INTERNATIONAL STANDARD

ISO/IEC 14496-11

First edition
2005-12-15

AMENDMENT 6
2009-03-15

Information technology — Coding of audio-visual objects —

Part 11: Scene description and application engine

AMENDMENT 6

Technologies de l'information — Codage des objets audiovisuels —

Partie 11: Description de scène et moteur d'application

AMENDEMENT 6

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14496-11:2005/Amd 6:2009

Reference number
ISO/IEC 14496-11:2005/Amd.6:2009(E)

© ISO/IEC 2009

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14496-11:2005/Amd 6:2009

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2009

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Amendment 6 to ISO/IEC 14496-11:2005 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14496-11:2005/Amd 6:2009

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14496-11:2005/Amd 6:2009

Information technology — Coding of audio-visual objects —

Part 11:
Scene description and application engine

AMENDMENT 6

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14496-11:2005/Amd 6:2009

After 8.13, add the following new subclause:

8.14 Scene Partitioning

8.14.1 Overview

In 3D streaming applications, a server often holds a compressed binary representation of the whole scene data. At the time a client connects, it receives a coarse version of the environment that suits more or less its actual location and requested precision. For the rest of the navigation, refinement data will be sent according to the observer trajectory within the scene.

At this stage, two scenarios are possible. The first one is called *server-driven scenario*; in this case, the server is assumed to be able to cope with the necessary computations for deciding exactly what refinements the client needs. Usually, the client has already sent his position and some hints of what he already has in his cache. According to this information, the server extracts a subset of the compressed binary representation, using some kind of MPEG-21 gBSD file.

The second possible scenario is the so-called client-based one. In this case, it is the client task to compute and request the necessary refinement data. In a perfect world, the server would have enough capability to constantly remain in server-driven mode. But in practical applications, when the number of clients grows, often reaching several thousands of terminals, the server can not cope anymore and has to cast to the most effective clients the task of identifying the needed refinements.

Another important thing to note, also raised after our practical implementations, is that this becomes general rule when dealing with peer-to-peer applications, i.e. when terminals can arbitrarily be considered as servers as well.

While the client-driven mode reduces the amount of information to send to the server (namely the hints on the cache content), one noticeable difference is that the client does not know exactly what could or should be sent in function of his position and orientation. What was known on the server side in the server-driven mode is unknown by the client in the client-driven mode.

The schema is based on an extensible syntax, such as the AFX backchannel. The purpose of this framework is to be able to any space partitioning conception, including the most general ones, as well as the most specific. The partitioning types considered so far are:

1) *BSP*: this had already been proposed at the Fairfax meeting, but the activity had not followed up at that time by lack of support and efficient design of the node. However, the technology itself has proved to be useful for adaptive transmission and rendering of large scenes, and applies to the most arbitrary scenes, independently on the tools used to compress the objects.

2) *Cells / Portals*: another widely used representation for selective transmission / rendering of large interior scenes is the Cell / Portal paradigm. This representation is a graph in which the nodes figure the various rooms in the building and the edges denote the possible visibility from one room to another.

© ISO/IEC 2009 – All rights reserved

3) *PVS (Potentially Visible Sets)*: also very widely used for exterior scenes, the purpose of PVS is the same as Cells and Portals with the difference that areas are not related to other visible areas but instead linked to the set of objects that are visible from this area.

4) *WaveletSubdivisionSurfaces*: this is a specific partitioning design, suited to the accommodation of geometric wavelet coefficients. This is based on bounding volumes that are strongly dependent on the shape of the base mesh.

5) *FootPrints*: this is the specific design that was originally demonstrated and that showed significant gain in both bandwidth and reconstruction time.

Generic tools, such as BSP, Cells and Portals and PVS are supposed to handle portions of scenes independently of the encoding scheme. This can be used for VRML scenes, or with objects for which the partitioning does not have to have finer granularity than the object itself, namely because its encoding does not provide multiresolution.

8.14.2 Node interface

```
PROTO SpacePartition [ #%NDT=SFWorldNode,SF3DNode %COD=N
      eventIn       MF3DNode   addChildren
      eventIn       MF3DNode   removeChildren
      exposedField  MF3DNode   children       []
      exposedField  SFUrl        SPStream NULL
]{}
```

8.14.2.1 Semantics and functionality

children: this is the target node. The partitioning information may apply to the children nodes and to its descent.

SPStream: this is the stream containing the Scene Partitioning information.

NOTE The partitioning nodes obey the following criteria:

- Each partitioning node is attached to a rendered node;
- The partitioning node influences the descent of the rendered node it is attached to;
- The partitioning nodes combine themselves according to the hierarchy of the scene graph;

Figure AMD6.1 shows an example illustrating these points.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14496-11:2005/Amd 6:2009

 © ISO/IEC 2009 – All rights reserved

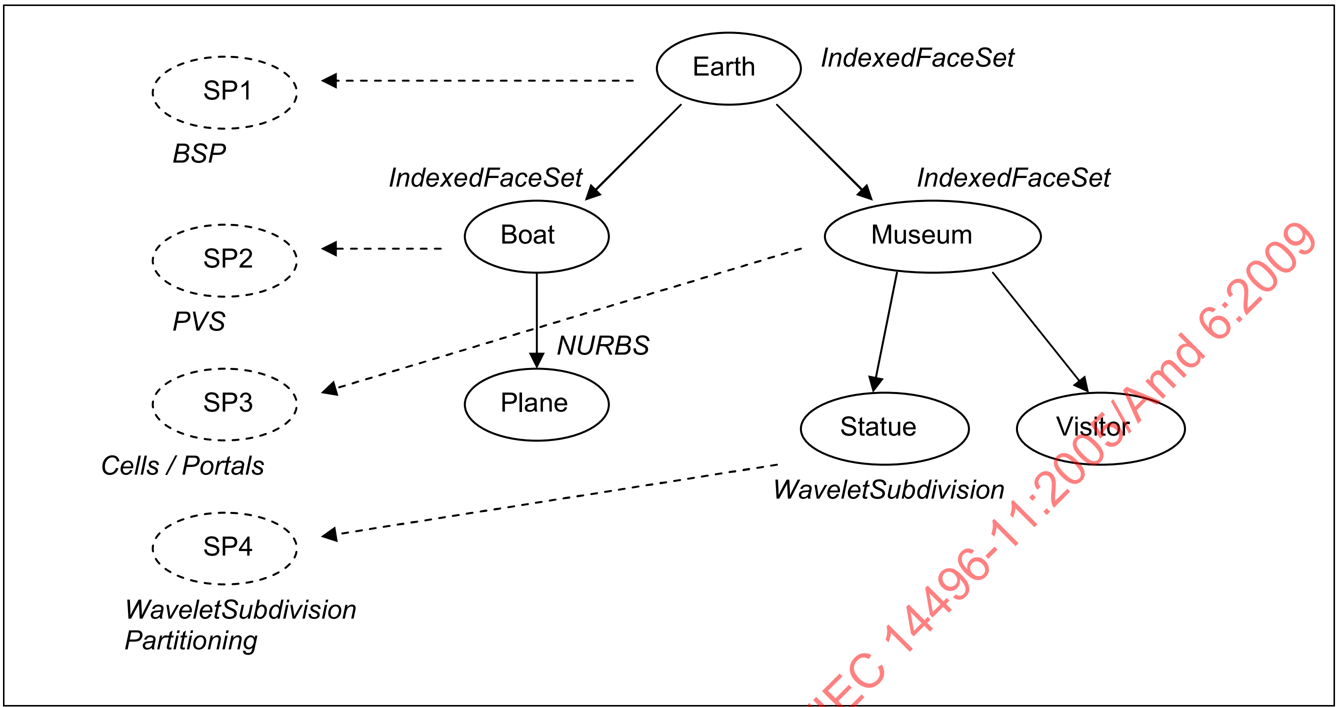


Figure AMD6.1 — example of organization of space partitioning nodes within a scene graph

In this example, one can see various space partitioning nodes (the SPs) occurring at various depth in the scene hierarchy. The type of each SP node is suited to the type of the object it is linked to. For example IndexedFaceSets representing the Earth and the Boat are partitioned using BSP and PVS. The museum, which is an interior subscene, is partitioned with Cells and Portals. The statue inside the museum, represented by WaveletSubdivisionSurfaces, is partitioned with the according declination of the node. Each SP node is dependent on every other SP node upper in the hierarchy. For instance the rendering of the statue is subject to adaptation lead by SP4, but is constrained by the visibility induced by SP3 and SP1, that are linked to parent nodes.

8.14.3 Scene Partitioning stream definition

8.14.3.1 SpacePartitionDecoderConfig

8.14.3.1.1 Syntax

```
class SpacePartitionDecoderConfig {
    int (8) DSItag;
    int (8) type;
    switch(type) {
            0: BSPDecoderConfig;
            1: CellPortalDecoderConfig;
        2: PVSDecoderConfig;
        3: SPFootprintDecoderConfig;
        4: WaveletDecoderConfig;
       )
}
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14496-11:2005/Amd 6:2009

8.14.3.1.2 Semantics

DSItag: Space Partition tag (0x0C)

type: space partition type

8.14.3.2 BSPDecoderConfig

8.14.3.2.1 Syntax

```
class BSPDecoderConfig {
      int(6) indexNbBits;
   int(6) coefNbBits;
   int(6) objCountNbBits;
   int(1) is3D;
}
```

8.14.3.2.2 Semantics

indexNbBits: number of bits used to encode BSP plane IDs

coefNbBits: number of bits used to encode BSP plane coefficients

objCountNbBits: number of bits used to encode the number of objects

is3D: identifier of the 2D (value 0) or 3D (value 1).

8.14.3.3 CellPortalDecoderConfig

8.14.3.3.1 Syntax

```
class CellPortalDecoderConfig {
   int(6) cellCountNbBits;
   int(6) totalCountNbBits;
   int(6) cellGeomNbBits;
      int(1) is3D;
}
```

8.14.3.3.2 Semantics

cellCountNbBits: number of bits used to encode number of cells in the stream

totalCountNbBits: number of bits used to encode total number of cells as well as cell IDs

cellGeomNbBits: number of bits used to encode cell geometry parameters

is3D: identifier of the 2D (value 0) or 3D (value 1).

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14496-11:2005/Amd 6:2009

 © ISO/IEC 2009 – All rights reserved

8.14.3.4 PVSDecoderConfig

8.14.3.4.1 Syntax

```
class PVSDecoderConfig {
    int(6) cellCountNbBits;
    int(6) objCountNbBits;
       int(6) pvsGeomNbBits;
}
```

8.14.3.4.2 Semantics

cellCountNbBits: number of bits used to encode the total number of cells

objCountNbBits: number of bits used to encode the total number of objects

8.14.3.5 SPFootprintDecoderConfig

8.14.3.5.1 Syntax

```
class SPFootprintDecoderConfig {
       int(8) type;
    unsigned int(5) rootChildrenRadiusNbBits;
    unsigned int(5) nbChildrenNbBits;
    unsigned int(5) nbSubTreesNbBits;
    float(32) acquisitionPrecision;
       float(32) minMetricError;
       float(32) maxMetricErrorEncodingFunction;
       unsigned int(16) nbRootChildren;
       unsigned int(5) indexNbBits;
       unsigned int(5) nbNodesInSubTreeNbBits;
       unsigned int(5) nbNodesOnFirstLevelOfSubTreeNbBits;
       unsigned int(5) nbSubTreesChildrenNbBits;
       unsigned int(5) nbNodesOnLastLevelNbBits;
       unsigned int(5) networkType;
       switch(networkType) {
        0: // no additional information;
        1: int(5) subTreeSizeNbBits;
           Int(5) geometryNodesSizeNbBits;
}
```

8.14.3.5.2 Semantics

type: type of the description structure

rootChildrenRadiusNbBits: number of bits used to decode the radius of the children (i.e. the bounding sphere)

nbChildrenNbBits: number of bits used to decode the number of hierarchical description node's children

nbSubTreesNbBits: number of bits used to decode number of sub-trees in a packet.

acquisitionPrecision: precision used during data acquisition.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14496-11:2005/Amd 6:2009

minMetricError: smallest metric error that is greater than 0.

maxMetricErrorEncodingFunction: maximum metric error used in the encoding function.

nbRootChildren: number of children nodes for current node.

indexNbBits: number of bits used to decode description node indices.

nbNodesInSubTreeNbBits: number of bits to used to decode the number of sub-tree nodes.

nbNodesOnFirstLevelOfSubTreeNbBits: number of bits used to decode the number of nodes included in the first level sub-tree.

NbSubTreesChildrenNbBits: number of bits used to decode the number of current sub-tree childrens.

nbNodesOnLastLevelNbBits: number of bits used to decode the number of nodes in the sub-tree first level.

networkType: communication type.

Type 0: client - server

Type 1: P2P

-subTreeSizeNbBits: number of bits used to decode the sub-tree size.

-geometryNodeSizeNbBits: number of bits used to decode the geometry size.

8.14.3.6 WaveletDecoderConfig

8.14.3.6.1 Syntax

```
class WaveletDecoderConfig {
    int(6) unitCountNbBits;
       int(6) faceCountNbBits;
       int(6) geomNbBits;
}
```

8.14.3.6.2 SpacePartitionNodeMessage

```
class SpacePartitionNodeMessage {
  switch(SpacePartitionDecoderConfig.type) {
    0: BSPNodeMessage;
    1: CellPortalNodeMessage;
    2: PVSNodeMessage;
    3: FootprintMessage;
    4: WaveletMessage;
  }
}
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14496-11:2005/Amd 6:2009

 © ISO/IEC 2009 – All rights reserved

8.14.3.7 BSPNodeMessage

8.14.3.7.1 Overview

BSP Message		
NbUnits	BSPUnit	BSPUnit

NbUnits : number of BSP Units defined below

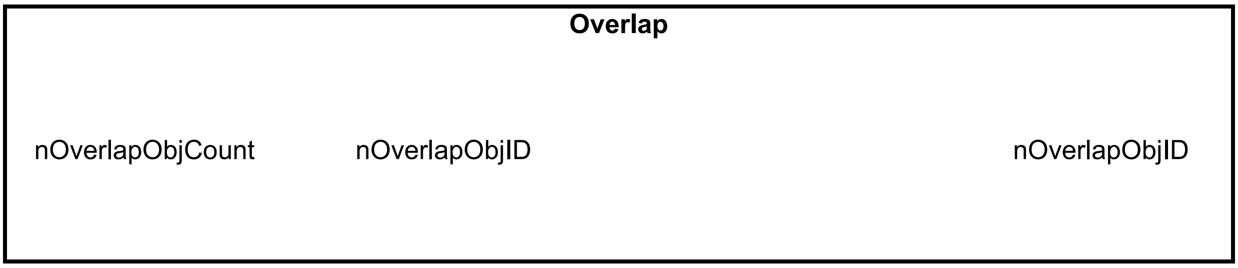
BSP Unit			
Header	Front	Overlap	Back

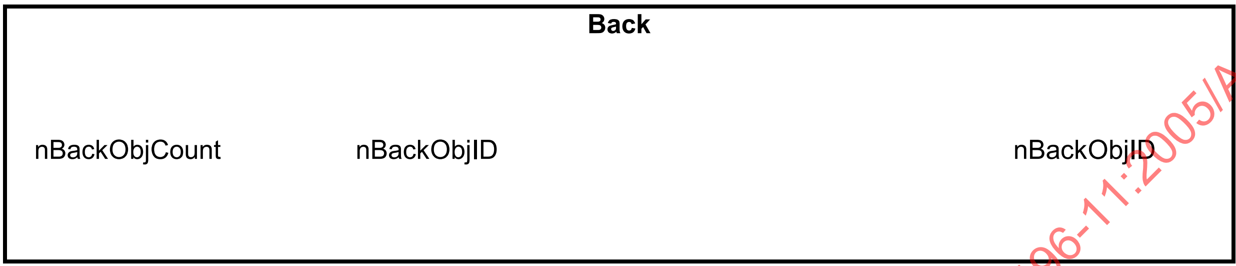
with:

Header								
nIndex	nIndexParent	A	b	c	d	nIndexFront	nIndexOverlap	nIndexBack

Front		
nFrontObjCount	nFrontObjID	nFrontObjID

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14496-11:2005/Amd 6:2009





8.14.3.7.2 Syntax

```
class BSPNodeMessage {
    unsigned int(8) NbUnits;
    for (i=0; i<NbUnits; i++) {
    int(BSPDecoderConfig.indexNbBits) nIndex;
    int(BSPDecoderConfig.indexNbBits) nParentIndex;
    float(BSPDecoderConfig.coefNbBits) a;
    float(BSPDecoderConfig.coefNbBits) b;
    if (BSPDecoderConfig.is3D) {
            float(BSPDecoderConfig.coefNbBits) c;
      }
    float(BSPDecoderConfig.coefNbBits) d;
    int(BSPDecoderConfig.indexNbBits) nIndexFront;
    int(BSPDecoderConfig.indexNbBits) nIndexOverlap;
      int(BSPDecoderConfig.indexNbBits) nIndexBack;
    int(BSPDecoderConfig.objCountNbBits) nFrontObjCount ;
    for (j=0 ; j<nFrontObjCount ; j++) {
       int(BSPDecoderConfig.indexNbBits) nFrontObjID;
      }
    int(BSPDecoderConfig.objCountNbBits) nOverlapObjCount ;
    for (k=0 ; k<nOverlapObjCount ; k++) {
       int(BSPDecoderConfig.indexNbBits) nOverlapObjID;
      }
      int(BSPDecoderConfig.objCountNbBits) nBackObjCount ;
    for (k=0 ; k<nBackObjCount ; k++) {
       int(BSPDecoderConfig.indexNbBits) nBackObjID;
      }
   }
}
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14496-11:2005/Amd 6:2009

© ISO/IEC 2009 – All rights reserved

8.14.3.7.3 Semantics

NbUnits: number of nodes in the BSP tree

nIndex: node ID

nParentIndex: parent node ID (-1 if none)

a: plane coefficient, following equation $ax+by+cz+d=0$

b: plane coefficient, following equation $ax+by+cz+d=0$

c: plane coefficient, following equation $ax+by+cz+d=0$

d: plane coefficient, following equation $ax+by+cz+d=0$

nIndexFront: front child node ID (-1 if none)

nIndexOverlap: overlap child node ID (-1 if none)

nIndexBack: back child node ID (-1 if none)

nFrontObjCount: number of objects front-side of the plane

nFrontObjID: front-side object ID

nOverlapObjCount: number of objects overlaping the plane

nOverlapObjID: overlaping object ID

nBackObjCount: number of objects back-side of the plane

nBackObjID: back-side object ID

8.14.3.8 Stream specific to cell&portals

8.14.3.8.1 Overview

Cell&Portal Message			
cellCount	totalCount	Cell	Cell

cellCount: number of cells in stream

totalCount: total number of cells

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14496-11:2005/Amd 6:2009

© ISO/IEC 2009 – All rights reserved

Cell		
cellID	CellGeometry	cellArray

CellGeometry					
centerX	centerY	centerZ	dX	dY	dZ

8.14.3.8.2 Syntax

```
class CellPortalNodeMessage {
    unsigned int(CellPortalDecoderConfig.cellCountNbBits) cellCount;
    unsigned int(CellPortalDecoderConfig.totalCountNbBits) totalCount;

    for (i=0; i<cellCount; i++) {
       int(CellPortalDecoderConfig.totalCountNbBits) cellID;
       int(CellPortalDecoderConfig.cellGeomNbBits) centerX;
       int(CellPortalDecoderConfig.cellGeomNbBits) centerY;
       if (CellPortalDecoderConfig.is3D)
       {
       int(CellPortalDecoderConfig.cellGeomNbBits) centerZ;
       }
       int(CellPortalDecoderConfig.cellGeomNbBits) dX;
       int(CellPortalDecoderConfig.cellGeomNbBits) dY;
    if (CellPortalDecoderConfig.is3D)
       {
       int(CellPortalDecoderConfig.cellGeomNbBits) dZ;
       }
       for (i=0; i<totalCount; i++)
       unsigned int cellArray;
      }
}
```

8.14.3.8.3 Semantics

cellCount: number of cells in stream

totalCount: total number of cells

cellID: cell ID

centerX: cell Bounding Box position in X

centerY: cell Bounding Box position in Y

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14496-11:2005/Amd 6:2009

© ISO/IEC 2009 – All rights reserved

centerZ: cell Bounding Box position in Z

dX: cell Bounding Box size in X

dY: cell Bounding Box size in Y

dZ: cell Bounding Box size in Z

PortalID: portal ID, inside cellule

cellArray: array giving list of visible cells from cell i

8.14.3.9 Stream specific to PVS

8.14.3.9.1 Overview

PVS Message

cellCount	objCount	PVSGrid	Cell	Cell	Cell

cellCount: total number of cells

objCount: total number of objects

PVSGrid: grid partition parameters (optional)

PVSGrid

zmin	zmax	xmin	xmax	ymin	ymax	dx	Dy

Cell

CellID	pvsArray

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14496-11:2005/Amd 6:2009

8.14.3.9.2 Syntax

```
class PVSNodeMessage {
    unsigned int(PVSDecoderConfig.cellCountNbBits) cellCount;
    unsigned int(PVSDecoderConfig.objCountNbBits) objCount;
    bool(1) bRegular;
    if (bRegular) {
        float(PVSDecoderConfig.pvsGeomNbBits) zmin;
        float(PVSDecoderConfig.pvsGeomNbBits) zmax;
        float(PVSDecoderConfig.pvsGeomNbBits) xmin;
        float(PVSDecoderConfig.pvsGeomNbBits) xmax;
        float(PVSDecoderConfig.pvsGeomNbBits) ymin;
        float(PVSDecoderConfig.pvsGeomNbBits) ymax;
      float(PVSDecoderConfig.pvsGeomNbBits) dx;
        float(PVSDecoderConfig.pvsGeomNbBits) dy;
    } else {
      PVSMesh;
    }
    for (i=0; i<nbCellCount; i++) {
      unsigned int(PVSDecoderConfig.cellCountNbBits) nCellID ;
      for (j=0; j<totalCount; j++)
        unsigned int pvsArray;
    }

    }
}
```

8.14.3.9.3 Semantics

cellCount: total number of cells

objCount: total number of objects

bRegular: partition based on a regular grid (1) or based on indexedfaceset (0)

zmin: minimum in Z

zmax: maximum in Z

xmin: grid minimum in X

xmax: grid maximum in Y

ymin: grid minimum in Y

ymax: grid maximum in Y

dx: grid step in X

dy: grid step in Y

nCellID: cell ID

pvsArray: array giving list of visible objects from cell i

PVSMesh: this is the mesh describing the cells in the non-regular case.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14496-11:2005/Amd 6:2009

 © ISO/IEC 2009 – All rights reserved

8.14.3.10 PVSMesh

8.14.3.10.1 Syntax

```
class PVSMesh {
        unsigned int(32) NbVertices;
    unsigned int(32) NbFaces;
    for (int i=0; i< NbVertices; i++) {
        int(32)VArray[i];
    for (int i=0; i< NbFaces; i++) {
        int(32) FArray[i];
       }

}
```

8.14.3.10.2 Semantics

NbVertices: this is the number of vertices in the mesh.

NbFaces: this is the number of faces in the mesh.

VArray: this is the array of points of the mesh. It has to be interpreted in the same way as the **Coordinates** field of an indexedFaceSet.

FArray: this is the array of facets of the mesh. It has to be interpreted in the same way as the **CoordIndex** field of an indexedFaceSet.

8.14.3.11 HierarchicalDescriptionPacket

8.14.3.11.1 Syntax

```
class HierarchicalDescriptionPacket {
       unsigned         int(HierarchicalDescriptionDecoderConfig.nbSubTreesNbBits)
       nbSubTrees;
       for (i= 0; i < nbSubTrees; i++) {
              HierarchicalDescriptionSubTree subTree;
       }
}
```

8.14.3.11.2 Semantics

nbSubTrees: number of hierarchical description sub-trees that are embedded in this packet.

The `HierachicalDescriptionSubTree` is the base class used only with description trees.

```
class HierarchicalDescriptionSubTree {
       switch(SPFootprintDecoderConfig.type) {
              0: FPHDescSubTree;
              1: // to be defined
       }
}
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14496-11:2005/Amd 6:2009

8.14.3.12 FPHDDescSubTree

8.14.3.12.1 Syntax

The `FPHDDescSubTree` is the specific class used only with description trees.

```
class FPHDescSubTree extends HierarchicalDescriptionSubTree {
      unsigned                int(SPFootprintDecoderConfig.nbNodesInSubTreeNbBits)
      nbNodesInSubTree;
      unsigned   int(SPFootprintDecoderConfig.nbNodesOnFirstLevelOfSubTreeNbBits)
      nbNodesOnFirstLevelOfSubTree;
      unsigned                           int(SPFootprintDecoderConfig.indexNbBits)
      indexParentFirstNodeInSubTree;
      unsigned int(SPFootprintDecoderConfig.indexNbBits) indexFirstNodeInSubTree;
      int(SPFootprintDecoderConfig.nbSubTreesChildrenNbBits) nbSubTreesChildren
      for (i= 0; i < nbSubTreesChildren; i++) {
            int(SPFootprintDecoderConfig.nbSubTreesNbBits) indexSubTreeChild
            int(SPFootprintDecoderConfig.nbNodesOnLastLevelNbBits)
            nbNodesOnLastLevel
            switch (SPFootprintDecoderConfig.networkType) {
                  0: // no additionnal informations
                  1:          int(SPFootprintDecoderConfig.subTreeSizeNbBits)
                  subTreeChildSize
            }
            for (i= 0; i < nbNodesInSubTree; i++) {
                  SPFootprintNodeMessage node;
            }
      }
}
```

8.14.3.12.2 Semantics

nbNodesInSubTree: number of nodes in the sub-tree.

nbNodesOnFirstLevelOfSubTree: number of nodes in the sub-tree first level.

indexParentFirstNodeInSubTree: father node index.

indexFirstNodeInSubTree: index of first node in the sub-tree.

nbSubTreesChildren: number of sub-tree children.

indexSubTreeChild: sub-tree child index.

nbNodesOnLastLevel: number of nodes in the sub-tree first level.

subTreeChildSize: size of current sub-tree.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14496-11:2005/Amd 6:2009

© ISO/IEC 2009 – All rights reserved

8.14.3.13 SPFootprintNodeMessage

8.14.3.13.1 Syntax

```
class SPFootprintNodeMessage {
      unsigned int(SPFootprintDecoderConfig.nbChildrenNbBits) nbChildren;
      if (nbChildren > 0) {
            unsigned int(8) encodedMetricError;
      }
      int(1) isFirstLevel;//this is a temporary non-parsable variable
      if (isFirstLevel) {
            float(32) gcX;
            float(32) gcY;
            float(32) gcZ;
            unsigned      int(SPFootprintDecoderConfig.rootChildrenRadiusNbBits)
            radius;
      }
      else {
            unsigned int(5) nbBitsDelta;
            int(1) isDeltaXNeg;
            unsigned int(nbBitsDelta) deltaX;
            int(1) isDeltaYNeg;
            unsigned int(nbBitsDelta) deltaY;
            int(1) isDeltaYNeg;
            unsigned int(nbBitsDelta) deltaZ;
            int(1) isDeltaRadiusNeg;
            unsigned int(nbBitsDelta) deltaRadius;
      }
}
```

8.14.3.13.2 Semantics

nbChildren: number of children nodes.

encodedMetricError: metric error of the node.

isFirstLevel: if true, node is assigned to root node.

gcX, **gcY**, **gcZ**: node gravity centre coordinates.

Radius: node radius.

nbBitsDelta: number of bits used to decode deltaX, deltaY, deltaZ and deltaRadius.

isDeltaXNeg: specifies whether deltaX is negative.

deltaX: used to determine child x sphere coordinate (i.e the difference between father node gravity centre X coordinate and current node gravity centre X coordinate).

deltaY and **deltaZ**: are the equivalents of deltaX but for the y and z coordinate respectively.

deltaRadius: used to determine the child sphere radius.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14496-11:2005/Amd 6:2009

© ISO/IEC 2009 – All rights reserved