *GSA Today* **2016**, *26*, 4–11. [[CrossRef](#)]
38. De Paor, D.G.; Wild, S.C.; Dordevic, M.M. Emergent and animated COLLADA models of the Tonga Trench and Samoa Archipelago: Implications for geoscience modeling, education, and research. *Geosphere* **2012**, *8*, 491–506. [[CrossRef](#)]
39. Zhu, L.; Wang, X.; Pan, X. Moving KML geometry elements within Google Earth. *Comput. Geosci.* **2014**, *72*, 176–183. [[CrossRef](#)]

40. Dordevic, M.M.; De Paor, D.G.; Whitmeyer, S. Geologic mapping in Google Earth: Tools and challenges. *Geol. Soc. Am. Abstr. Prog.* **2014**, *46*, 92.
41. Mochales, T.; Blenkinsop, T.G. Representation of paleomagnetic data in virtual globes: A case study from the Pyrenees. *Comput. Geosci.* **2014**, *70*, 56–62. [[CrossRef](#)]
42. Zhu, L.F.; Wang, X.F.; Zhang, B. Modeling and visualizing borehole information on virtual globes using KML. *Comput. Geosci.* **2014**, *62*, 62–70. [[CrossRef](#)]
43. Postpischl, L.; Danecek, P.; Morelli, A.; Pondrelli, S. Standardization of seismic tomographic models and earthquake focal mechanisms data sets based on web technologies, visualization with keyhole markup language. *Comput. Geosci.* **2011**, *37*, 47–56. [[CrossRef](#)]
44. Yamagishi, Y.; Yanaka, H.; Suzuki, K.; Tsuboi, S.; Isse, T.; Obayashi, M.; Tamura, H.; Nagao, H. Visualization of geoscience data on Google Earth: Development of a data converter system for seismic tomographic models. *Comput. Geosci.* **2010**, *36*, 373–382. [[CrossRef](#)]
45. Zhu, L.; Kan, W.; Zhang, Y.; Sun, J. Visualizing the Structure of the Earth's Lithosphere on the Google Earth Virtual-Globe Platform. *ISPRS Int. J. Geo-Inf.* **2016**, *5*, 26. [[CrossRef](#)]
46. Yamagishi, Y.; Suzuki, K.; Tamura, H.; Yanaka, H.; Tsuboi, S. Visualization of geochemical data for rocks and sediments in Google Earth: Development of a data converter application for geochemical and isotopic data sets in database systems. *Geochem. Geophys. Geosyst.* **2011**, *12*, 428–452. [[CrossRef](#)]
47. Chen, A.J.; Leptoukh, G.; Kempler, S.; Lynnes, C.; Savtchenko, A.; Nadeau, D.; Farley, J. Visualization of A-Train vertical profiles using Google Earth. *Comput. Geosci.* **2009**, *35*, 419–427. [[CrossRef](#)]
48. Erickson, T.A.; Michalak, A.M.; Lin, J.C. A Data System for Visualizing 4-D Atmospheric CO₂ Models and Data. *OSGeo J.* **2010**, *8*, 37–47.
49. Wang, Y.; Huynh, G.; Williamson, C. Integration of Google Maps/Earth with microscale meteorology models and data visualization. *Comput. Geosci.* **2013**, *61*, 23–31. [[CrossRef](#)]
50. Smith, T.M.; Lakshmanan, V. Real-time, rapidly updating severe weather products for virtual globes. *Comput. Geosci.* **2011**, *37*, 3–12. [[CrossRef](#)]
51. Leidig, M.; Teeuw, R. Free software: A review, in the context of disaster management. *Int. J. Appl. Earth Obs.* **2015**, *42*, 49–56. [[CrossRef](#)]
52. Tomaszewski, B. Situation awareness and virtual globes: Applications for disaster management. *Comput. Geosci.* **2011**, *37*, 86–92. [[CrossRef](#)]
53. Habib, E.; Ma, Y.; Williams, D. Development of a web-based hydrologic education tool using Google Earth resources. *Geol. Soc. Am. Abstr. Prog.* **2012**, *492*, 431–439.
54. Lee, T.K.; Guertin, L. Building an education game with the Google Earth application programming interface to enhance geographic literacy. *Geol. Soc. Am. Spec. Pap.* **2012**, *492*, 395–401.
55. Zhu, L.; Pan, X.; Gao, G. Assessing Place Location Knowledge Using a Virtual Globe. *J. Geogr.* **2016**, *115*, 72–80. [[CrossRef](#)]
56. Amini, F.; Rufiange, S.; Hossain, Z.; Ventura, Q.; Irani, P.; McGuffin, M.J. The Impact of Interactivity on Comprehending 2D and 3D Visualizations of Movement Data. *IEEE Trans. Vis. Comput. Graph.* **2015**, *21*, 122–135. [[CrossRef](#)] [[PubMed](#)]
57. Andrienko, G.; Andrienko, N.; Dykes, J.; Fabrikant, S.I.; Wachowicz, M. Geovisualization of dynamics, movement and change: Key issues and developing approaches in visualization research. *Inf. Vis.* **2008**, *7*, 173–180. [[CrossRef](#)]
58. Ferreira, K.R.; Vinhas, L.; Monteiro, A.M.V.; Camara, G. Moving Objects and KML Files. In Proceedings of the 28th IEEE International Conference on Data Engineering (ICDE 2012) Workshop on Spatio Temporal data Integration and Retrieval, Arlington, VA, USA, 1–5 April 2012; pp. 355–359.
59. Huang, Y.-K. Within Skyline Query Processing in Dynamic Road Networks. *ISPRS Int. J. Geo-Inf.* **2017**, *6*, 137. [[CrossRef](#)]
60. Li, Z.; Guan, X.; Li, R.; Wu, H. 4D-SAS: A Distributed Dynamic-Data Driven Simulation and Analysis System for Massive Spatial Agent-Based Modeling. *ISPRS Int. J. Geo-Inf.* **2016**, *5*, 42. [[CrossRef](#)]
61. Dordevic, M.M.; Whitmeyer, S.J. MaRGEE: Move and Rotate Google Earth Elements. *Comput. Geosci.* **2015**, *85*, 1–9. [[CrossRef](#)]
62. Ferreira, K.R.; Vinhas, L.; Monteiro, A.M.V.; Camara, G. Moving objects and spatial data sources. *Rev. Bras. Cartogr.* **2012**, *64*, 796–806.

63. Saeedi, S.; Liang, S.; Graham, D.; Lokuta, M.F.; Mostafavi, M.A. Overview of the OGC CDB Standard for 3D Synthetic Environment Modeling and Simulation. *ISPRS Int. J. Geo-Inf.* **2017**, *6*, 306. [CrossRef]
64. Potapov, E.; Hronusov, V. Extreme dynamic mapping: Animals map themselves on the “Cloud”. *Geol. Soc. Am. Spec. Pap.* **2012**, *492*, 139–145.
65. Chaturvedi, K.; Kolbe, T.H. Dynamizers—Modeling and implementing dynamic properties for semantic 3D city models. In Proceedings of the Eurographics Workshop on Urban Data Modelling and Visualisation, Delft, The Netherlands, 23 November 2015; pp. 43–48.
66. CZML Guide. Available online: <https://github.com/AnalyticalGraphicsInc/czml-writer/wiki/CZML-Guide> (accessed on 4 November 2017).
67. CZML Packet. Available online: <https://github.com/AnalyticalGraphicsInc/czml-writer/wiki/Packet> (accessed on 4 November 2017).
68. The JSON Data Interchange Format. Available online: <http://www.ecma-international.org/publications/standards/Ecma-404.htm> (accessed on 4 November 2017).
69. ECMAScript 2015 Language Specification. Available online: <http://www.ecma-international.org/ecma-262/6.0> (accessed on 4 November 2017).
70. CZML Structure. Available online: <https://github.com/AnalyticalGraphicsInc/czml-writer/wiki/CZML-Structure> (accessed on 4 November 2017).
71. A Summary of the International Standard Date and Time Notation. Available online: <http://www.cl.cam.ac.uk/~mgk25/iso-time.html> (accessed on 4 November 2017).
72. Time-dynamic Properties over Different Intervals. Available online: http://www.visualearth.org/czml/dynamic_czml_polygon.html (accessed on 4 November 2017).
73. CZML Example 3D Model. Available online: http://www.visualearth.org/czml/data/czmlexample_3dmodel.czml (accessed on 4 November 2017).
74. Time-tagged Properties for a 3D Model. Available online: http://www.visualearth.org/czml/dynamic_czml_3dmodel.html (accessed on 4 November 2017).
75. Interpolatable Property. Available online: <https://github.com/AnalyticalGraphicsInc/czml-writer/wiki/InterpolatableProperty> (accessed on 4 November 2017).
76. Zhu, L.F.; Sun, J.Z.; Li, C.L.; Zhang, B. SolidEarth: A new Digital Earth system for the modeling and visualization of the whole Earth space. *Front. Earth Sci.* **2014**, *8*, 524–539. [CrossRef]
77. HTML Standard: Communication. Available online: <https://html.spec.whatwg.org/multipage/comms.html#server-sent-events> (accessed on 4 November 2017).
78. Zhu, L.; Li, Z.; Wang, Z. Representing Time-Dynamic Geospatial Objects on Virtual Globes using CZML—Part II: Impact, Comparison, and Future Developments. *ISPRS Int. J. Geo-Inf.* **2018**, accepted.



© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).