Spatial 3D Model (S3M) Specification

SUPERMAP SOFTWARE CO., LTD

Version 0.9

14/12/2018

1 Scope

This specification specifies the logical structure and physical storage format of 3D spatial data tiles. It is suitable for data transmission, exchange and high-performance visualization of massive and multi-source 3D spatial data in network environment and offline environment. It is applicable to different terminals (mobile devices， browsers， desktops) on 3D GIS-related applications.

2 Data stream specification

2.1 Endian specification

The binary stream storage involved in this specification, the byte order is specified as Little-Endian, that is, the low byte is discharged at the low address end of the memory.

2.2 Basic data type definition

The basic data types covered by this specification and their descriptions are shown in table 1.

Table 1 Numerical data type description

Type	Number of bytes	Ranges	Description
byte	1	[0,255]	Single byte
bool	1	0|1	Boolean
int16	2	[-32768, 32767]	Short type
uint16	2	[0, 65535]	Unsigned short
int32	4	[-2147483648, 2147483647]	Integer
uint32	4	[0, 4294967295]	Unsigned integer
float	4	$[-3.4*10^{-38}, 3.4*10^{38}]$	Single precision floating point
double	8	$[-1.7*10^{-308}, 1.7*10^{308}]$	Double precision floating point
char	1	--	Character type
wchar	2	--	Wide character type

2.3 String type

The character data types involved in this specification are described by String objects, which are encoded in Unicode. The character set is specified as UTF16, that is, each character

occupies two bytes.

```
String {
Int32        length;        //number of strings
Wchar        str[length];    //string content
}
```

3 Data Organization Structure

3.1 Logical organization of data

This specification organizes the three-dimensional data in a certain geospatial scope in tile manner, represented by the TileTreeSet object; TileTreeSetInfo is its description information, which is an overall description of the data; if the TileTreeSet is constructed based on the point, line, polygon or model dataset, There may be AttributeInfos that represent attribute description information for each data set.

Space division of three-dimensional data in a specified spatial range, each spatial division corresponding to a tree structure organization tile collection, represented by a TileTree object; IndexTree is index information of its tree structure; TileTree may have attribute data AttributeData, recorded in TileTree Attribute data for each object.

Each TileTree is subdivided step by step from top to bottom, and each space partition corresponds to one tile, represented by a Tile object. The relationship between objects is shown in Figure 1.

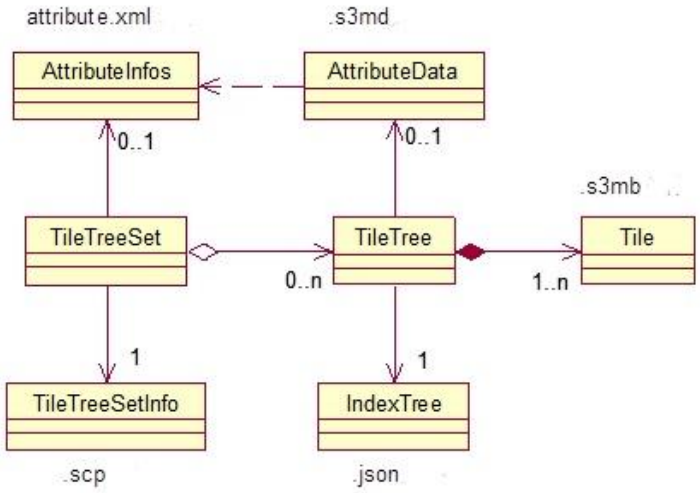

Figure 1 Data organization UML diagram

3.2 File types

The data specified in this specification mainly includes: description files, index tree files, data files, and attribute files. The file organization of each object storage is shown in Table 2.

Table 2 File organization of object storage

Object	Storage	File type	Description
TileTreeSetInfo	.scp	Description file	Description of the entire data
AttributeInfos	attribute.xml	Attribute description file	Description of each data set attribute in TileTreeSet
TileTree	Folder	Data folder	Store all data in the tile range
AttributeData	.xml	Attribute data file	Attribute data of all objects belong to the tile
IndexTree	.json	Index tree file	All PagedLOD information belong to the tile
Tile	.s3mb	data file	A S3MB file stores data in a spatial division of the LOD layer

The description file (.scp) and data folder are the basic components; the description file contains the path file name (.json) for each TileTree; The index file is a description of the tree structure of the tile data, and can obtain the bounding box of each tile file of each layer, the switching information of the LOD, the attached child node file, etc. without loading the actual data. The main role is to accelerate the efficiency of tile file retrieval; The attribute data includes one attribute description file (attribute.xml) and a .xml file storing each tile attribute data in each TileTree, optionally.

3.3 Tree structure description

All Tiles in TileTree form a tree-like logical structure. The PagedLOD describes the parent-child relationship, and the top-down transition from the coarse layer to the fine layer. Take the quadtree as an example, its division structure is shown in Figure 2.

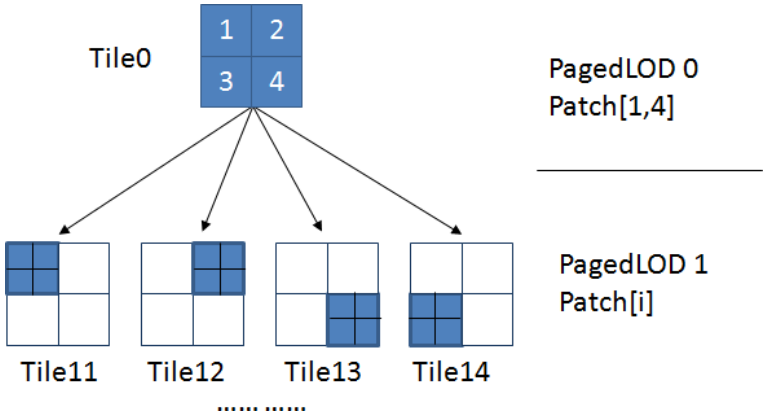

Figure 2 the structure of TileTree

Each LOD level has one or more partitions in the horizontal direction of the tree structure, each partition is represented by PagedLOD; PagedLOD identifies the LOD level of the Tile in the TileTree, and an LOD level can be composed of one or more PagedLODs.

Each PagedLOD has one or more partitions in the horizontal direction of the tree structure, and each partition is represented by a patch. Each patch has zero or one parent patch, and there are zero or more child patches. Each child patch is a space division of the parent patch; the parent and child relationship of the patch is constructed into a tree structure.

Each patch consists of zero or more data packets, represented by Geode; each Geode contains one or more entity objects (ModelEntity), and a matrix acts on all skeleton objects in the Geode.

ModelEntity includes three entity types: Skeleton, Material, and Texture, which correspond to three types of entity objects: skeleton, material, and texture. Geode records the name of the entity object contained in the data packet, so that the same entity object can be referenced by different Geodes. The UML diagram of the related object is shown in Figure 3.

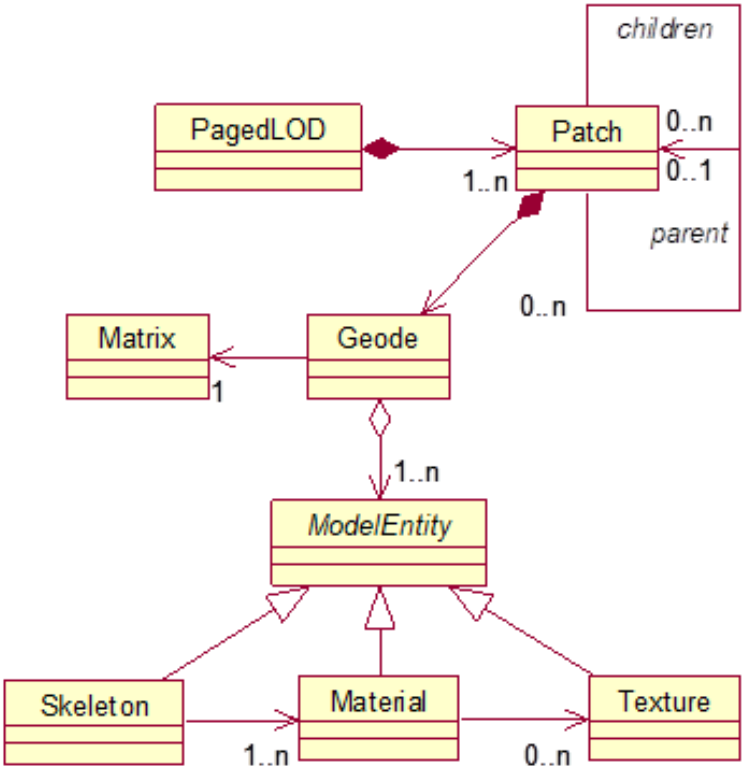

Figure 3 tree structure UML diagram

4 Data Files

4.1 s3mb file logical structure

4.1.1 The main structure

The data file is the main component of the data and consists of .s3mb (Spatial 3D Model Binary) files; each S3MB file stores three-dimensional spatial data within a spatial division of the PagedLOD. The object UML diagram is shown in Figure 4.

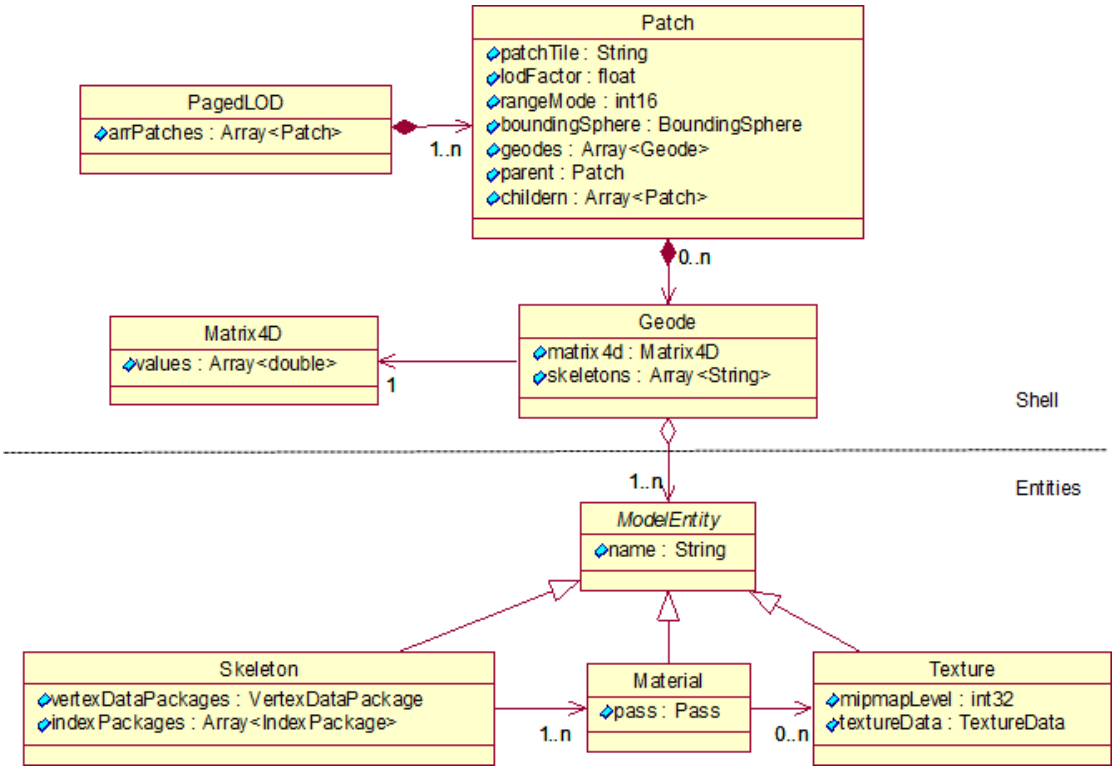

Figure 4 S3MB file storage object UML diagram

——PagedLOD: composed of Patches

- arrPatches: all patches for this Paged LOD

——Patch: contains data within the specified geo-space

- patchTile: the name of the s3mb file where the patch is located
- lodFactor: the switching factor, which is the threshold for LOD switching, used in conjunction with the switching mode
- rangeMode: Switch mode, see 6.3.2.2.
- boundingShpere: bounding shpere of the tile
- childTile: relative path to the mounted subfile
- geodes: spatial data packets
- parent: parent node
- children: an array of child nodes, that is, all patches in the attached subfile

——Geode: spatial data packet

- matrix4d: matrix acting on the skeletons in the geode packet
- skeletons: an array of skeleton names

——Matrix 4D: 4*4 matrix, row main sequence

- values: an array of 16 doubles

——ModelEntity: entity object's base class

- name: the entity name, which is the unique identifier of the entity object in TileTree

—— Skeleton: skeleton object, inherited from ModelEntity

- vertexDataPackages: vertex packet
- indexPackage: vertex index packet

——Material: Material object, inherited from ModelEntity

- pass: render channel

——Texture: texture object, inherited from ModelEntity

- mipmaplevel: the level of the texture object MipMap
- textureData: texture data

4.1.2 Skeleton objects

A Skeleton object consists of a vertex package (VertexDataPackage) and one or more vertex index packages(IndexPacakge). A vertex package includes vertex coordinates, normals, colors, texture coordinates, etc.; a vertex index package is a description of the skeleton structure construction, and each vertex index packet has one or more passes to specify how the vertices are rendered. The UML diagram of the skeleton related object is shown in Figure 5.

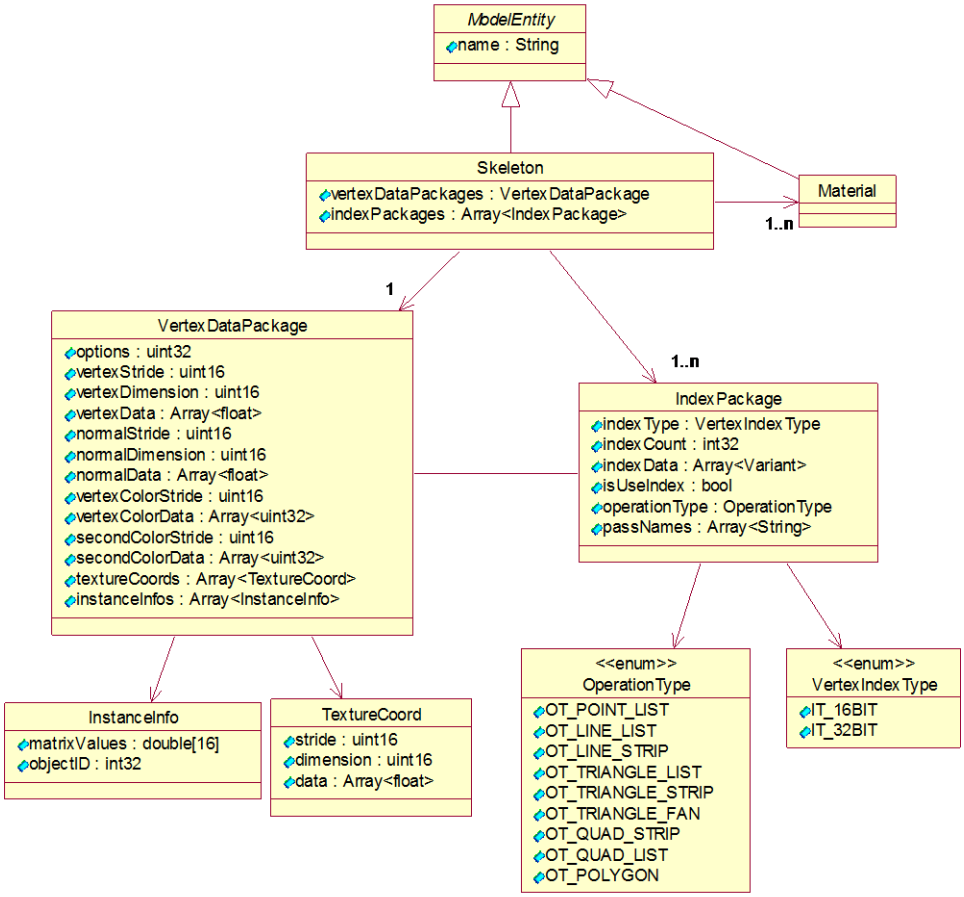

Figure 5 skeleton object UML diagram

—— VertexDataPackage: vertex packet

- options: data option information, storing extended data
- vertexStride: the offset of the vertex coordinates in the array
- vertexDimension: vertex dimension
- vertexData: an array of vertex coordinate values
- normalStride: the offset of the normal vector in the array
- normalDimension: the dimension of the normal vector
- normalData: an array of normal vector component values
- vertexColorStride: the offset of the vertex color in the array
- vertexColorData: vertex color array
- secondColorStride: second vertex color offset
- secondColorData: second vertex color array (can store object ID)
- textureCoords: texture coordinate array
- instanceInfos: instantiated information array

—— InstanceInfo: instantiation information

- matrixValues: 4*4 matrix values
- objectID: object ID

—— TextureCoord: texture coordinates

- stride: offset
- dimension: texture coordinate dimension
- data: an array of texture coordinate values

——IndexPackage: vertex index package

- indexType: vertex index type
- indexCount: the number of vertex indexes
- indexData: vertex data, depending on the vertex index type, may be a Short array or an Integer array
- isUseIndex: whether to use the index
- operationType: how the index is organized
- passNames: the name of the Pass object to use when rendering the object

——OperationType: how the index is organized

- OT_POINT_LIST: a single point
- OT_LINE_LIST: two-dot line
- OT_LINE_STRIP: linestring
- OT_TRIANGLE_LIST: Triangle
- OT_TRIANGLE_STRIP: Striped triangle
- OT_TRIANGLE_FAN: Sector triangle composition
- OT_QUAD_STRIP: Quadrilateral
- OT_QUAD_LIST: Striped quadrilateral
- OT_POLYGON: Polygon

——VertexIndexType: the data type of the vertex index

- IT_16BIT: 16-bit unsigned integer
- IT_32BIT: 32-bit unsigned integer

Generally, according to the number of vertices, IT_32BIT is adopted when the number of vertices is greater than 65535, otherwise IT_16BIT is adopted.

4.1.3 Material object

The material object is composed of Pass, and the texture object name used by the material is recorded in Pass. The UML diagram of the material related object is shown in Figure 6.

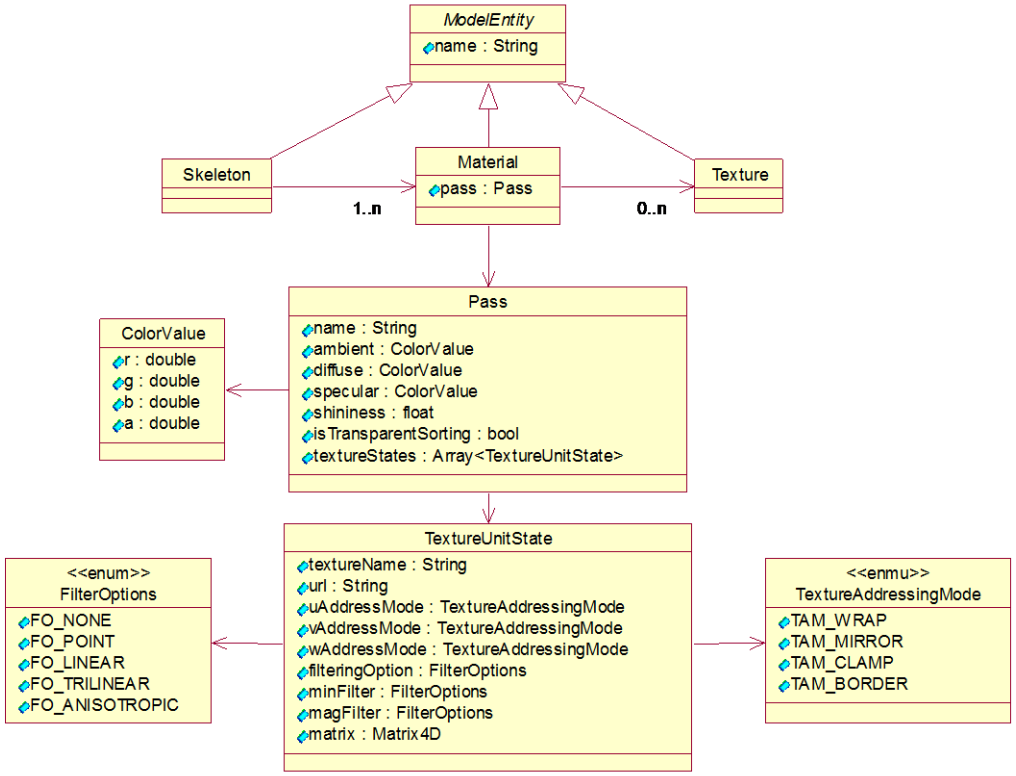

Figure 6 material object UML diagram

——Material: Material object

- pass:Pass object

——Pass:

- name: the name of the Pass
- ambient: ambient light color
- diffuse: the color of the scattered light
- specular: specular color
- shininess: reflected light color
- isTransparentSorting: Whether to use transparent sorting
- textureStates: texture information

——ColorValue: color value

- r: red component value
- g: green component value
- b: blue component value

——TextureUnitState: texture information

- textureName: texture name
- url: texture reference path
- uAddressMode: texture mode u map mode
- vAddressMode: texture coordinate v direction map mode
- wAddressMode: texture coordinate w direction mapping mode
- filteringOption: texture interpolation mode

- minFilter: the interpolation mode used when the texture is reduced
- magFilter: interpolation mode used when texture is enlarged
- matrix: texture matrix

——FilterOption:

- FO_NONE: No filtering
- FO_POINT: adjacent sampling
- FO_LINEAR: two-line filtering
- FO_TRILINEAR: Three-line filtering
- FO_ANISOTROPIC: Anisotropic filtering

—— TexureAddressingMode:

- TAM_WRAP: Copy the entire texture
- TAM_MIRROR: Symmetric flip
- TAM_CLAMP: Edge pixels to fill all texture coordinates greater than 1, edge lengthened
- TAM_BORDER: border pixels to fill all texture coordinates greater than 1, the border is elongated

4.1.4 Texture object

The texture object UML diagram is shown in Figure 7.

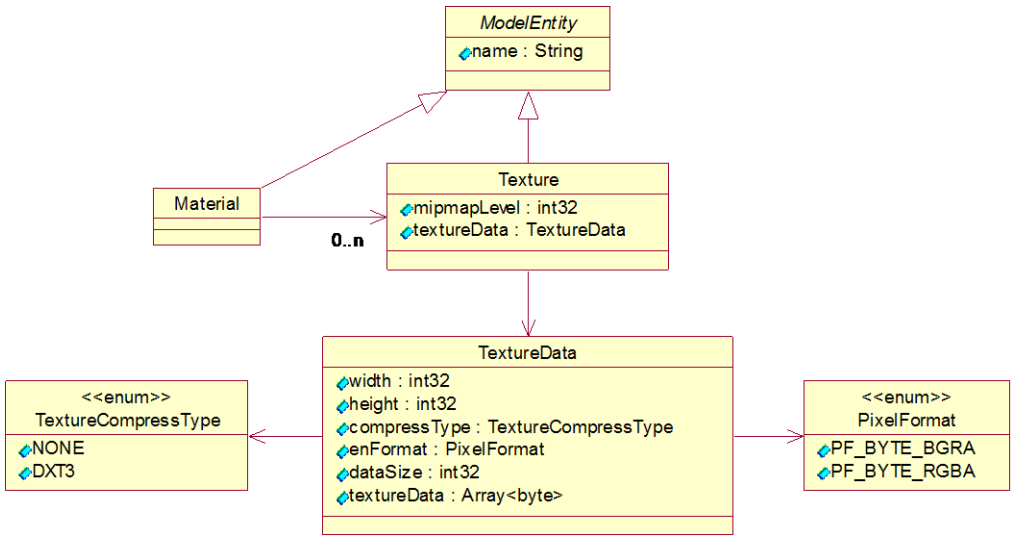

Figure 7 texture object UML diagram

—— Texture: texture object

- mipmapLevel: MipMap layer number
- textureData: texture data

—— TextureData: texture data

- width: the number of horizontal pixels
- height: vertical pixel format
- compressType: texture compression method
- enFormat: the pixel format of the texture
- dataSize: the binary stream size of the texture

- textureData: texture data binary stream

——PixelFormat: pixel format

- PF_BYTE_BGRA: BGRA format
- PF_BYTE_RGBA: RGBA format

4.2 Binary stream description of s3mb file

4.2.1 Main components of the s3mb file

The s3mb file is stored in a binary stream, as shown in Figure 4. It contains four parts: Header, Shell, Entities, and InstanceInfo:

S3MBFile{

Char header[4]; //File header, mandatory content is 'S''3''M''\0', length is 4 bytes

Shell shell; //Store PagedLOD, Patch, Geode object, the object structure diagram is shown in Figure 4.

Entities entities; //Entity data, including skeleton (Skeleton), material (Material), texture (Texutre)

InstanceInfo instanceinfo;// instantiation information

6.3.2.2 Shell binary stream description

Shell{

PagedLOD pagedLods[count];

};

PagedLOD{

Int32 numPatches; //Number of Patch objects

Patch patches[numPatches];

};

Patch{

Float fLodDistance; //LOD switching factor

RangeMode rMode; //LOD switching mode, stored as int16

BoundingSphere boundingSphere; //Bounding the ball

String strChildTile; / / relative path of the mounted subfile

Int32 numGeodes; / / number of Geode included

Geode geodes[numGeodes];

};

Eumn RangeMode{

Distance_From_EyePoint, / / switch according to the distance to the camera

Pixel_Size_OnScreen //Switch according to the pixel size projected onto the screen

};

```
BoundingSphere{
    Double x; // center point x coordinate
    Double y; // center point y coordinate
    Double z; // center point z coordinate
Double r; //the bounding sphere radius
};

Geode{
    Matrix4d matrix;
    Int32 numSkeletons;
    String skeletonNames[numSkeletons];
};

Matrix4d{ // 4*4 matrix, row main sequence
   Double values[16];
```

6.3.2.3 Entities binary stream description

```
Entities{
   Int32 numSkeletons; //skeleton
   Skeleton skeletons[numSkeletons];
   Int32 numMaterials; //material
   Material materials[numMaterials];
   Int32 numTextures; //texture
   Texture textures[numTextures];
};

Skeleton{
  String name;
  VertexDataPackage dataPack;
  Int32 numIndexPacks;
  IndexPacakge indexPacks[numIndexPacks];
};

VertexDataPackage{
  Byte reserved[4]; //reserved

  Uint32 numVertex; //vertex
  Uint16 vertexDimension;
  Uint16 vertexStride;
  Float vertexData[numVertex* vertexDimension];

  Uint32 numNormal; //normal
```

```
Uint16 normalDimension;
  Uint16 normalStride;
  Float normalData[numNormal* normalDimension];

  Int32 numVertexColor; //vertex color
  Uint16 vertexColorStride;
  Byte reserved[2];
  Uint32 vertexColorData[numVertexColor];

  Int32 numVertexSecondColor; //SecondColor
  Uint16 vertexColorSecondStride;
  Byte reserved[2];
  Uint32 vertexSecondColorData[numVertexColor];

  Uint16 numTextureCoord; //Texture coordinates
  TextureCoord textureCoords[numTextureCoord];

Uint16 numInstanceInfo; // instantiate information
  InstanceInfo instanceInfo[numInstanceInfo];
};

TextureCoord{
  Uint32 numCoods;
  Uint16 dimension;
  Uint16 stride;
  Float values[numCoods*dimension];
};

InstanceInfo{ // instantiate information
  Double values[16]; //matrix, row main sequence
  Uint32 objectID; //object ID
};

IndexPacakge{
  Uint32 numIndexes;
  IndexType enIndexType; //Save as byte
  Byte reserved;
  OperationType opType; //Save as byte
  Byte reserved;
  Variant indexData[numIndexes]; //index value
(Note:
IndexType is IT_16BIT, variant type is uint16; IndexType is IT_32BIT, variant type is uint32)
  Int32 numPass;
  String passNames[numPass];
```

```
};

IndexType{
  IT_16BIT = 0, //index value is represented by uint16
  IT_32BIT = 1 / / index value is represented by uint32
};

OperationType{
OT_POINT_LIST = 1, // single point
OT_LINE_LIST = 2, //two dot line
OT_LINE_STRIP = 3, //string
OT_TRIANGLE_LIST = 4, // triangle
OT_TRIANGLE_STRIP = 5, // strip triangle
OT_TRIANGLE_FAN = 6, // sector triangle composition
OT_QUAD_STRIP = 8, // strip quadrilateral
OT_QUAD_LIST = 9, // strip quadrilateral
OT_POLYGON = 10, //polygon
};

Material{
  String strMaterial; / / material is described by xml, stored in String type
};

Texture{
  String strName;
  Int32 numMipMapLevel;
  TextureData texData;
};

TextureData{
  Uint32 width;
  Uint32 height;
  TextureCompressType compressType; //Save as uint32
  Uint32 datasize;
  PixelFormat enPixelFormat; // stored as uint32
  Byte data[datasize];
};
TextureCompressType{
  TC_NONE = 0,
  TC_DXT3 = 14,
};
PixelFormat{
  PF_BYTE_BGRA = 12,
  PF_BYTE_RGBA = 13,
```

4.3 Attribute file

4.3.1 Composition of attribute file

The attribute file includes an attribute description file and attribute data files. The attribute description file name is specified as attribute.xml, which is in the same directory as the description file (.scp); the attribute data file name is the same as the tile data root node file name, the file extension is .s3md, and the data file (.s3mb) is at the same directory.

4.3.2 Attribute description file

The attribute description file describes the ID range and field information of each layer and is stored in xml format. The meaning of each label is shown in table 3:

Table 3 Attribute Description File Label Meaning

Tag name	Meaning
LayerInfos	Attribute description information for each layer
LayerInfo	Property description for a single layer
LayerName	Layer name
IDRange	The range of the ID that represents the maximum and minimum ID of the layer object
FieldInfos	Field information collection

The meaning of each label of the field information is shown in table 4:

Table 4 Field Information Label

Tag name	Meaning
Name	Field name
Alias	Field alias
Type	Field Type
Size	Field length
IsRequired	Field value cannot be null

4.3.3 Attribute data file

The attribute data file contains the attribute description information of each layer and each attribute value of each object and is stored in an xml format file. See Appendix D for the schema. The meaning of each label in the attribute data is shown in table 5.

Table 5 Attribute data file label meaning

Tag name	Meaning

Layer	Layer label
IDRange	The range of the ID, indicating the maximum and minimum ID of the object in the corresponding tile range
FieldInfos	Field information collection, see Table 3
Records	Set of attribute values for each object
Record	The properties of a single object
Values	Object property value collection