

METHODS AND APPARATUS FOR ASSOCIATING ATTRIBUTES AT DIFFERENT LEVELS OF A HIERARCHICAL MESH CODEC

CROSS REFERENCE

- 5 **[0001]** This application claims the priority of European Patent Application No. 23306552.3, filed 20 September 2023, entitled "Methods and Apparatus for Associating Attributes at Different Levels of a Hierarchical Mesh Codec," which is incorporated herein by reference in its entirety.

BACKGROUND

- 10 **[0002]** The present disclosure relates to systems and methods for encoding and decoding a mesh. While the present disclosure is not limited to any specific mesh coding technology, some embodiments may make use of edgebreaker technology. Edgebreaker is a technology that is capable of efficiently coding the connectivity of a triangular mesh. In its most straightforward implementation, a mesh encoded using edgebreaker is represented by an ordered series made up of the symbols, C, L, E, R, and S, called the "CLERS" sequence. Generally speaking, starting
15 with an initial triangle, these symbols describe different ways to attach a new triangle, providing information on whether or how different edges of the new triangle are connected to one or more of the existing triangles.

- 20 **[0003]** Following the MPEG V-Mesh (now renamed V-DMC) call for proposals, the solution proposed by Apple was selected to become the foundation of the MPEG V-Mesh Test Model (TM). The proposal is described in K. Mammou, J. Kim, A. Tourapis and D. Podborski, "m59281 - [V-CG] Apple's Dynamic Mesh Coding CfP Response," Apple Inc., 2022. Summaries of figures from the proposal are provided in FIG. 1 and FIG. 2 of the present disclosure.

- 25 **[0004]** The test model, for some frames, uses a static mesh coder for the encoding of a base mesh that is then subdivided to obtain an approximation of the original mesh. Currently, the V-Mesh test model encodes a base mesh using the Google Draco implementation of an edgebreaker-based mesh encoder. Specifically, the implementation uses the "spirale reversi" version of edgebreaker as described in M. Isenburg and J. Snoeyink, "Spirale Reversi: Reverse decoding of the edgebreaker encoding," *Computational Geometry*, vol. 20, pp. 39-52, 2001.

[0005] For V-DMC, a framework is developed by extending V3C. V3C is described at ISO/IEC 23090-5:2021, Information technology — Coded representation of immersive media — Part 5: Visual volumetric video-based coding (V3C) and video-based point cloud compression (V-PCC).

[0006] In MPEG, the V-DMC standard has started to work on static mesh codec. The static mesh codec is specified in Annex-I of ISO/IEC 23090-29. The static mesh codec in Annex I is the default codec to compress static mesh. This includes mesh geometry, mesh connectivity and mesh attributes. The mesh attributes provide additional information about the mesh vertices or mesh faces.

[0007] The V-DMC framework may involve different attributes specified for a dynamic mesh sequence. In V-DMC, the underlying static mesh codec may decode mesh attributes per face or per vertex. These attributes may provide additional information about the static mesh e.g., color, texture coordinates, normals, reflectance information, transparency information, and/or user-defined attributes.

[0008] The attributes from the static mesh codec may further be used by the basemesh codec. A basemesh bitstream is further encapsulated in a V3C bitstream. The hierarchical structure of a V-DMC bitstream is shown in FIG. 6. An extension mechanism in V3C bitstream may provide information about the type of attributes that a V3C bitstream/framework will use from the underlying basemesh codec. Additionally, the basemesh bitstream may provide a mechanism to provide information about the type of mesh attribute of interest from the underlying static mesh codec.

SUMMARY

[0009] A dynamic mesh encoding method according to some embodiments comprises: obtaining an encoded static mesh, the encoded static mesh including a plurality of static mesh attributes; encoding a basemesh based on the encoded static mesh and motion information, the encoded basemesh including lower-level mapping information that assigns a lower-level attribute identifier to each static mesh attribute in at least a subset of the static mesh attributes; and encoding a dynamic mesh based on the encoded basemesh and atlas information, the encoded dynamic mesh including upper-level mapping information that assigns to each of a plurality of upper-level attributes a respective one of the lower-level attribute identifiers.

[0010] In some embodiments, one or more of the attributes are identified by an index. In some embodiments one or more of the attributes are identified by an arbitrary unique identifier.

[0011] In some embodiments, the lower-level attribute identifier is an index. In other embodiments, the lower-level attribute identifier is an arbitrary unique identifier.

[0012] A dynamic mesh decoding method according to some embodiments comprises: obtaining an encoded dynamic mesh, the encoded dynamic mesh including an encoded

basemesh, atlas information, and upper-level mapping information that assigns a respective lower-level attribute identifiers to each of a plurality of upper-level attributes; decoding the encoded basemesh, the encoded basemesh including an encoded static mesh, motion information, and lower-level mapping information, wherein the static mesh includes a plurality of static mesh attributes, and wherein the lower-level mapping information assigns a lower-level attribute identifier to each of a plurality of the static mesh attributes; and decoding the encoded static mesh, including decoding at least a first static mesh attribute, wherein the first static mesh attribute maps to at least one of the upper-level attributes through the lower-level mapping information and the upper-level mapping information.

10 **[0013]** In some embodiments, at least some (or all) static mesh attributes that do not map to an upper-level attribute are not decoded, or their values may be discarded.

[0014] Example embodiments further include the following: corresponding decoding techniques, apparatus comprising one or more processors configured to perform the methods herein, apparatus comprising at least one processor and a computer-readable medium storing instructions for performing the methods described herein, a computer-readable medium storing a mesh encoded according to the methods herein, and a signal conveying a mesh encoded according to the methods herein.

BRIEF DESCRIPTION OF THE DRAWINGS

20 **[0015]** The following detailed description will be better understood when read in conjunction with the appended drawings, in which there are shown examples of one or more of the multiple embodiments of the present disclosure. It should be understood, however, that the embodiments described herein are not limited to the precise arrangements and instrumentalities shown in the drawings.

[0016] FIG. 1 is a functional block diagram of an example mesh encoding system.

25 **[0017]** FIG. 2 is a functional block diagram of an example mesh decoding system.

[0018] FIG. 3 illustrates an example of an edgebreaker mesh codec according to some embodiments.

[0019] FIG. 4 is a flow chart of an encoding process according to some embodiments.

[0020] FIG. 5 is a flow chart of a decoding process according to some embodiments.

30 **[0021]** FIG. 6 schematically illustrates encapsulation of a different sub-bitstreams in a bitstream encoding a dynamic mesh such as a V-DMC bitstream.

[0022] FIG. 7 illustrates indexing of mesh attributes from different levels in the V-DMC bitstream.

35 **[0023]** FIG. 8 illustrates mapping of mesh attributes via unique arbitrary identifier (UAI) from different levels in the V-DMC bitstream.

[0024] FIG. 9 illustrates mapping of attributes between V3C extension and basemesh codec via indexing and mapping of attribute between basemesh codec and static mesh codec via UAI.

[0025] FIG. 10 is a block diagram of an example of a system in which various aspects and embodiments are implemented.

5

DETAILED DESCRIPTION

Overview of mesh geometry coding.

[0026] FIG. 3 illustrates an example of an edgebreaker mesh codec that may be used for mesh coding according to some embodiments. The top row is the encoding line, and the bottom row is the decoding line. As illustrated in FIG. 3, example embodiments include some or all of the following. During encoding, pre-processing 302 may be used to clean potential connectivity issues (non-manifolds edges and vertices) that can exist on the input mesh. This cleanup is performed because the edgebreaker algorithm by itself does not work on meshes having such connectivity issues. In some embodiments, cleaning non manifold edges and vertices involves duplicating a number of points. Some embodiments keep track of those duplicated vertices to merge those when decoding. This enables to reduce the number of points in the decoded mesh but calls for sending some additional information in the bitstream. In some embodiments, this pre-processing 302 further includes adding some dummy points to fill potential holes on the surface because the edgebreaker algorithm alone does not handle holes. Example embodiments operate to fill the holes before encoding and recreate the holes after decoding. Example embodiments use "virtual" dummy points and generate and encode dummy triangles attached to these dummy points, but the 3D positions of those points are not encoded or decoded. In some embodiments, the vertex attributes are quantized if needed. Those attributes can be provided to the coder already quantized.

[0027] Example embodiments encode the connectivity of the mesh using a modified version of the edgebreaker algorithm from which generates a CLERS table (a table made of 'C', 'L', 'E', 'R', and 'S' symbols). This stage also generates some tables in memory that are used for the attribute prediction stage. The vertex attributes are then predicted, starting by the position ones. Then, other attributes are predicted, eventually relying on the position predictions, which is the case for the texture UV coordinates. Configuration and metadata are also provided in the bitstream, the CLERS table, some other connectivity and all the attribute prediction residuals are entropy coded and added to the bitstream.

[0028] In an example decoding method, all of the entropy coded sub-bitstreams are decoded. The mesh connectivity is reconstructed using the CLERS table and the edgebreaker algorithm. Some extra information may be used to manage the handles, which describe the topology. The mesh connectivity is used as well as a minimal set of vertex positions expressed in 3D coordinates

to predict all the other per-vertex positions. The attribute residuals are applied to correct the predictions and obtain the reconstructed vertex positions. Then the other attributes are decoded, potentially relying on decoded positions, as for UV coordinates. The connectivity of attributes using separate index tables is reconstructed using per a per edge binary seam information that is entropy coded.

[0029] In a post-processing stage, the dummy triangles are removed. The non-manifold issues may be re-created in case the coder is configured to perform lossless coding, and the vertex attributes may be de-quantized if the model was quantized by encoder.

[0030] FIG. 4 is a flow chart of an encoding process that may be used in some embodiments.

[0031] FIG. 5 is a flow chart of a decoding process that may be used in some embodiments.

Overview of attribute relationships in mesh coding.

[0032] It is desirable to carefully map the mesh attributes coded by the MPEG static mesh codec to the attributes which will be exposed to the basemesh codec for further use. The bitstream coded using basemesh codec provides a reference to the attributes coded in the static mesh bitstream. With the same spirit, in the V3C extension, namely V3C parameter set V-DMC extension, the attributes are coded in the basemesh bitstream. The mesh attributes of the underlying static mesh bitstream are abstracted by the basemesh codec, which in turn expose some attributes to the V3C high level syntax extension.

[0033] All basemesh attributes with different basemesh attribute types can be potentially exposed in the V3C parameter set extension for V-DMC. However, the current semantics of basemesh attribute type misses on some of the required attributes, as described below in Section 1.b.

[0034] Some embodiments allow for one-to-one correspondence between the attributes defined in V3C extension and basemesh codec, as well as basemesh codec and static mesh codec. For example, two ATTR_NORMALS attributes can correspond to the information provided in two MESH_ATTR_NORMALS in the static mesh codec, given that the attribute being mapped are equivalent. However, the rules of equivalence are not described in the current working draft of ISO/IEC 23090-29.

[0035] Also, the basemesh codec may choose a subset of the mesh attributes. For example, the static mesh bitstream can contain two MESH_ATTR_NORMAL attributes. On the other hand, the basemesh bitstream may only be interested in one normal attribute and require mapping ATTR_NORMAL to one of the MESH_ATTR_NORMAL attributes in the static mesh codec. Since the basemesh bitstream abstracts that static mesh codec, this functionality can also enable exposure of only a subset of the coded attributes from the static mesh bitstream to the V3C extension via basemesh bitstream. There may be cases where the V3C extension may only use

a subset of basemesh attributes. Such flexibility can be achieved with a proper selection mechanism as described in different embodiments described herein.

[0036] In some situations, only one attribute of an attribute type exists in the static mesh codec, but the basemesh bitstream signals to refer to many of the same attribute, causing duplication. In
5 some embodiments, such references are forbidden. This is done in some embodiments by defining the range of the number of attributes to be referenced in the upper levels in the V-DMC hierarchy to adhere to the number of attributes available in the lower levels of the V-DMC hierarchy, as described below in Section 1.

[0037] Example embodiments address the potential for inconsistent attribute mapping between
10 the V3C extension and the basemesh codec. In the V3C extension for V-DMC, `vps_ext_bmesh_data_attribute_count` is one of the syntax elements. The syntax element indicates the number of attributes in the basemesh for the atlas with an atlas ID. In the basemesh sequence parameter set, `bmsps_mesh_attribute_count` indicates the number of attributes associated with the meshes coded using a static mesh codec.

15 [0038] Currently the range of `vps_ext_bmesh_data_attribute_count` is not specified. However, it is assumed that since `vps_ext_bmesh_data_attribute_count` references the basemesh attribute, the range of `vps_ext_bmesh_data_attribute_count` must be in the range of the number of attributes available in the basemesh i.e., `bmsps_mesh_attribute_count`.

[0039] Some embodiments provide for an equal attribute count for the V3C extension and the
20 basemesh codec. The two mentioned syntax element may correspond to the same value, such that `vps_ext_bmesh_data_attribute_count` = `bmsps_mesh_attribute_count`. The corresponding attribute may have the same type. In which case there is no requirement mapping attribute as it is an ideal case of one-to-one correspondence. However, there may be a case when the type of the referenced attributes is not the same. That is, `vps_ext_bmesh_attribute_type` ≠
25 `bmsps_mesh_attribute_type_id`. This may cause issues with bitstream conformance.

[0040] Example embodiments address different attribute counts for the V3C extension and the basemesh codec. For example, in some scenarios, syntax element `vps_ext_bmesh_data_attribute_count` is not equal to `bmsps_mesh_attribute_count`. In a case where the `vps_ext_bmesh_data_attribute_count` is greater than `bmsps_mesh_attribute_count`,
30 then there is a possibility of decoder crash, as the V3C extension may want to refer to a basemesh attribute which does not exist in the basemesh bitstream. However, there can be cases where the V3C extension wants to refer to an attribute multiple times. Such a case is impractical and may not be interesting for the applications. However, the decoder would not crash.

[0041] On the other hand, in case a case where the `vps_ext_bmesh_data_attribute_count` is
35 less than `bmsps_mesh_attribute_count`, then a subset of the basemesh attributes is referenced by the V3C extension. The current syntax in V3C extension lacks the ability to signal which attributes in the attribute list of basemesh bitstream are to be referenced. In some cases, there

may be multiple of the same attribute present in the basemesh bitstream e.g., a basemesh bitstream that contains two texture coordinate attributes and one color attribute. The current syntax in the V3C extension lacks the ability to signal how to choose a subset from the multiple of the same attribute (e.g., only one texture coordinate).

- 5 **[0042]** Some embodiments address issues of inconsistent attribute mapping between the basemesh codec and the static mesh codec. In the basemesh bitstream, `bmsps_mesh_attribute_count` is one of the syntax elements. The syntax element indicates the number of attributes associated with the mesh. The mesh is provided by the static mesh codec. In the MPEG edge breaker static mesh codec, `mesh_attribute_count` indicates the number of
10 encoded attributes. Currently the range of `bmsps_mesh_data_attribute_count` is constrained to be in the range of 0 to 127, inclusive. However, the range for `mesh_attribute_count` has not been specified.

- [0043]** Some embodiments address issues that can arise even when there is an equal attribute count for basemesh codec and static mesh codec. For example, the two mentioned syntax
15 elements may correspond to the same value, with `bmsps_mesh_attribute_count` being equal to `mesh_attribute_count`. The corresponding attribute may have the same type. In which case there is no requirement mapping attribute as it is an ideal case of one-to-one correspondence. However, there may be a case when the type of the referenced attributes is not the same. That is, `bmsps_mesh_attribute_type_id` is not the same as `mesh_attribute_type`. This may cause
20 issues with bitstream conformance.

- [0044]** Some embodiments address issues relating to different attribute counts for the basemesh codec and the static mesh codec, for example where syntax element `bmsps_mesh_attribute_count` is not equal to `mesh_attribute_count`. In a case where the `bmsps_mesh_attribute_count` is greater than `mesh_attribute_count`, then there is a possibility of
25 decoder crash as the basemesh bitstream may want to refer to a static mesh attribute which does not exist in the static mesh bitstream. However, there can be case where the basemesh bitstream wants to refer to a mesh attribute multiple time. Such a case is impractical and may not be interesting for the applications. However, the decoder would not crash.

- [0045]** In a case where the `bmsps_mesh_attribute_count` is less than `mesh_attribute_count`,
30 then a subset of the basemesh attributes is referenced by the basemesh bitstream. The current syntax in basemesh SPS lacks the ability to signal which attributes in the attribute list of basemesh bitstream are to be referenced.

- [0046]** In some cases, there may be multiples of the same attribute present in the static mesh bitstream e.g., a static mesh bitstream contains two texture coordinate attributes and one-color
35 attribute. The current syntax in the basemesh bitstream lacks the ability to signal how to choose a subset from the multiple of the same attribute (e.g., only one texture coordinate).

Issues addressed in some embodiments.

[0047] The hierarchical structure of the V-DMC codec, along with the complex and flexible usage of attributes may lead to inconsistent referencing of mesh attributes between different levels of the codec structure. For example, the basemesh codec may only be interested in the
 5 “Texture coordinate” attribute from the static mesh. But the static mesh bitstream may encapsulate other attributes along with “Texture coordinate”. Similarly, the V3C codec may only be interested in the “Normal” attribute from the basemesh codec. However, the basemesh codec may encapsulate additional attributes along with “NORMALS”. This may cause compatibility issues if the required attribute is absent from the underlying bitstream. The V-DMC specification
 10 does not provide a mechanism to ensure consistency of attribute usage and referencing within the hierarchical structure of the codec.

[0048] Furthermore, there may be multiple attributes of the same type present in the static mesh bitstream. However, the current V-DMC specification lacks the information to signal which one of the attributes of the same type is of interest to the basemesh codec. The same situation applies
 15 between the basemesh codec and the V3C codec. The V-DMC specification does not provide a mechanism to indicate which attribute(s) of the same type may be used by the parent codec.

Implementation of some example embodiments.

[0049] Example embodiments provide a mapping mechanism to associate the attributes from V3C extension to basemesh bitstream as well as basemesh bitstream to the static mesh
 20 bitstream. Several different embodiments are described herein.

[0050] § 1. Some embodiments impose additional semantic constraints on a bitstream encoding a mesh. In the current working draft, some of the syntax elements do not have a well-defined semantics or some of the items are missing. Some embodiments described herein include additions to the syntax of existing data structures; some such additions are marked with a dagger (†) in examples below.
 25

[0051] § 1.a. Some embodiments impose a constraint on the attribute count in the VPS extension. For example, where `vps_ext_bmesh_data_attribute_count[j]` indicates the number of attributes in the basemesh for the atlas with atlas ID `j`, and where the value of `vps_ext_mesh_data_attribute_count[j]` is in the range of 0 to `N`, inclusive, a constraint may be
 30 imposed such that `N` is equal to `bmsps_mesh_attribute_count - 1`. In that case, the modified semantics of `vps_ext_bmesh_data_attribute_count[j]` may read as follows:

`vps_ext_bmesh_data_attribute_count[j]` indicates the number of attributes in the basemesh for the atlas with atlas ID `j`. `vps_ext_mesh_data_attribute_count[j]` shall be in the range of 0 to `N`, inclusive where `N` is `bmsps_mesh_attribute_count - 1`.
 35

[0052] § 1.b. Some embodiments impose a constraint on a mesh attribute count in the basemesh. For example, where `bmsps_mesh_attribute_count` indicates the number of attributes

associated with the meshes, and where the value of `bmsps_mesh_attribute_count` is in the range of 0 to N, inclusive, a constraint may be imposed such that N is the number of attributes provided in the mesh bitstream coded using a static mesh codec identified via `bmsps_intra_mesh_codec_id`. In that case, the modified semantics of

5 `bmsps_mesh_attribute_count` may read as follows:

`bmsps_mesh_attribute_count` indicates the number of attributes associated with the meshes. `bmsps_mesh_attribute_count` shall be in the range of 0 to N, inclusive where N is the number of attributes provided in the mesh bitstream coded using a static mesh codec identified via `bmsps_intra_mesh_codec_id`.

10

[0053] § 1.c. Some embodiments introduce a texture coordinate attribute to the basemesh codec. For example, where `bmsps_mesh_attribute_type_id[i]` indicates the attribute type of the attribute with index i for the mesh, the value `bmsps_mesh_attribute_type_id[i] = 6` may be used to indicate that the attribute type is a texture coordinate. The semantics of

15 `bmsps_mesh_attribute_count` may read as follows:

`bmsps_mesh_attribute_type_id[i]` indicates the attribute type of the attribute with index i for the mesh. Table H-1 describes the list of supported attributes and their relationship with `bmsps_mesh_attribute_type_id[i]`.

20 **[0054]** Table H-2 may be used to describe the list of supported attributes and their relationship with `bmsps_mesh_attribute_type_id[i]`. Table H-2 may be modified with the addition of the line marked with a dagger (†) as follows:

Table H-1

<code>bmsps_mesh_attribute_type_id[i]</code>	Identifier	Attribute type
0	ATTR_TEXTURE	Texture
1	ATTR_MATERIAL_ID	Material ID
2	ATTR_TRANSPARENCY	Transparency
3	ATTR_REFLECTANCE	Reflectance
4	ATTR_NORMAL	Normals
5	ATTR_FACEGROUP_ID	Facegroup ID
† 6	ATTR_TEXCOORD	Texture coordinate
6..14	ATTR_RESERVED	Reserved
15	ATTR_UNSPECIFIED	Unspecified

25 **[0055]** In this embodiment, a new `bmsps_mesh_attribute_type_id` is introduced. The new `bmsps_mesh_attribute_type_id` is an identifier for texture coordinate. The texture coordinate is a two-component and/or three-component attribute providing information to mapped pixels from a

texture to a 3D mesh geometry. Two component texture coordinates are commonly used in the practice.

[0056] § 2. Some embodiments provide a reference to an attribute through the use of an attribute index. A basemesh may have several attributes. The attribute provides additional information about the meshes. The additional information corresponds to non-position information. Basemesh attributes are referenced by the V3C extension for V-DMC. The referred basemesh attributes are used by the V3C framework for processes such as reconstruction, decoding of other components, etc. Similarly, the attributes from the static mesh bitstream are referenced by the basemesh codec. The basemesh codec uses the static mesh attributes for reconstruction, decoding of other components, or other purposes.

[0057] FIG. 7 illustrates indexing of Mesh attributes from different levels in the V-DMC bitstream. As shown in the example in FIG. 7, the V-DMC extension of a V3C codec references two of the three attributes of the basemesh codec. Further, the basemesh codec references three of the four attributes of the static mesh codec.

[0058] In an example embodiment, the referencing is performed by the upper structure using an index of an attribute. The index of the attribute corresponds to the position of the attribute in the attribute array of the underlying codec.

[0059] In an example embodiment of a mesh encoding method, an encoded static mesh 702 is obtained. The encoded static mesh 702 includes a plurality of static mesh attributes 704, 706, 708, 710. A basemesh 712 is encoded based on the encoded static mesh 702 and motion information. The encoded basemesh includes lower-level mapping information, such as information associated with attributes 714, 716, 718 that associates a lower-level attribute identifier with each static mesh attribute in at least a subset of the static mesh attributes. In the embodiment of FIG. 7, the lower-level attribute identifier may be an index. For example, the basemesh mapping information 714 may include an index "0" indicating that it is associated with the first static mesh attribute 704, the basemesh mapping information 716 may include an index "1" indicating that it is associated with the second static mesh attribute 706, and the basemesh mapping information 718 may include an index "3" indicating that it is associated with the fourth static mesh attribute 710. As seen in this example, not all attributes in the encoded static mesh 702 are necessarily referenced in the encoded basemesh; for example, static mesh attribute 708 is not referenced in the mapping information of the encoded static mesh 702. An example encoding method further includes encoding a dynamic mesh 720 based on the encoded basemesh 712 and atlas information. The encoded dynamic mesh 720 includes upper-level mapping information for attributes 722, 724 that associates each of a plurality of upper-level attributes with a respective one of the lower-level attribute identifiers. For example, in FIG. 7, the mapping information of dynamic mesh attribute 722 includes an index (e.g. a "0") identifying basemesh attribute 714 (which in turn is associated with static mesh attribute 704); and the

mapping information of dynamic mesh attribute 724 includes an index (e.g. a “2”) identifying basemesh attribute 718 (which in turn is associated with static mesh attribute 710).

[0060] In an example embodiment of a mesh decoding method, an encoded dynamic mesh 720 is obtained. The encoded dynamic mesh includes an encoded basemesh 712, atlas information, and upper-level mapping information for attributes 722, 724. The upper-level mapping information associates a respective lower-level attribute identifier with each of a plurality of upper-level attributes. For example, the mapping information of dynamic mesh attribute 722 includes an index (e.g. a “0”) identifying basemesh attribute 714, and the mapping information of dynamic mesh attribute 724 includes an index (e.g. a “2”) identifying basemesh attribute 718. The encoded basemesh 712 is decoded. The encoded basemesh 712 includes an encoded static mesh 702, motion information, and lower-level mapping information for attributes 714, 716, 718. The static mesh includes a plurality of static mesh attributes 704, 706, 708, 710. The lower-level mapping information associates a lower-level attribute identifier with each of a plurality of the static mesh attributes. The lower-level attribute identifier may be an index. For example, the basemesh mapping information for attribute 714 may include an index “0” indicating that it is associated with the first static mesh attribute 704, the basemesh mapping information for attribute 716 may include an index “1” indicating that it is associated with the second static mesh attribute 706, and the basemesh mapping information for attribute 718 may include an index “3” indicating that it is associated with the fourth static mesh attribute 710. The encoded static mesh 702 is decoded, including decoding at least a first static mesh attribute, such as attribute 704 and/or 710, wherein the first static mesh attribute maps to at least one of the upper-level attributes (in this example, attribute 722 or 724) through the lower-level mapping information and the upper-level mapping information.

[0061] § 2.a. Some embodiments operate to identify the index of basemesh attributes in the V-DMC stream. A referencing system according to some embodiments provides the mechanism to pick a set of attributes from underlying sub-bitstream/sub-sub bitstream as well as to allow the use of multiple of the same attribute types. In some embodiments, the syntax of the `vps_vdmc_extension` may be modified with the addition of the lines marked with a dagger (†) as follows:

30

vps_vdmc_extension (numBytesInV3CPayload) {	Descriptor
for(j=0; j<atlas_count_minus1+1; j++){	
vps_ext_bmesh_data_attribute_count [j]	u(8)
for(i = 0; i < vps_ext_bmesh_data_attribute_count[j]; i++) {	
vps_ext_bmesh_attribute_type [j][i]	u(4)
† vps_ext_bmesh_attribute_index [j][i]	u(4)
† ...	
}	
...	
}	
}	

[0062] In some examples, the semantics of **vps_ext_bmesh_attribute_index**[j][i] may be as follows:

- 5 **vps_ext_bmesh_attribute_index**[j][i] specifies the index of the i-th attribute in the array of attributes signaled through the basemesh for the i-th attribute type.

[0063] § 2.b. Some embodiments operate to identify the index of the static mesh attribute in the basemesh bitstream. In the basemesh SPS, an **attribute_index/property_index** is added to the attributes in the static mesh codec. The **attribute_index** may refer to the implicit id of the encoded attribute (implicit id based on the order of encoding). In some embodiments, the syntax of the **bmesh_sequence_parameter_set_rbsp** may be modified with the addition of the lines marked with a dagger (†) as follows:

bmesh_sequence_parameter_set_rbsp () {	Descriptor
...	
bmsps_mesh_attribute_count	u(7)
for(i = 0; i < bmsps_mesh_attribute_count; i++) {	
bmsps_mesh_attribute_type_id [i]	u(4)
† bmsps_mesh_attribute_index [i]	u(8)?
...	
}	
...	

[0064] In some examples, the semantics of **bmsps_mesh_attribute_index**[i] may be as follows:

- 15 **bmsps_mesh_attribute_index**[i] specifies the index of the i-th attribute in the array of attributes in the mesh for the i-th attribute type.

20 [0065] In some embodiments, an encoder encodes or otherwise obtains an encoded static mesh, where the encoded static mesh including a plurality of static mesh attributes. The encoder encodes a basemesh based on the encoded static mesh and motion information. The encoded basemesh includes lower-level mapping information that assigns a lower-level attribute identifier to each static mesh attribute in at least a subset of the static mesh attributes. For example, a data structure such as **bmesh_sequence_parameter_set_rbsp** may be used to associate each lower-

level attribute identifier, which may be an index i , with a corresponding static mesh attribute indicated by `bmsps_mesh_attribute_index[i]`. (The value of i may be implicit, e.g. it may be inferred by the ordering of the values `bmsps_mesh_attribute_index[i]` in the bitstream.) The encoder also encodes a dynamic mesh based on the encoded basemesh and atlas information.

- 5 The encoded dynamic mesh includes upper-level mapping information that associates each upper-level attribute with a respective lower-level attribute identifier in at least a subset of the lower-level attribute identifiers. For example, a data structure such as `vps_vdmc_extension` may include information associating each upper-level attribute (which may be associated with an index i), with a corresponding lower-level attribute identifier `vps_ext_bmesh_attribute_index[j][i]`. (The value of i may be implicit, e.g. it may be inferred by the ordering of the values `vps_ext_bmesh_attribute_index[j][i]` in the bitstream.) Different values of j may be associated with different atlases.

- [0066]** In some embodiments, a decoder obtains an encoded dynamic mesh, e.g. in a received bitstream. The encoded dynamic mesh includes an encoded basemesh, atlas information, and upper-level mapping information that associates each of a plurality of upper-level attributes with a respective lower-level attribute identifier. For example, a data structure such as `vps_vdmc_extension` may include information associating each upper-level attribute (which may be identified by an index i), with a corresponding lower-level attribute identifier `vps_ext_bmesh_attribute_index[j][i]`. (The value of i may be implicit, e.g. it may be inferred by the ordering of the values `vps_ext_bmesh_attribute_index[j][i]` in the bitstream.) Different values of j may be associated with different atlases. The decoder decodes the encoded basemesh, where the encoded basemesh includes an encoded static mesh, motion information, and lower-level mapping information. The static mesh includes a plurality of static mesh attributes, and the lower-level mapping information assigns a lower-level attribute identifier to each of a plurality of the static mesh attributes. For example, a data structure such as `bmesh_sequence_parameter_set_rbsp` may be used to associate each lower-level attribute identifier, which may be an index i , with a corresponding static mesh attribute indicated by `bmsps_mesh_attribute_index[i]`. (The value of i may be implicit, e.g. it may be inferred by the ordering of the values `bmsps_mesh_attribute_index[i]` in the bitstream.) The decoder decodes the encoded static mesh, including decoding at least a first static mesh attribute, wherein at least the first static mesh attribute has an assigned first lower-level attribute identifier, and the first lower-level attribute identifier is associated with at least a first upper-level attribute. (In a case where a first static mesh attribute has an assigned first lower-level attribute identifier, and where that lower-level attribute identifier is associated with an upper-level attribute, the first static mesh attribute may be described as mapping to the upper-level attribute.)

[0067] It may be noted that the term “upper-level” refers in some embodiments to the VDMC level, although the principles described here are not limited to any specific implementation or

nomenclature for mesh coding. Similarly, the term “lower-level” refers in some embodiments to the basemesh level, without limitation as to any specific coding standard.

[0068] § 3. Some embodiments perform referencing through the use of unique attribute identifiers. In an example embodiment, instead of referencing by an index position, unique attribute identifiers (UAI) may be used. UAI is mechanism that may be used to refer an attribute uniquely. In some embodiments, this functionality is employed for attributes in the basemesh bitstream as well.

[0069] FIG. 8 illustrates mapping of mesh attributes via unique attribute identifiers from different levels in the V-DMC bitstream. As shown in the example in FIG. 8, the V-DMC extension of a V3C codec references two of the three attributes of the basemesh codec via basemesh UAI. Further the basemesh codec references three of the four attributes of the static mesh codec via mesh UAI. In this embodiment, the referencing is performed by the upper structure using an UAI of an attribute codec in the lower levels.

[0070] § 3.a. Some embodiments operate by identifying the attribute in the basemesh bitstream via a unique identifier in the V3C codec. At the V3C V-DMC extension, a reference is made to a basemesh attribute by coding a unique arbitrary identifier for each attribute. In case there are multiple attributes of different types, each instance is also coded with its respective UAI. The UAI allows for the identification of an attribute item in the basemesh attribute list. The identifier may be kept arbitrary to allow flexibility to not have strict mapping defined for UAI to an attribute. In some embodiments, the syntax of the vps_vdmc_extension may be modified with the addition of the lines marked with a dagger (†) as follows:

vps_vdmc_extension (numBytesInV3CPayload) {	Descriptor
for(j=0; j<atlas_count_minus1+1; j++){	
vps_ext_bmesh_data_attribute_count[j]	u(8)
for(i = 0; i < vps_ext_bmesh_data_attribute_count[j]; i++) {	
vps_ext_bmesh_attribute_type[j][i]	u(4)
† vps_ext_bmesh_attribute_unique_id[j][i]	u(4)
† ...	
}	
...	
}	
}	

[0071] In some examples, the semantics of vps_ext_bmesh_attribute_unique_id[j][i] may be as follows:

vps_ext_bmesh_attribute_unique_id[j][i] specifies the unique attribute identifier for i-th attribute coded in the basemesh bitstream for the i-th attribute type in the atlas with atlasID j.

[0072] In order to support this functionality, in some embodiments, the currently specified basemesh codec in ISO/IEC 23090-29 is modified by adding a syntax element to the basemesh

bitstream which specifies a UAI for each attribute present in the basemesh bitstream. For example, a syntax element named `bmsps_attribute_unique_id` may be introduced. The syntax element specifies an arbitrary unique identifier for each attribute in basemesh bitstream. In some embodiments, the syntax of the `bmesh_sequence_parameter_set_rbsp` may be modified with the addition of the lines marked with a dagger (†) as follows:

<code>bmesh_sequence_parameter_set_rbsp() {</code>	Descriptor
<code>...</code>	
<code> bmsps_mesh_attribute_count</code>	<code>u(7)</code>
<code> for(i = 0; i < bmsps_mesh_attribute_count; i++) {</code>	
<code> † bmsps_attribute_unique_id [i]</code>	<code>U(8)</code>
<code> bmsps_mesh_attribute_type_id[i]</code>	<code>u(4)</code>
<code> ...</code>	
<code> }</code>	
<code>...</code>	

[0073] In some examples, the semantics of `bmsps_attribute_unique_id` may be as follows:

`bmsps_attribute_unique_id` specifies the unique attribute identifier for i-th attribute coded in the mesh bitstream for the i-th attribute type.

[0074] § 3.b. Some embodiments operate to identify the attribute in the mesh bitstream via unique identifier in the basemesh codec. In some embodiments, the syntax of the `bmesh_sequence_parameter_set_rbsp` may be modified with the addition of the lines marked with a dagger (†) as follows:

<code>bmesh_sequence_parameter_set_rbsp() {</code>	Descriptor
<code>...</code>	
<code> bmsps_mesh_attribute_count</code>	<code>u(7)</code>
<code> for(i = 0; i < bmsps_mesh_attribute_count; i++) {</code>	
<code> bmsps_mesh_attribute_type_id[i]</code>	<code>u(4)</code>
<code> † bmsps_mesh_attribute_unique_id[i]</code>	<code>u(8)</code>
<code> ...</code>	<code>u(1)</code>
<code> }</code>	
<code>...</code>	

[0075] In some examples, the semantics of `bmsps_mesh_attribute_unique_id[ i ]` may be as follows:

`bmsps_mesh_attribute_unique_id[ i ]` specifies the unique identifier of the i-th attribute coded in the mesh bitstream for the i-th attribute type.

[0076] In some embodiments, an encoder encodes or otherwise obtains an encoded static mesh, where the encoded static mesh including a plurality of static mesh attributes. The encoder encodes a basemesh based on the encoded static mesh and motion information. The encoded

basemesh includes lower-level mapping information that assigns a lower-level attribute identifier to each static mesh attribute in at least a subset of the static mesh attributes. For example, a data structure such as `bmesh_sequence_parameter_set_rbsp` may be used to associate each lower-level attribute identifier, which may be a basemesh-level arbitrary unique attribute identifier such as `bmsps_attribute_unique_id`, with a corresponding static mesh attribute indicated by a static mesh level arbitrary unique identifier such as `bmsps_mesh_attribute_unique_id`. The encoder also encodes a dynamic mesh based on the encoded basemesh and atlas information. The encoded dynamic mesh includes upper-level mapping information that assigns each upper-level attribute (which may use an index *i* as an upper-level attribute identifier) to a respective lower-level attribute identifier in at least a subset of the lower-level attribute identifiers. For example, a data structure such as `vps_vdmc_extension` may include information associating each upper-level attribute (which may be identified by an index *i*), with a corresponding lower-level attribute identifier, which may be a basemesh-level arbitrary unique attribute identifier such as `vps_ext_bmesh_attribute_unique_id[j][i]`. (The value of *i* may be implicit, e.g. it may be inferred by the ordering of the values `bmsps_mesh_attribute_index[i]` in the bitstream.)

[0077] In some embodiments, a decoder obtains an encoded dynamic mesh, e.g. in a received bitstream. The encoded dynamic mesh includes an encoded basemesh, atlas information, and upper-level mapping information that assigns each upper-level attribute to a respective lower-level attribute identifier. For example, a data structure such as `vps_vdmc_extension` may include information associating each upper-level attribute (which may be identified by an index *i*), with a corresponding lower-level attribute identifier, which may be a basemesh-level arbitrary unique attribute identifier such as `vps_ext_bmesh_attribute_unique_id[j][i]`. (The value of *i* may be implicit, e.g. it may be inferred by the ordering of the values `bmsps_mesh_attribute_index[i]` in the bitstream.) The decoder decodes the encoded basemesh, where the encoded basemesh includes an encoded static mesh, motion information, and lower-level mapping information. The static mesh includes a plurality of static mesh attributes, and the lower-level mapping information assigns a lower-level attribute identifier to each of a plurality of the static mesh attributes. For example, a data structure such as `bmesh_sequence_parameter_set_rbsp` may be used to associate each lower-level attribute identifier, which may be a basemesh-level arbitrary unique attribute identifier such as `bmsps_attribute_unique_id`, with a corresponding static mesh attribute indicated by a static mesh level arbitrary unique identifier such as `bmsps_mesh_attribute_unique_id`. The decoder decodes the encoded static mesh, including decoding at least a first static mesh attribute, wherein at least the first static mesh attribute has an assigned first lower-level attribute identifier, and the first lower-level attribute identifier is associated with at least a first upper-level attribute.

[0078] § 4. Some embodiments use a unique attribute identifier for referring to the static mesh and indexing for referring to the basemesh codec. Such embodiments may be implemented as a

combination of Sections 2 and 3. A potential advantage of such embodiments as compared to embodiments that use unique identifiers at both levels is that attributes in V3C can follow an index based referencing mechanism for basemesh attributes. This reduces the need to introduce a unique identifier for basemesh attributes.

5 [0079] On the other hand, the static mesh codec (which may be an MPEG defined, or non-MPEG defined static mesh codec such as Draco) may provide additional information about the attribute such as a unique identifier associated with each attribute.

[0080] FIG. 9 illustrates mapping of attributes between the V3C extension and the basemesh codec via indexing and mapping of attribute between basemesh codec and static mesh codec via
10 UAI. As shown in the example in FIG. 9, the V-DMC extension of a V3C codec references two of the three attributes of the basemesh codec via their respective indexes. Further the basemesh codec references three of the four attributes of the static mesh codec via mesh UAI.

[0081] § 4.a. Some embodiments identify the attribute in the basemesh bitstream via an index in the V3C codec. In some embodiments, the syntax of the vps_vdmc_extension may be modified
15 with the addition of the lines marked with a dagger (†) as follows:

		Descriptor
vps_vdmc_extension (numBytesInV3CPayload) {		
for(j=0; j<atlas_count_minus1+1; j++){		
...		
for(i = 0; i < vps_ext_bmesh_data_attribute_count[j]; i++) {		
vps_ext_bmesh_attribute_type [j][i]		u(4)
† vps_ext_bmesh_attribute_index [j][i]		u(4)
...		
}		
...		
}		
}		

[0082] In some examples, the semantics of vps_ext_bmesh_attribute_index may be as follows:

20 **vps_ext_bmesh_attribute_index** specifies the index of the i-th attribute coded in the basemesh bitstream of i-th attribute type for atlas with atlasID j.

[0083] § 4.b. Some embodiments operate to identify an attribute in the mesh bitstream via unique identifier in basemesh codec. In some embodiments, the syntax of the bmesh_sequence_parameter_set_rbsp may be modified with the addition of the lines marked
25 with a dagger (†) as follows:

bmesh_sequence_parameter_set_rbsp() {	Descriptor
...	
bmsps_mesh_attribute_count	u(7)
for(i = 0; i < bmsps_mesh_attribute_count; i++) {	
bmsps_mesh_attribute_type_id[i]	u(4)
† bmsps_mesh_attribute_unqieue_id[i]	u(8)
...	
}	
...	

[0084] In some examples, the semantics of **bmsps_mesh_attribute_unique_id** may be as follows:

- 5 **bmsps_mesh_attribute_unique_id** specifies the unique identifier of the i-th attribute coded in the mesh bitstream.

[0085] In some embodiments, an encoder encodes or otherwise obtains an encoded static mesh, where the encoded static mesh including a plurality of static mesh attributes. The encoder encodes a basemesh based on the encoded static mesh and motion information. The encoded basemesh includes lower-level mapping information that assigns a lower-level attribute identifier to each static mesh attribute in at least a subset of the static mesh attributes. For example, a data structure such as **bmesh_sequence_parameter_set_rbsp** may be used to associate each lower-level attribute identifier, which may be an index i, with a corresponding static mesh attribute indicated by **bmsps_mesh_attribute_index[i]**. (The value of i may be implicit, e.g. it may be inferred by the ordering of the values **bmsps_mesh_attribute_index[i]** in the bitstream.) The encoder also encodes a dynamic mesh based on the encoded basemesh and atlas information. The encoded dynamic mesh includes upper-level mapping information that assigns each upper-level attribute to a respective lower-level attribute identifier in at least a subset of the lower-level attribute identifiers. For example, a data structure such as **vps_vdmc_extension** may include information associating each upper-level attribute (which may be identified by an index i) with a corresponding lower-level attribute identifier, which may be a basemesh-level arbitrary unique attribute identifier such as **vps_ext_bmesh_attribute_unique_id[j][i]**. (The value of i may be implicit, e.g. it may be inferred by the ordering of the values **bmsps_mesh_attribute_index[i]** in the bitstream.)

[0086] In some embodiments, a decoder obtains an encoded dynamic mesh, e.g. in a received bitstream. The encoded dynamic mesh includes an encoded basemesh, atlas information, and upper-level mapping information that assigns to each of a plurality of upper-level attribute identifiers a respective lower-level attribute identifier. For example, a data structure such as **vps_vdmc_extension** may include information associating each upper-level attribute, which may be identified by an index i, with a corresponding lower-level attribute identifier, which may be a

basemesh-level arbitrary unique attribute identifier such as
 vps_ext_bmesh_attribute_unique_id[j][i]. (The value of i may be implicit, e.g. it may be inferred
 by the ordering of the values bmsps_mesh_attribute_index[i] in the bitstream.) The decoder
 decodes the encoded basemesh, where the encoded basemesh includes an encoded static
 5 mesh, motion information, and lower-level mapping information. The static mesh includes a
 plurality of static mesh attributes, and the lower-level mapping information assigns a lower-level
 attribute identifier to each of a plurality of the static mesh attributes. For example, a data structure
 such as bmesh_sequence_parameter_set_rbsp may be used to associate each lower-level
 attribute identifier, which may be an index i, with a corresponding static mesh attribute indicated
 10 by bmsps_mesh_attribute_index[i]. (The value of i may be implicit, e.g. it may be inferred by the
 ordering of the values bmsps_mesh_attribute_index[i] in the bitstream.) The decoder decodes
 the encoded static mesh, including decoding at least a first static mesh attribute, wherein at least
 the first static mesh attribute has an assigned first lower-level attribute identifier, and the first
 lower-level attribute identifier is associated with an upper-level attribute.

15 **[0087] § 5.** Some embodiments operate to enable implicit attributes ordering or explicit
 attributes ordering. In some embodiments, the V3C extension contains references to the
 attributes in the basemesh bitstream. The attributes described in the V3C extension have the
 same order in terms of attribute type as the attributes are ordered in the terms of attribute types
 in the basemesh bitstream. For example, in the V3C extension, attributes are described in the
 20 following order: ATTR_NORMAL, ATTR_REFLECTANCE, ATTR_TRANSPARENCY. In the
 basemesh bitstream, the attributes are described in the following order: ATTR_NORMAL,
 ATTR_REFLECTANCE, ATTR_TRANSPARENCY. Such a V-DMC bitstream may not require
 signaling of a mapping between the attributes between different levels of V-DMC hierarchy. This
 is because the Nth element in the V3C extension attribute array is referencing the Nth element in
 25 the basemesh bitstream with the same attribute type where maximum value for N is
 bmsps_mesh_attribute_count. There may be a scenario where V3C extension references fewer
 attributes than the number given by, bmsps_mesh_attribute_count but the attributes are ordered
 in the V3C extension bitstream and basemesh bitstream identical in terms of attribute types.

[0088] § 5.a. Some embodiments operate to signal an explicit mapping for the V3C extension.
 30 In cases where the ordering of the attributes differs in V3C extension and basemesh codec, there
 is a need to explicitly refer to IDs (either using an index as shown in Section 2.a or a UAI as
 shown in Section 3.a) at the V-DMC extension level. In some embodiments, the syntax of the
 vps_vdmc_extension may be modified with the addition of the lines marked with a dagger (†) as
 follows:

vps_vdmc_extension (numBytesInV3CPayload) {	Descriptor
for(j=0; j<atlas_count_minus1+1; j++){	
...	
† vps_ext_bmesh_data_attribute_explicit_mapping_flag[j]	u (1)

for(i = 0; i < vps_ext_bmesh_data_attribute_count[j]; i++) {	
...	
vps_ext_bmesh_data_attribute_type[j][i]	u(4)
† if (vps_ext_bmesh_data_attribute_explicit_mapping_flag) {	
† vps_ext_bmesh_data_attribute_index[j][i]	u(4)
† }	
...	
}	
...	
}	

[0089] In some examples, the semantics of

vps_ext_bmesh_data_attribute_explicit_mapping_flag[j] may be as follows:

5

vps_ext_bmesh_data_attribute_explicit_mapping_flag[j] specifies whether mapping index for attributes is required to refer to attributes in the basemesh codec for atlas with atlasID j.

10 [0090] In some examples, the semantics of **vps_ext_bmesh_data_attribute_index** may be as follows:

vps_ext_bmesh_data_attribute_index specifies the index of the i-th attribute coded in the basemesh bitstream of i-th attribute type for atlas with atlasID j.

15 [0091] Some embodiments provide an explicit mapping via UAI. In some embodiments, the syntax of the **vps_vdmc_extension** may be modified with the addition of the lines marked with a dagger (†) as follows:

vps_vdmc_extension (numBytesInV3CPayload) {	Descriptor
for(j=0; j<atlas_count_minus1+1; j++){	
...	
† vps_ext_bmesh_data_attribute_explicit_mapping_flag[j]	u (1)
for(i = 0; i < vps_ext_bmesh_data_attribute_count[j]; i++) {	
...	
vps_ext_bmesh_data_attribute_type[j][i]	u(4)
† if (vps_ext_bmesh_data_attribute_explicit_mapping_flag) {	
† vps_ext_bmesh_data_attribute_unique_id[j][i]	u(4)
† }	
...	
}	
...	
}	

20 [0092] In some examples, the semantics of

vps_ext_bmesh_data_attribute_explicit_mapping_flag[j] may be as follows:

vps_ext_bmesh_data_attribute_explicit_mapping_flag[j] specifies whether mapping UAI for attributes is required to refer to attributes in the basemesh codec for atlas with atlasID j.

- 5 **[0093]** In some examples, the semantics of **vps_ext_bmesh_attribute_unique_id[j][i]** may be as follows:

10 **vps_ext_bmesh_attribute_unique_id[j][i]** specifies the unique attribute identifier for i-th attribute coded in the basemesh bitstream for the i-th attribute type in the atlas with atlasID j.

[0094] § 5.b. Some embodiments operate to provide signaling of an explicit mapping for basemesh codec attributes. Signaling may be used at the basemesh level to identify whether an explicit mapping is provided for the attributes in the mesh bitstream.

- 15 **[0095]** At the static mesh level, there can be more than one attribute with the same attribute type. Each attribute of the same type may correspond to different information. Each attribute item of the same attribute type is assigned an implicit id which is same as the order index it is in stored in the attributes list. For example, the first attribute of each type may have an implicit id set to 0, the second will have an id of 1, and so on. In some embodiments, the syntax of the
- 20 **bmesh_sequence_parameter_set_rbsp** may be modified with the addition of the lines marked with a dagger (†) as follows:

bmesh_sequence_parameter_set_rbsp() {	Descript or
...	
† bmsps_mesh_attribute_explicit_mapping_flag	U(1)
bmsps_mesh_attribute_count	u(7)
for(i = 0; i < bmsps_mesh_attribute_count; i++) {	
† If (bmsps_mesh_attribute_explicit_mapping_flag) {	
† bmsps_mesh_attribute_index [i]	U(8)
† }	
bmsps_mesh_attribute_type_id[i]	u(4)
...	
}	
...	

[0096] In some examples, the semantics of **bmsps_mesh_attribute_explicit_mapping_flag** may be as follows:

- 25 **bmsps_mesh_attribute_explicit_mapping_flag** specifies whether mapping index for attributes is required to refer to attributes in the static mesh codec.

[0097] In some examples, the semantics of **bmsps_mesh_attribute_index[i]** may be as follows:

bmsps_mesh_attribute_index[i] specifies the index of the i-th attribute in the array of attributes in the mesh for the i-th attribute type.

- 5 **[0098]** In some embodiments, the syntax of the bmesh_sequence_parameter_set_rbsp may be modified with the addition of the lines marked with a dagger (†) as follows:

bmesh_sequence_parameter_set_rbsp() {	Descript or
...	
† bmsps_mesh_attribute_explicit_mapping_flag	U(1)
bmsps_mesh_attribute_count	u(7)
for(i = 0; i < bmsps_mesh_attribute_count; i++) {	
† If (bmsps_mesh_attribute_explicit_mapping_flag) {	
† bmsps_mesh_attribute_unique_id [i]	U(8)
† }	
bmsps_mesh_attribute_type_id[i]	u(4)
...	
}	
...	

- 10 **[0099]** In some examples, the semantics of bmsps_mesh_attribute_explicit_mapping_flag may be as follows:

bmsps_mesh_attribute_explicit_mapping_flag specifies whether mapping UAI for attributes is required to refer to attributes in the static mesh codec.

- 15 **[0100]** In some examples, the semantics of bmsps_mesh_attribute_unique_id[i] may be as follows:

bmsps_mesh_attribute_unique_id[i] specifies the unique identifier of the i-th attribute coded in the mesh bitstream for the i-th attribute type.

- 20 **[0101]** § 6. Some embodiments impose constraints on attribute mapping. In some such embodiments, attributes from a V3C extension are also explicitly mapped via the mechanism listed in Sections 2.a, 3.a, 4.a, and/or 5.a. Using the mapping, attributes at the V3C extension refer to basemesh attributes which have same or near similar semantics in terms of attribute type.
- 25 For example, an attribute with vps_ext_bmesh_attribute_type_id: 0 = ATTR_TEXTURE may be mapped with a basemesh attribute with bmesh_attribute_type: 0 = ATTR_TEXTURE. Example embodiments provide constraints that prohibit mapping between attribute of different semantics between V3C extension and basemesh.

- 30 **[0102]** In the case of attribute mapping between basemesh codec and static mesh codec, some embodiments impose constraints such that the mapped attributes are equivalent. In some

embodiments, the equivalence of the attributes between basemesh codec and static mesh may be derived from the following rules:

- Equivalent semantics e.g., basemesh attribute type: MESH_ATTR_MATERIAL_ID in basemesh bitstream is functionally equivalent to ATTR_MATERIAL_ID in a static mesh
- Equivalent based on number of components: The number of components for a basemesh attribute may be equal to the number of components for a mapped attribute in the static mesh codec e.g., A TEXTURE in basemesh codec is three component attribute which is equivalent to three-component attribute COLOR in static mesh codec.
- If MESH_ATTR_GENERIC is signaled in the static mesh bitstream, then depending on the number of components the MESH_ATTR_GENERIC attribute holds, it is mapped to an equivalent attribute. That is a three-component ATTR_GENERIC attribute can be mapped to ATTR_TEXTURE in the basemesh bitstream.

[0103] In some embodiments, the equivalence may also be applicable to a non-MPEG static mesh codec for example Draco.

[0104] § 7. Some embodiments provide a decoding process for a basemesh codec. In example embodiments, a conformant decoding process for basemesh codec ensures that all the mesh attributes described in the syntax of basemesh bitstream are decodable and are available for further referencing.

[0105] In example embodiments, attributes from a static mesh which are being referenced by the basemesh bitstream are made available in the static mesh bitstream. The mesh attributes being referenced by the basemesh bitstream are decoded. In a case where the basemesh bitstream refers to all the mesh attributes, then all the attributes of the static mesh bitstream may be decoded.

[0106] In some embodiments, a decoder determines for at least one static mesh attribute whether that static mesh attribute maps to any upper-level attribute. If the static mesh attribute does not map to any upper-level attribute, the decoder may make the determination not to decode that attribute (or if it does decode the attribute, the decoder may discard the decoded values rather than storing them in memory).

[0107] In some embodiments, the basemesh bitstream may reference a subset of the mesh attributes based on different mapping mechanisms as described in Sections 2, 3, 4 and 5.

[0108] In some embodiments, the basemesh decoder may instruct the static mesh codec to decode only a subset of the mesh attributes in the mesh attribute array as described by the syntax in the basemesh bitstream. The referenced subset of decoded mesh attributes may be made available in the memory buffer.

[0109] In some embodiments, each entry of attribute in the mesh attribute is decoded and stored in memory buffer as an array-like structure. The length of the decoding array may be equal to mesh_attribute_count. In a case where the basemesh decoder instructs the static mesh decoder to decode only a subset of the mesh attribute as described by the syntax in the basemesh

bitstream, then the decoding array may be equal to `bmsps_mesh_attribute_count`. In this case, the mesh attributes are decoded in an order as they appear in the basemesh bitstream.

[0110] Example embodiments address the problem of referencing and mapping of attributes at different levels of V-DMC hierarchy. Such embodiments enable bitstream conformance for sub-bitstreams as well as providing decoder conformance.

Example system hardware.

[0111] Example embodiments of encoders and/or decoders (collectively coders) configured to implement embodiments described herein may be implemented using systems such as the system of FIG. 10. FIG. 10 is a block diagram of an example of a system in which various aspects and embodiments are implemented. System 1000 can be embodied as a device including the various components described below and is configured to perform one or more of the aspects described in this document. Examples of such devices, include, but are not limited to, various electronic devices such as personal computers, laptop computers, smartphones, tablet computers, digital multimedia set top boxes, digital television receivers, personal video recording systems, connected home appliances, and servers. Elements of system 1000, singly or in combination, can be embodied in a single integrated circuit (IC), multiple ICs, and/or discrete components. For example, in at least one embodiment, the processing and encoder/decoder elements of system 1000 are distributed across multiple ICs and/or discrete components. In various embodiments, the system 1000 is communicatively coupled to one or more other systems, or other electronic devices, via, for example, a communications bus or through dedicated input and/or output ports. In various embodiments, the system 1000 is configured to implement one or more of the aspects described in this document.

[0112] The system 1000 includes at least one processor 1010 configured to execute instructions loaded therein for implementing, for example, the various aspects described in this document. Processor 1010 can include embedded memory, input output interface, and various other circuitries as known in the art. The system 1000 includes at least one memory 1020 (e.g., a volatile memory device, and/or a non-volatile memory device). System 1000 includes a storage device 1040, which can include non-volatile memory and/or volatile memory, including, but not limited to, Electrically Erasable Programmable Read-Only Memory (EEPROM), Read-Only Memory (ROM), Programmable Read-Only Memory (PROM), Random Access Memory (RAM), Dynamic Random Access Memory (DRAM), Static Random Access Memory (SRAM), flash, magnetic disk drive, and/or optical disk drive. The storage device 1040 can include an internal storage device, an attached storage device (including detachable and non-detachable storage devices), and/or a network accessible storage device, as non-limiting examples.

[0113] System 1000 includes an encoder/decoder module 1030 configured, for example, to process data to provide an encoded video or decoded video, and the encoder/decoder module 1030 can include its own processor and memory. The encoder/decoder module 1030 represents

module(s) that can be included in a device to perform the encoding and/or decoding functions. As is known, a device can include one or both of the encoding and decoding modules. Additionally, encoder/decoder module 1030 can be implemented as a separate element of system 1000 or can be incorporated within processor 1010 as a combination of hardware and software as known to those skilled in the art.

5 [0114] Program code to be loaded onto processor 1010 or encoder/decoder 1030 to perform the various aspects described in this document can be stored in storage device 1040 and subsequently loaded onto memory 1020 for execution by processor 1010. In accordance with various embodiments, one or more of processor 1010, memory 1020, storage device 1040, and
10 encoder/decoder module 1030 can store one or more of various items during the performance of the processes described in this document. Such stored items can include, but are not limited to, the input video, the decoded video or portions of the decoded video, the bitstream, matrices, variables, and intermediate or final results from the processing of equations, formulas, operations, and operational logic.

15 [0115] In some embodiments, memory inside of the processor 1010 and/or the encoder/decoder module 1030 is used to store instructions and to provide working memory for processing that is needed during encoding or decoding. In other embodiments, however, a memory external to the processing device (for example, the processing device can be either the processor 1010 or the encoder/decoder module 1030) is used for one or more of these functions.
20 The external memory can be the memory 1020 and/or the storage device 1040, for example, a dynamic volatile memory and/or a non-volatile flash memory. In several embodiments, an external non-volatile flash memory is used to store the operating system of, for example, a television. In at least one embodiment, a fast external dynamic volatile memory such as a RAM is used as working memory for video coding and decoding operations, such as for MPEG-2 (MPEG refers
25 to the Moving Picture Experts Group, MPEG-2 is also referred to as ISO/IEC 13818, and 13818-1 is also known as H.222, and 13818-2 is also known as H.262), HEVC (HEVC refers to High Efficiency Video Coding, also known as H.265 and MPEG-H Part 2), or VVC (Versatile Video Coding, a new standard being developed by JVET, the Joint Video Experts Team).

[0116] The input to the elements of system 1000 can be provided through various input devices as indicated in block 1130. Such input devices include, but are not limited to, (i) a radio frequency (RF) portion that receives an RF signal transmitted, for example, over the air by a broadcaster,
30 (ii) a Component (COMP) input terminal (or a set of COMP input terminals), (iii) a Universal Serial Bus (USB) input terminal, and/or (iv) a High Definition Multimedia Interface (HDMI) input terminal. Other examples include composite video.

35 [0117] In various embodiments, the input devices of block 1130 have associated respective input processing elements as known in the art. For example, the RF portion can be associated with elements suitable for (i) selecting a desired frequency (also referred to as selecting a signal,

or band-limiting a signal to a band of frequencies), (ii) downconverting the selected signal, (iii) band-limiting again to a narrower band of frequencies to select (for example) a signal frequency band which can be referred to as a channel in certain embodiments, (iv) demodulating the downconverted and band-limited signal, (v) performing error correction, and (vi) demultiplexing to
5 select the desired stream of data packets. The RF portion of various embodiments includes one or more elements to perform these functions, for example, frequency selectors, signal selectors, band-limiters, channel selectors, filters, downconverters, demodulators, error correctors, and demultiplexers. The RF portion can include a tuner that performs various of these functions, including, for example, downconverting the received signal to a lower frequency (for example, an
10 intermediate frequency or a near-baseband frequency) or to baseband. In one set-top box embodiment, the RF portion and its associated input processing element receives an RF signal transmitted over a wired (for example, cable) medium, and performs frequency selection by filtering, downconverting, and filtering again to a desired frequency band. Various embodiments rearrange the order of the above-described (and other) elements, remove some of these
15 elements, and/or add other elements performing similar or different functions. Adding elements can include inserting elements in between existing elements, such as, for example, inserting amplifiers and an analog-to-digital converter. In various embodiments, the RF portion includes an antenna.

[0118] Additionally, the USB and/or HDMI terminals can include respective interface processors
20 for connecting system 1000 to other electronic devices across USB and/or HDMI connections. It is to be understood that various aspects of input processing, for example, Reed-Solomon error correction, can be implemented, for example, within a separate input processing IC or within processor 1010 as necessary. Similarly, aspects of USB or HDMI interface processing can be implemented within separate interface ICs or within processor 1010 as necessary. The
25 demodulated, error corrected, and demultiplexed stream is provided to various processing elements, including, for example, processor 1010, and encoder/decoder 1030 operating in combination with the memory and storage elements to process the datastream as necessary for presentation on an output device.

[0119] Various elements of system 1000 can be provided within an integrated housing. Within
30 the integrated housing, the various elements can be interconnected and transmit data therebetween using suitable connection arrangement 1140, for example, an internal bus as known in the art, including the Inter-IC (I2C) bus, wiring, and printed circuit boards.

[0120] The system 1000 includes communication interface 1050 that enables communication
with other devices via communication channel 1060. The communication interface 1050 can
35 include, but is not limited to, a transceiver configured to transmit and to receive data over communication channel 1060. The communication interface 1050 can include, but is not limited

to, a modem or network card and the communication channel 1060 can be implemented, for example, within a wired and/or a wireless medium.

[0121] Data is streamed, or otherwise provided, to the system 1000, in various embodiments, using a wireless network such as a Wi-Fi network, for example IEEE 802.11 (IEEE refers to the
5 Institute of Electrical and Electronics Engineers). The Wi-Fi signal of these embodiments is received over the communications channel 1060 and the communications interface 1050 which are adapted for Wi-Fi communications. The communications channel 1060 of these embodiments is typically connected to an access point or router that provides access to external networks including the Internet for allowing streaming applications and other over-the-top communications.
10 Other embodiments provide streamed data to the system 1000 using a set-top box that delivers the data over the HDMI connection of the input block 1130. Still other embodiments provide streamed data to the system 1000 using the RF connection of the input block 1130. As indicated above, various embodiments provide data in a non-streaming manner. Additionally, various embodiments use wireless networks other than Wi-Fi, for example a cellular network or a
15 Bluetooth network.

[0122] The system 1000 can provide an output signal to various output devices, including a display 1100, speakers 1110, and other peripheral devices 1120. The display 1100 of various embodiments includes one or more of, for example, a touchscreen display, an organic light-emitting diode (OLED) display, a curved display, and/or a foldable display. The display 1100 can
20 be for a television, a tablet, a laptop, a cell phone (mobile phone), or other device. The display 1100 can also be integrated with other components (for example, as in a smart phone), or separate (for example, an external monitor for a laptop). The other peripheral devices 1120 include, in various examples of embodiments, one or more of a stand-alone digital video disc (or digital versatile disc) (DVR, for both terms), a disk player, a stereo system, and/or a lighting
25 system. Various embodiments use one or more peripheral devices 1120 that provide a function based on the output of the system 1000. For example, a disk player performs the function of playing the output of the system 1000.

[0123] In various embodiments, control signals are communicated between the system 1000 and the display 1100, speakers 1110, or other peripheral devices 1120 using signaling such as
30 AV.Link, Consumer Electronics Control (CEC), or other communications protocols that enable device-to-device control with or without user intervention. The output devices can be communicatively coupled to system 1000 via dedicated connections through respective interfaces 1070, 1080, and 1090. Alternatively, the output devices can be connected to system 1000 using the communications channel 1060 via the communications interface 1050. The display 1100 and
35 speakers 1110 can be integrated in a single unit with the other components of system 1000 in an electronic device such as, for example, a television. In various embodiments, the display interface 1070 includes a display driver, such as, for example, a timing controller (T Con) chip.

[0124] The display 1100 and speaker 1110 can alternatively be separate from one or more of the other components, for example, if the RF portion of input 1130 is part of a separate set-top box. In various embodiments in which the display 1100 and speakers 1110 are external components, the output signal can be provided via dedicated output connections, including, for example, HDMI ports, USB ports, or COMP outputs.

[0125] The embodiments can be carried out by computer software implemented by the processor 1010 or by hardware, or by a combination of hardware and software. As a non-limiting example, the embodiments can be implemented by one or more integrated circuits. The memory 1020 can be of any type appropriate to the technical environment and can be implemented using any appropriate data storage technology, such as optical memory devices, magnetic memory devices, semiconductor-based memory devices, fixed memory, and removable memory, as non-limiting examples. The processor 1010 can be of any type appropriate to the technical environment, and can encompass one or more of microprocessors, general purpose computers, special purpose computers, and processors based on a multi-core architecture, as non-limiting examples.

Additional embodiments.

[0126] A method according to some embodiments comprises: obtaining an encoded static mesh, the encoded static mesh including a plurality of static mesh attributes; encoding a basemesh based on the encoded static mesh and motion information, the encoded basemesh including lower-level mapping information that assigns a lower-level attribute identifier to each static mesh attribute in at least a subset of the static mesh attributes; and encoding a dynamic mesh based on the encoded basemesh and atlas information, the encoded dynamic mesh including upper-level mapping information that assigns to each of a plurality of upper-level attributes a respective one of the lower-level attribute identifiers.

[0127] In some embodiments, each of the static mesh attributes is identified by an index. In other embodiments, each of the static mesh attributes is identified by an arbitrary unique identifier.

[0128] In some embodiments, the lower-level attribute identifier is an index. In other embodiments, the lower-level attribute identifier is an arbitrary unique identifier.

[0129] In some embodiments, the lower-level mapping information further identifies an attribute type for each of the static mesh attributes in the subset of the static mesh attributes. In some embodiments, the upper-level mapping information further identifies an attribute type associated with each of the upper-level attributes.

[0130] In some embodiments, the static mesh has a plurality of faces, and wherein at least one of the static mesh attributes is an attribute of each of the faces. In some embodiments, the static mesh has a plurality of vertices, and wherein at least one of the static mesh attributes is an attribute of each of the vertices.

[0131] A method according to some embodiments comprises: obtaining an encoded dynamic mesh, the encoded dynamic mesh including an encoded basemesh, atlas information, and upper-level mapping information that assigns a respective lower-level attribute identifiers to each of a plurality of upper-level attributes; decoding the encoded basemesh, the encoded basemesh
5 including an encoded static mesh, motion information, and lower-level mapping information, wherein the static mesh includes a plurality of static mesh attributes, and wherein the lower-level mapping information assigns a lower-level attribute identifier to each of a plurality of the static mesh attributes; and decoding the encoded static mesh, including decoding at least a first static mesh attribute, wherein the first static mesh attribute maps to at least one of the upper-level
10 attributes through the lower-level mapping information and the upper-level mapping information.

[0132] In some embodiments, at least some (or all) static mesh attributes that do not map to an upper-level attribute are not decoded, or their values may be discarded.

[0133] Some embodiments include an apparatus comprising one or more processors, the apparatus being configured to perform any of the methods described herein.

15 **[0134]** Some embodiments include at least one processor and a computer-readable medium storing instructions for performing any of the methods described herein.

[0135] Some embodiments include a computer-readable medium storing instructions for performing any of the methods described herein.

20 **[0136]** Some embodiments include a computer-readable medium storing a mesh encoded according to any of the encoding methods described herein.

[0137] Some embodiments include a signal conveying a mesh encoded according to any of the methods described herein.

[0138] Note that various hardware elements of one or more of the described embodiments are referred to as “modules” that carry out (i.e., perform, execute, and the like) various functions that
25 are described herein in connection with the respective modules. As used herein, a module includes hardware (e.g., one or more processors, one or more microprocessors, one or more microcontrollers, one or more microchips, one or more application-specific integrated circuits (ASICs), one or more field programmable gate arrays (FPGAs), one or more memory devices) deemed suitable by those of skill in the relevant art for a given implementation. Each described
30 module may also include instructions executable for carrying out the one or more functions described as being carried out by the respective module, and it is noted that those instructions could take the form of or include hardware (i.e., hardwired) instructions, firmware instructions, software instructions, and/or the like, and may be stored in any suitable non-transitory computer-readable medium or media, such as commonly referred to as RAM, ROM, etc.

35 **[0139]** Although features and elements are described above in particular combinations, one of ordinary skill in the art will appreciate that each feature or element can be used alone or in any

combination with the other features and elements. In addition, the methods described herein may be implemented in a computer program, software, or firmware incorporated in a computer-readable medium for execution by a computer or processor. Examples of computer-readable storage media include, but are not limited to, a read only memory (ROM), a random access
5 memory (RAM), a register, cache memory, semiconductor memory devices, magnetic media such as internal hard disks and removable disks, magneto-optical media, and optical media such as CD-ROM disks, and digital versatile disks (DVDs). A processor in association with software may be used to implement a radio frequency transceiver for use in a WTRU, UE, terminal, base station, RNC, or any host computer.

10

CLAIMS

1. A mesh encoding method comprising:
 - obtaining an encoded static mesh, the encoded static mesh including a plurality of static mesh attributes;
 - 5 encoding a basemesh based on the encoded static mesh and motion information, the encoded basemesh including lower-level mapping information that associates a lower-level attribute identifier with each static mesh attribute in at least a subset of the static mesh attributes; and
 - 10 encoding a dynamic mesh based on the encoded basemesh and atlas information, the encoded dynamic mesh including upper-level mapping information that associates each of a plurality of upper-level attributes with a respective one of the lower-level attribute identifiers.
2. A mesh encoder apparatus comprising one or more processors configured to perform at least:
 - obtaining an encoded static mesh, the encoded static mesh including a plurality of static mesh attributes;
 - 15 encoding a basemesh based on the encoded static mesh and motion information, the encoded basemesh including lower-level mapping information that associates a lower-level attribute identifier with each static mesh attribute in at least a subset of the static mesh attributes; and
 - 20 encoding a dynamic mesh based on the encoded basemesh and atlas information, the encoded dynamic mesh including upper-level mapping information that associates each of a plurality of upper-level attributes with a respective one of the lower-level attribute identifiers.
3. The method of claim 1, or the apparatus of claim 2, wherein each of the static mesh attributes is identified by an index.
4. The method of claim 1 or claim 3 as it depends from claim 1, or the apparatus of claim 2 or
25 claim 3 as it depends from claim 2, wherein the lower-level attribute identifier is an index.
5. The method of claim 1 or claims 3-4 as they depend from claim 1, or the apparatus of claim 2 or claims 3-4 as they depend from claim 2, wherein the lower-level mapping information further identifies an attribute type for each of the static mesh attributes in the subset of the static mesh attributes.
- 30 6. The method claim 1 or claims 3-5 as they depend from claim 1, or the apparatus of claim 2 or claims 3-5 as they depend from claim 2, wherein the upper-level mapping information further identifies an attribute type associated with each of the upper-level attributes.
7. A mesh decoding method comprising:

- obtaining an encoded dynamic mesh, the encoded dynamic mesh including an encoded basemesh, atlas information, and upper-level mapping information that associates a respective lower-level attribute identifiers with each of a plurality of upper-level attributes;
- 5 decoding the encoded basemesh, the encoded basemesh including an encoded static mesh, motion information, and lower-level mapping information, wherein the static mesh includes a plurality of static mesh attributes, and wherein the lower-level mapping information associates a lower-level attribute identifier with each of a plurality of the static mesh attributes; and
- 10 decoding the encoded static mesh, including decoding at least a first static mesh attribute, wherein the first static mesh attribute maps to at least one of the upper-level attributes through the lower-level mapping information and the upper-level mapping information.
8. A mesh decoder apparatus comprising one or more processors configured to perform at least:
- 15 obtaining an encoded dynamic mesh, the encoded dynamic mesh including an encoded basemesh, atlas information, and upper-level mapping information that associates a respective lower-level attribute identifiers with each of a plurality of upper-level attributes;
- decoding the encoded basemesh, the encoded basemesh including an encoded static mesh, motion information, and lower-level mapping information, wherein the static mesh includes a plurality of static mesh attributes, and wherein the lower-level mapping information
- 20 associates a lower-level attribute identifier with each of a plurality of the static mesh attributes; and
- decoding the encoded static mesh, including decoding at least a first static mesh attribute, wherein the first static mesh attribute maps to at least one of the upper-level attributes through the lower-level mapping information and the upper-level mapping
- 25 information.
9. The method of claim 7 or the apparatus of claim 8, further comprising determining for at least a second static mesh attribute whether the second static mesh attribute maps to any upper-level attribute, and wherein the second static mesh attribute is not decoded in response to a determination that the second static mesh attribute does not map to any upper-level attribute.
- 30 10. The method of claim 7 or claim 9 as it depends from claim 7, or the apparatus of claim 8 or claim 9 as it depends from claim 8, wherein each of the static mesh attributes is identified by an index.
11. The method of claim 7 or claims 9-10 as they depend from claim 7, or the apparatus of claim 8 or claims 9-10 as they depend from claim 8, wherein the lower-level attribute identifier is
- 35 an index.

12. The method of claim 7 or claims 9-11 as they depend from claim 7, or the apparatus of claim 8 or claims 9-11 as they depend from claim 8, wherein the lower-level mapping information further identifies an attribute type for each of the static mesh attributes in the subset of the static mesh attributes.
- 5 13. The method of claim 7 or claims 9-12 as they depend from claim 7, or the apparatus of claim 8 or claims 9-12 as they depend from claim 8, wherein the upper-level mapping information further identifies an attribute type associated with each of the upper-level attributes.
14. The method of any of claim 7 or claims 9-13 as they depend from claim 7, or the apparatus of claim 8 or claims 9-13 as they depend from claim 8, wherein the static mesh has a plurality of faces, and wherein at least one of the static mesh attributes is an attribute of each of the faces.
- 10
15. The method of any of claim 7 or claims 9-14 as they depend from claim 7, or the apparatus of claim 8 or claims 9-14 as they depend from claim 8, wherein the static mesh has a plurality of vertices, and wherein at least one of the static mesh attributes is an attribute of each of the vertices.
- 15