第９部

Software Defined Media

SDM WGメンバー

第１章　はじめに

Software Defined Media（SDM）コンソーシアムは，2014年に設立され，オブジェクト指向のディジタルメディアと，インターネット環境が前提の映像・音響空間を用いたビジネス創造を目指してきた．昨年までの報告書では，SDMのコンセプトやアーキテクチャ，試作プラットフォーム[51, 52]，データ収録[53]，収録データベース設計[54, 55, 56]，SDMプラットホームを利用したアプリケーションであるSDM360^2 [57, 58]やLiVRation [59, 60]、Web360^2[61]などを報告してきた．

本報告書では，2020年度に実施した活動を報告する．

第２章　SDM Ontology Version 2.0の策定

SDMコンソーシアムが提案するSDM Ontologyは，メディアデータとともに様々な種類・粒度，かつ，膨大な量のメタデータを階層構造に整理し管理することを可能とする．一般的なメディア制作の流れは収録，編集，配信，再生の段階に分かれ，従来の提案であるSDM Ontology Version 1 [55, 56]では，メディアの収録段階における環境や内容，対象に関する記述を可能とする設計となっていたが，収録段階に続く，編集，配信，再生の段階については記述対象外であった．今回，主にメディア編集・加工段階に関する記述も対象範囲に含むようにSDMOntology Version 1の構造を見直し，修正および拡張したSDM Ontology Version 2 [54]を提案する．[54]の提案をもとに語彙の構造や命名規則を整理し，オントロジーを洗練させた．SDM Ontologyの記述可能な範囲の概要を図1に示す．

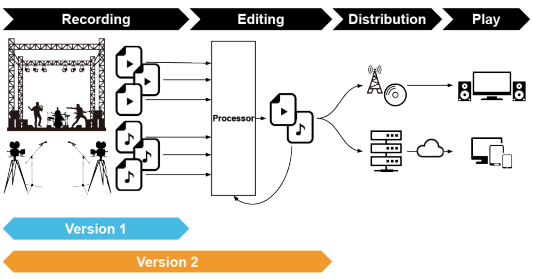

図1　メディア制作フローの概要とSDM Ontologyの記述可能な範囲．SDM Ontology Version 1では収録段階まで，SDM Ontology Version 2では収録段階および編集・加工段階までを主な対象範囲とする．

2.1　要求事項

メディア収録の情報は静的であり，一度メタデータを階層構造化してしまえばアーカイブとして扱うことができる．したがって，収録段階のみを対象としていたSDM Ontology Version 1では，静的データを記述することが可能な構造に設計されていた．一方で，メディアの編集・加工作業は収録されたデータや編集されたデータをもとに繰り返しおこなわれることも多く，一般に動的といえるため従来の静的データを記述するための構造設計では対応しきれない．

また，Version 1では，カメラやマイクなどの収録機器とメディアファイルは1対1関係で対応付けられることが前提となっていたが，編集段階を考えると必ずしもメディアファイルは収録機器だけが生成するものではなく，編集用のソフトウェアやミキサーなどのメディアプロセッサから生成される場合もある．さらには，編集作業に用いられるメディアプロセッサの情報を記述するための構造が設計にない．

このような問題を解消するためにSDM Ontology Version 1の設計を見直して，オントロジーを修正・拡張する必要がある．修正・拡張に関する主な要求事項を以下に記す．

動的な情報を記述するための再帰構造の導入

メディアの編集・加工作業は多数の編集工程を経ておこなわれる場合が多い．素材となる収録データを加工して「1次加工物」を生成し，これを素材としてさらに加工することで「2次加工物」を生成する．この過程を繰り返して生まれた「n次加工物」がメディアコンテンツの最終的な完成形として世の中に出回っていくことが一般的である．繰り返しによる動的情報は再帰的な構造によって記述することが可能であり，メディアデータの編集履歴を作成することが可能となる．また，再帰的記述は動的な構造の表現だけでなく包含関係の表現も可能とする．包含関係の表現は静的情報の記述に対しても非常に有効であり，Version 1では記述できなかった詳細な情報構造を表現できるようになる．例えば，マイクアレイとは，複数のマイクを配置し単一のマイクだけでは得られない音の空間情報を収録する収録機器であり，マイクアレイ自体を1つの収録機器と見ることもできれば，マイクアレイがもつ単一のマイクを収録機器の最小単位としてマイクアレイはそれらを包含する収録機器である，と記述することもできる．逆に，同じ対象を収録する収録機器群をグループ化して収録機器グループを記述したり，収録機器グループ群をさらにグループ化することで収録機器環境そのものを記述することも可能となる．このように再帰的記述の導入は，メディアの編集・加工段階の表現だけでなく詳細な情報の表現にも貢献し，SDM Ontology Version 2における大きな変更点の1つである．

時空間メディアへの抽象化

不足していたメディア編集・加工段階の構造設計と並行して従来設計の見直しをおこなう．SDM Ontology Version 1の設計では，音楽イベントに踏み込んだクラスや語彙も多く用意していたが，それぞれの分野・イベントに特化した記述は，既存の外部のオントロジーと連携することで表現可能であると考える．Version 2では，オントロジーの表現対象はメディアコンテンツの制作フローの核となる部分にとどめ，分野ごとの専門的な知識やイベント内容などの特に具体的な情報は適宜外部オントロジーと組み合わせて記述することとする．さらに，対象メディアを三次元映像・音声メディア前提だったものから「時空間情報をもつメディア」へと抽象度を高め，センサー群によるデータメディアなども表現対象となるようにオントロジーを拡張した．このような抽象化によってSDM Ontologyはより幅広い種類のメディアを取り扱い，メディアコンテンツの制作を管理する汎用性の高いオントロジーとなる．

2.2 SDM Ontology Version 2の概要

2.1で述べた要求事項をもとにオントロジーの修正・拡張をおこない，SDM Ontology Version 2を提案する．

以下の各項では，オントロジーの構造を図2から図3，図4，図5までに示すように図示しながら説明する．図中の各ノードはオントロジーのクラスであり，赤字で記されたクラスはSDM Ontologyにおいて特に重要な，オントロジー構造の核である基本クラスを表し，青字のクラスはその他のクラスを表す．クラスノード内に書かれた内容はDatatype Property（データ型プロパティ）と呼ばれるオントロジーの語彙（述語）であり，目的語に文字列や数値，URIなどのリテラルをとる．ノード間をつなぐ黒色矢印の傍らに書かれているのはObject Property（オブジェクトプロパティ）と呼ばれる語彙（述語）であり，矢印の根本側のクラスがもつ述語である．目的語には矢印が指す先のクラスインスタンスをとり，クラスインスタンス間の関係性を表す．一方で，図3以降に描かれるノード間をつなぐオレンジ色矢印はクラスの親子関係を表し，矢印根本のクラスは矢印が指すクラスのサブクラスである．

それぞれの語彙は原則として，SDM Ontologyのベース URIであるhttp://sdm.hongo.wide.ad.jp/sdmo/からの相対URIで示している．図2のMediaクラスではノードに“Media”と記されているが，正式にはhttp://sdm.hongo.wide.ad.jp/sdmo/Mediaという絶対URIによって識別される．また，Mediaクラスのノード内の“startAt”というデータ型プロパティはhttp://sdm.hongo.wide.ad.jp/sdmo/startAtで識別される．ただし，外部オントロジーを利用している場合には「接頭辞+相対URI」の形で示してあり，例えば，接頭辞“schema:”がhttp://schema.org/ [62]を表

し，相対URIが“name”である“schema:name”は，http://schema.org/nameと等価である．ベースURIや接頭辞の情報は図中では省略しているが，通常，オントロジー定義を記述するOWL（Web Ontology Language）[*1]ファイルやRDF（Resource Description Framework）[*2]データのファイルの先頭にまとめて記述される．

2.2.1 構造概要

SDM Ontology Version 2の構造概要を図2に示す．基本クラスを設計するにあたり次の考察をおこなった．現実空間内での出来事は収録・編集作業によってデジタルデータに記録され，仮想空間内で取り扱うことが可能となる．逆に，デジタルデータを用いて生成した仮想空間は再生システムによって現実空間内で再生することが可能となる．SDM Ontologyはここでいうデジタルデータをと呼び，Mediaの収録，編集・加工，再生[*3]の流れを包括的に記述するために設計したオントロジーである．現実空間の事象をMediaに記録するシステムをRecorder，Mediaを編集・加工，または，分析，生成するシステムをProcessor，Mediaを再生するシステムをPlayerと呼ぶ．また，Mediaやこれらのシステムは様々なメタデータをもち，これをContextと呼ぶ．ここで述べた要素をすべてメディア制作において重要であると考え，SDM Ontology Version 2ではこれらの要素を表現するためのContextクラス，Recorderクラス，Processorクラス，Playerクラス，Mediaクラスの5つを基本クラスとする．基本クラスの詳細は2.2.2から2.2.4にかけて説明する．

2.1で述べたように，SDM Ontology Version2では「時空間情報をもつメディア」を取り扱うことを前提とするため，メタデータの中でも特に時空間情報は重要な意味をもつと考え，時空間メタデータの扱いに特化したCoordinateSystemクラス，Geometryクラスを定義し，Contextクラスとは分離して表現する．CoordinateSystemクラスで時空間座標系を定義し，Geometryクラスによって時空間座標系上の点（座標）を表現する．例えば，CoordinateSystemクラスで右手系のxyz空間座標系を定義し，Geometryクラスではこの座標系における点（1, 2, 3）を表現するなどして運用する．CoordinateSystemクラスのインスタンスとGeometryクラスのインスタンスはmeasures/measuredByプロパティで相互参照し座標系定義をおこない，Geometryクラスのインスタンスは基本クラスの各インスタンスとgeometryOf/geometryAtプロパティのリンクによって時空間メタデータを与える．CoordinateSystemクラスとGeometryクラスはそれぞれ，ignf:CoordinateSystem [63][*4]，geom:Point [64][*5]のサブクラスとして定義しているが，SDM Ontologyを用いたメディア管理の実運用時には座標系定義や座標記述よりも多くの時空間情報を表現しなければならないことが予想され，表現力の不足が懸念される．CoordinateSystemクラスとGeometryクラスの詳細な構造の検討や整備については今後の課題としたい．

図2 右下にあるMediaEventクラスはMediaクラスの包含関係を表現するための補助クラスであり，詳細は2.2.4で述べる．

2.2.2 Contextクラス

ContextクラスはSDM Ontologyの基本クラスがもつ様々なメタデータを包括的に記述するためのクラスである．メタデータの種類を大別してSDMEventクラス，Targetクラス，Programクラスをサブクラスとしてもつ．Contextクラス，および，そのサブクラスに関する構造図を図3に示す．

SDMEventクラスは収録対象である現実空間内での事象（イベント）に関する日時や場所をはじめとする様々な収録環境のメタデータを記述し，Targetクラスは収録対象である人物やモノを記述する．収録対象のイベントがもついくつかの意味をもつ事象のまとまりをプログラムと呼び，Programクラスではこれを記述する．例えば，あるコンサートを記述する場合，コンサート名や開催日時・場所はSDMEventクラス，指揮者や演奏者，楽器はTargetクラス，演目はProgramクラスのように整理して記述可能

＊1 https://www.w3.org/OWL/
＊2 https://www.w3.org/RDF/
＊3 SDM Ontology Version 2は特に編集・加工段階の表現に注力して設計したが，後々必要になってくるだろう再生段階の記述に関しても最低限導入している
＊4 http://data.ign.fr/def/ignf#CoordinateSystem
＊5 http://data.ign.fr/def/geometrie#Point

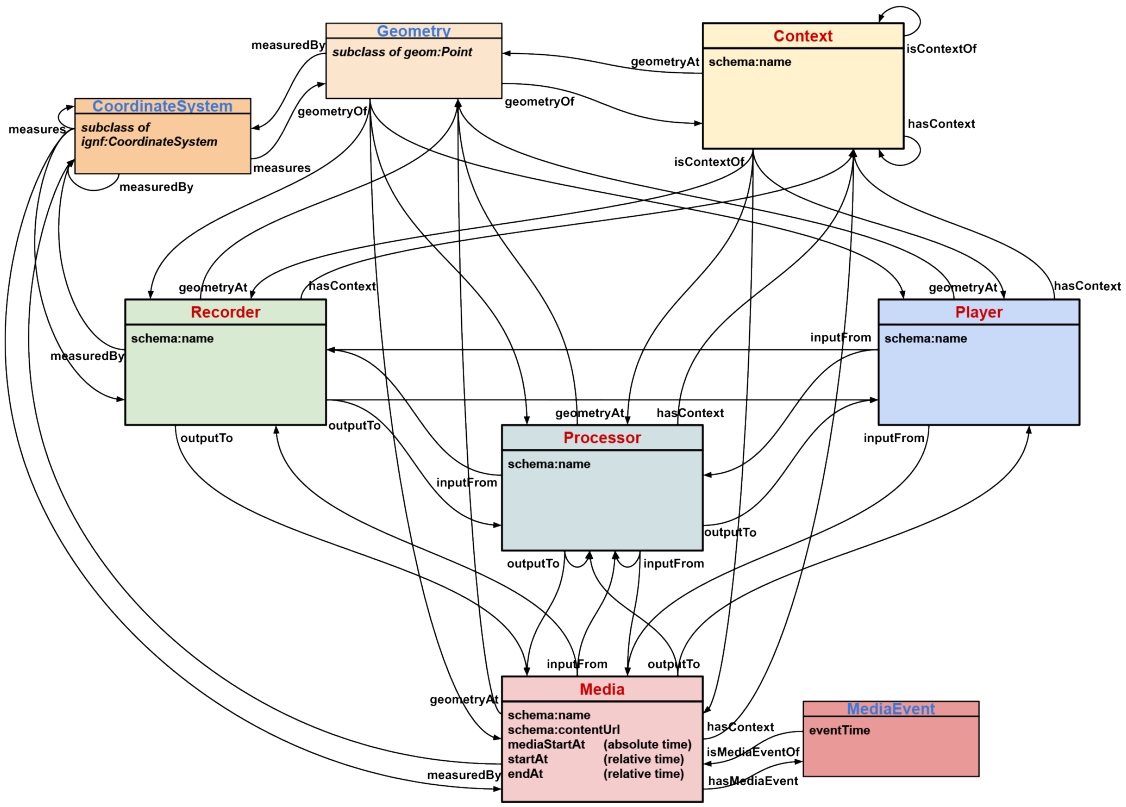

図2　SDM Ontology Version 2の構造概要．各ノードはクラスであり，赤字は基本クラスを表し，青字はその他のクラスを表す．ノード内に書かれた語彙はデータ型プロパティであり，ノード間をつなぐ黒色矢印はオブジェクトプロパティである．

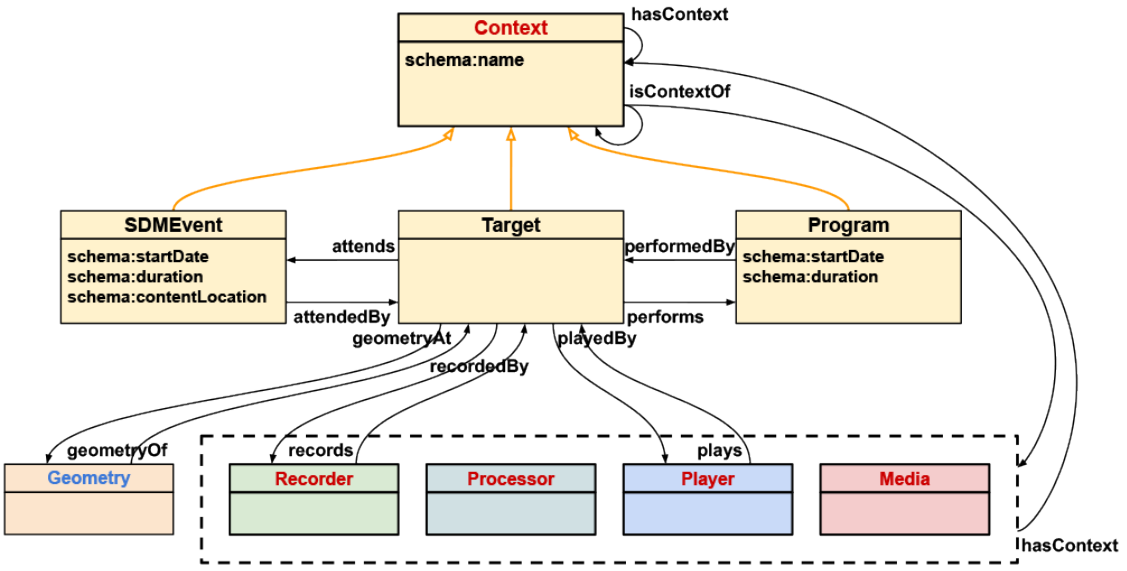

図3　SDM Ontology Version 2のContextクラスの概要．各ノードはクラスであり，ノード内に書かれた語彙はデータ型プロパティ，ノード間をつなぐ黒色矢印はオブジェクトプロパティである．また，ノード間をつなぐオレンジ色矢印はクラスの親子関係を表す．

である．ただし，プログラムを収録対象のモノであるという見方をすれば，Programクラスの内容はTargetクラスで記述することも可能であるため，将来的にProgramクラスは廃止される可能性がある．

SDM Ontology Version 1ではSDMEventクラスやTargetクラスは，Contextクラスと並ぶ基本クラスとして定義されていたが，いずれも大局的に見るとメタデータの記述をおこなう共通の働きをもつことからSDMEventクラスやTargetクラスをContextクラスのサブクラスとして定義し直した．また，Version 1は音楽イベントのメディア収録を前提として設計したオントロジーであったため，音楽イベントに特化したクラスなど具体的で詳細なクラスがいくつか用意されていたが，Version 2では音楽イベントに限らず「時空間情報をもつメディア」を取り扱うことのできるよう具体的なクラス・プロパティは削除し構造をシンプルに，抽象度を高めた．したがって，Version 2を用いて音楽イベントのデータを記述する場合にはSDM Ontologyでメディア制作の核の部分を記述し，演奏された楽曲や演奏者，楽器などの情報は音楽を専門とする外部のオントロジーと連携して記述することになる．

2.2.3　Recorderクラス，Processorクラス，Playerクラス

Recorderクラス，Processorクラス，Playerクラスはそれぞれ図4に示すように用途を詳細に表現できるサブクラスと包含関係を記述するためのサブクラスをもつ．

Recorderクラス，Processorクラス，Playerクラスのそれぞれで同じ構造をとるため，ここでは代表してRecorderクラスについて述べる．Recorderクラスのサブクラスにはより用途が詳細なDataRecorderクラス，AudioRecorderクラス，VideoRecorderクラスと包含関係を表現するCompositeRecoderクラスを定義する．例えば，ビデオカメラやスマートフォンなどでの動画撮影では，一般に映像収録とステレオ(2ch)音声収録が同時におこなわれる．SDM Ontology Version 1ではこれらをRecorderクラス，あるいは，映像収録がメインである場合はVideoRecorderクラスのインスタンスとして記述することは可能であった．Version 2では従来通り単純にRecorderクラスあるいはVideoRecorderクラスのインスタンスと記述することもできれば，ビデオカメラやスマートフォンがもつ映像収録機構を切り出してVideoRecorderクラスのインスタンス，ステレオ音声収録機構を2chまとめて，あるいは，1chずつ切り

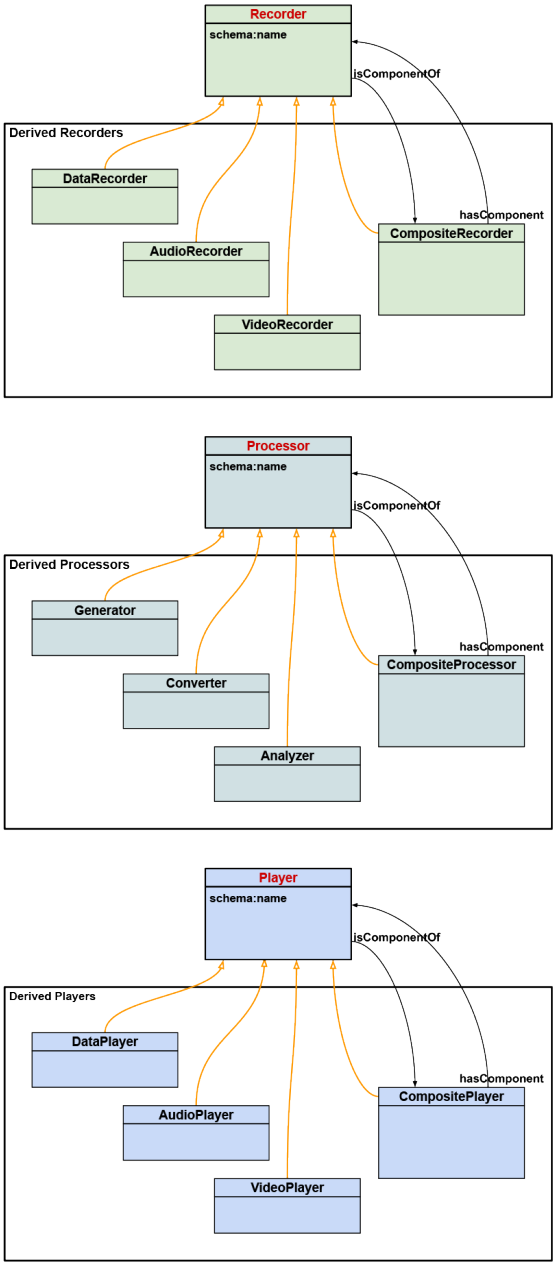

図4　SDM Ontology Version 2のRecorderクラス，Processorクラス，Playerクラスの概要．各ノードはクラスであり，ノード内に書かれた語彙はデータ型プロパティ，ノード間をつなぐ黒色矢印はオブジェクトプロパティである．また，ノード間をつなぐオレンジ色矢印はクラスの親子関係を表す．

出してAudioRecorderクラスのインスタンスとして定義し，ビデオカメラやスマートフォン自体はこれらのVideoRecorderクラス，AudioRecorderクラスインスタンスを包含するCompositeRecorderクラスのインスタンスとして記述することも可能となる．2.1で述べたマイクアレイの例も同様の方法で記述することができる．また，カメラやマイクあるいはセンサなどのそれぞれ独立な収録機器が同じ対象を収録していた場合，これらをグループ化したい場合がある．この場合にもCompositeRecorderクラスによる包含関係の記述によって収録機器グループの記述が可能であり，再帰的な包含関係の記述によって収録環境全体を1つのグループとして記述することも可能となる．Processorクラス，Playerクラスでも同様であり，包含関係を記述する“Composite”クラスの再帰的な特性によって収録，編集・加工，再生システムはそれぞれ機器の最小単位（ミクロ）から複合系（マクロ）までユースケースにあわせた自由な粒度で記述することが可能となる．

2.2.4 Mediaクラス

MediaクラスについてもRecorderクラス，Processorクラス，Playerクラスと同様にサブクラスをもつ．ただし，メディア編集の記述のために図5に示すように，Recorderクラス，Processorクラス，Playerクラスの構造とは少し異なりMediaEventクラスとの連携が発生する．

メディア管理においては，メディアデータの記録時刻を管理するための絶対的な時間軸（通常は現実空間内の時間軸）のほかに，記録されたメディアデータの再生区間を表す相対的な時間軸など，複数の時間軸を意識する必要があり，時間情報の取り扱いは重要である．Mediaクラスには時間情報を扱うプロパティとして，メディアデータの記録を開始した時刻を表すmediaStartAtプロパティ，メディアファイルの有効な再生区間を示すためのstartAt/endAtプロパティを用意した．mediaStartAtプロパティが絶対時刻を記述するのに対して，startAt/endAtプロパティはメディアファイルの先頭を原点とする相対時刻を指定することに注意されたい．画像データなど時

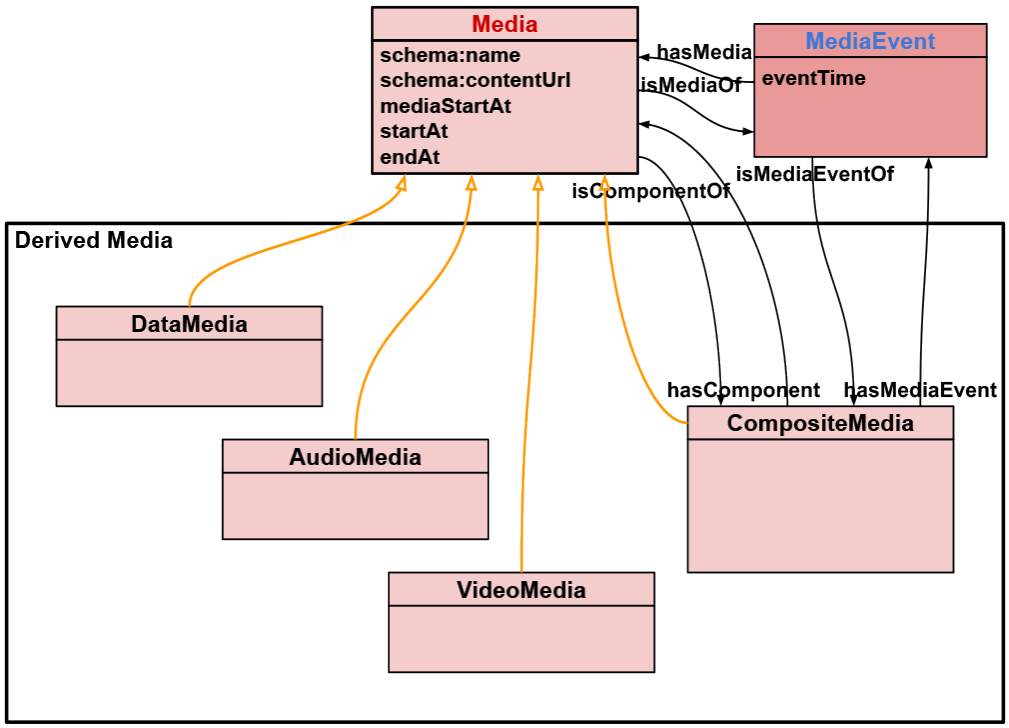

図5 SDM Ontology Version 2のMediaクラスの概要．各ノードはクラスであり，ノード内に書かれた語彙はデータ型プロパティ，ノード間をつなぐ黒色矢印はオブジェクトプロパティである．また，ノード間をつなぐオレンジ色矢印はクラスの親子関係を表す．

間情報を区間としてもたないメディアデータについてはstartAt/endAtプロパティをともに0とし，再生時間を0と定義することで表現することができる．

メディアのグループ化においては，Recorderクラス，Processorクラス，Playerクラスの包含関係とは異なり，グループ化した各メディアの時間的な配置も重要になる場合が多いと考える．CompositeMediaクラスを用いてメディアの包含関係を記述する際には，原則として各メディアの配置位置を記すMediaEventクラスのインスタンスを包含対象として指定することで，包含するメディアとその時間的な配置の組み合わせを表現することができる．例えば，CDやDVD，Blu-rayディスク上の音声や映像，字幕情報などのメディアデータは複数のトラックあるいはチャプターに分かれて時間軸上に配置されている．これらのメディアデータをトラックやチャプターを単位として適切なAudioMedia/VideoMedia/DataMediaクラスのインスタンスとして定義したのち，MediaEventクラスのインスタンスにそれぞれのメディアが配置されるべき時間位置（トラックやチャプターの開始位置）を指定し，CompositeMediaクラスにグループ化していくことでCDやDVD，Blu-rayディスク全体のメディアデータを記述することが可能となる．

一方で，MediaEventクラスを介することなく直接Mediaクラスのインスタンスを包含するような記述も可能であり，この場合は時間的な配置を気にしない単なる包含関係を示すことができるほか，暗黙的に包含するすべてのメディアを再生区間の時刻0（原点）の位置に配置することを表す．この表現方法では，同じメディアデータ内に映像と音声が同時に記録されている場合や複数のチャンネルをもつ音声を，映像・音声ごとや音声チャンネルごとに分離して個々のメディアインスタンスとして定義したのち，CompositeMediaクラスで包含関係を表現するなどの運用方法が考えられる．

第３章　SDMO対応アプリと普及活動

3.1　Web360^2

SDM Ontology Version 2の普及，および，コンテンツ制作者の視点からオントロジーの構造を考察する目的で，SDM Ontology Version 2準拠のデータをもとに動作するアプリケーションWeb360^{2}の開発し，実験を通して評価をおこなった．Web360^2 [61, 65]は，インタラクティブに360°動画および立体音響を視聴体験できるWebアプリケーションである．具体的にはWebブラウザ上で，収録済みの360°動画を用いた自由視点視聴，および，視聴者の動作に追従し，個別の音源をインタラクティブに制御することができる立体音響の提示を目指す．そのため，立体音響の手法として，従来より広く用いられているチャンネルベースや360°動画との相性がよいとされるHOA/アンビソニックスベースではなく，個別の音源の制御が容易なオブジェクトベース方式を採用する．[61, 65]を発表した時点では，視聴体験可能なコンテンツの情報やメディアデータはすべてソースコードに埋め込まれていたが，現行バージョンでは，SDM Ontology Version 2との連携によってLODから情報を取得するよう実装されている．また，JavaScriptのフレームワークであるVue.js[*6]を用いてUI（User Interface）/UX（User Experience）デザインを含めて刷新した．

Web360^{2}はGithub Pages[*7]にて公開している．関連するアプリケーションとWeb360^{2}の比較概要を表1にまとめる．現行バージョンでの視聴位置の移動は実験段階のβ機能であり，視聴位置の切り替えにはネットワーク環境に依存した遅延が起こる．

3.1.1　要求事項

先に述べた目的を踏まえてWeb360^{2}の要求事項として以下を想定した．

SDM Ontology Version 2準拠のデータの活用：
SDM Ontology Version 2の普及が前提にあるため，SDM Ontology Version 2準拠のデータの活用は必須事

＊6　https://vuejs.org/
＊7　https://sdm-wg.github.io/web360square-vue/

項である．

Webアプリケーション：

専用のアプリケーションをインストールすることなく，Webブラウザから直接利用可能な手軽さを目指す．SDM Ontology Version 2の普及において，誰でも手軽に体験できることは望ましいと考える．

自由な映像音声の視聴：

視聴者が自由に視聴位置や視聴角度を決めることができ，位置・角度および視聴オブジェクトがもつ3次元位置情報をもとに自動で適切な映像・音声をレンダリングして提示できる．

インタラクティブ性：

ジャイロセンサ搭載デバイスでは，インタラクティブな視聴角度の変化を体験することができる．また，視聴者の操作によって，各音源のオブジェクトに個別にアクセスしON/OFFの切り替えができる．これらの機能により，視聴者の興味関心に基づいた自由度の高い視聴体験が可能となる．

ストリーミング配信：

360°動画をHTTP Live Streaming（HLS）でストリーミング配信する．ストリーミング配信に対応することで，再生開始までの待機時間の短縮や余計な通信を省くことが可能となる．

3.1.2　設計

Web360²のシステム設計概要を図6に示す．視聴者の動作や音声オブジェクトへのタッチ操作を入力として受け付け，操作に応じた映像や音声をシステム内でリアルタイムにレンダリングして出力することでインタラクティブ性を実現する．映像・音声のメディアファイルはそれぞれ独立しており，システム内で同期をとりながら再生する．映像ファイルに360°動画，音声ファイルにはAudioSprite[*8]

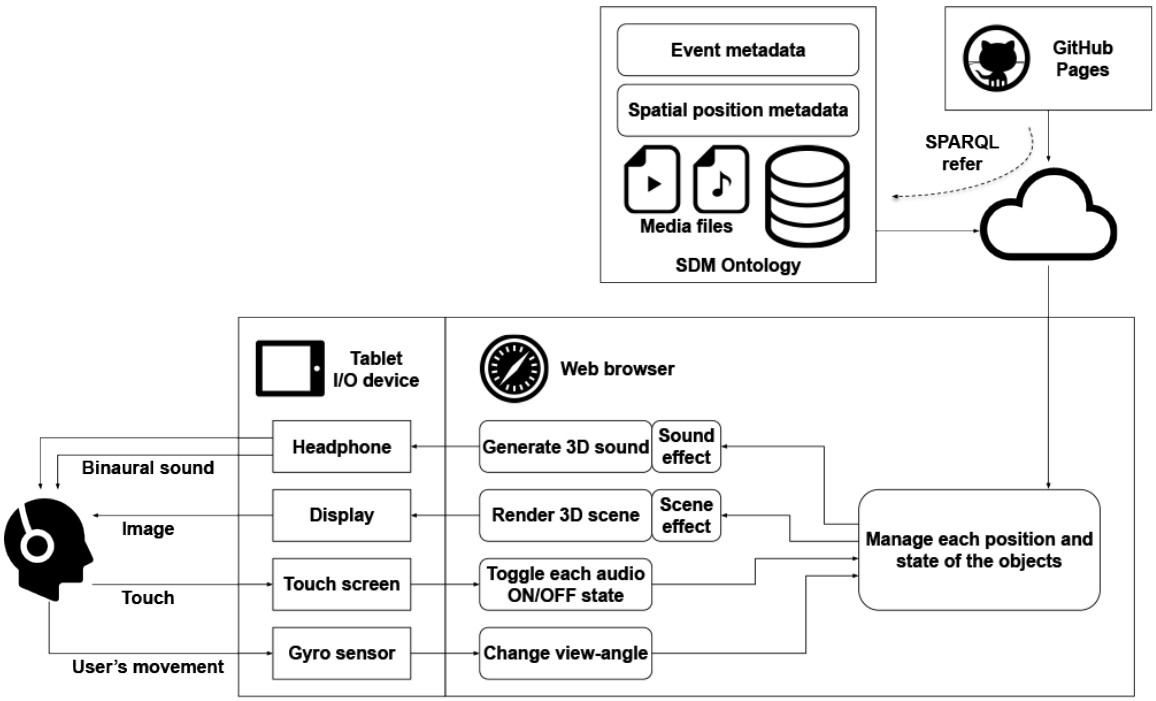

図6　Web360²のシステム設計．ユーザのインタラクティブな操作に応じてリアルタイムに映像・音声をレンダリングする．視聴コンテンツの情報はSPARQLクエリを用いてRDFストアに問い合わせて取得する．

＊8　https://github.com/tonistiigi/audiosprite

によって各音源ファイルを1つにまとめた音源群が収録されている．AudioSpriteは複数の音声ファイルを，時間間隔をとりながら並べて1つの音声ファイルにまとめる技術であり，再生の際のリクエスト数を減らすことができるため読み込み時間を短縮することができる．また，すべての音声が1つのファイルにまとまっていることで，音声ごとの読み込み遅延を気にする必要がなくなり，音声間の同期がとりやすいことも利点として挙げることができる．

360°動画はHLSでストリーミング配信され，視聴者を中心とする仮想的な球面内側に投影して再生される．また，音声オブジェクトのもつ3次元位置情報に基づいて，動画を投影する仮想的な球内部の各地点に音声可視化オブジェクトを描画し，各音声の強弱を表現したり，タッチ操作によってON/OFFを切り替えることができる．視聴者の動作は端末に搭載されたジャイロセンサを通してシステムに入力され視聴角度が変更される．視聴角度とON状態にある各音声オブジェクトの位置関係から聴こえるべき音声を生成し出力する．

3.1.3 素材データ

Web360^{2}の開発に用いた素材データは，2016年1月10日に慶應義塾大学日吉キャンパス内に建つ藤原洋記念ホールで開催された慶應義塾大学コレギウム・ムジクム古楽アカデミーのコンサートの収録データ，および，2017年1月26日に六本木ミッドタウン内にあるBillboard Live Tokyoで開催されたMusilogue Bandのジャズセッションの収録データである．いずれもSDMコンソーシアムの活動の一環で過去に収録したデータであり，[53, 55]に，それぞれの収録環境や収録方法が記述されている．

Web360^{2}で活用するために，収録当時に記録していた収録に関する環境や内容の情報をもとに，SDMOntology Version 2準拠のRDFデータファイルを，簡潔で可読性の高いTurtle記法で作成・整理し，SDMコンソーシアムが運用するGraphDB[*9]で構築されたRDFストア[*10]にデータを登録した．例えば，図7に示すように収録対象や収録機器，生成されるメディアファイルとその編集・加工の過程など，様々な情報を整理・分類する．図8は各素材データ収録時の収録機器の配置を表し，このような収録に関する情報を整理して図9に表すRDFデータを作成および登録した．Web360^{2}は，RDFストアにWebAPIを通じてSPARQLクエリを発行することでアプリケーションの動作に必要なデータを取得し，活用することができる．

3.1.4 実装

Web360^{2}は，JavaScriptのフレームワークであるVue.jsをベースに，WebVRのフレームワークであるA-Frame[*11]とWeb上で高度なオーディオ処理などを実行できるWeb Audio API[*12]を用いて実装した．コンテンツ視聴部分の実装概要を図10に示し，視聴体験中のスクリーンショットを図11に示す．

SPARQLクエリによる動的なデータ取得

Web360^{2}は，3.1.3項で言及したように，収録に関する様々な情報をまとめたRDFデータを格納するRDFストアにSPARQLクエリを発行してアプリケーションの動作に

表1　Web360^{2}と関連するアプリケーションとの比較

システム	環境	360° 動画	視聴位置操作	視聴角度操作	各音源操作	ライブ配信
SDM360^2 [7]	Unity	✓	✓	✓	✓	×
LiVRation [9, 10]	Unity + HMD	✓	✓	✓	✓	✓
Inside Music [16]	Web	×	✓	✓	✓	×
Web360^2	Web	✓	✓（β 機能）	✓	✓	×

＊9　http://graphdb.ontotext.com/

＊10　本稿では，RDFのデータを保存・格納するトリプルストア，および，RDFのクエリ言語であるSPARQLを用いてデータの問い合わせをするSPARQLエンドポイントの機能を有するサービスのことを指す．https://sdm.hongo.wide.ad.jp:7200/で運用している．

＊11　https://aframe.io/

＊12　ttps://www.w3.org/TR/webaudio/

必要なデータを取得する．現状，発行するクエリは以下の2種類である．

1. 視聴体験可能なイベントの取得
2. あるイベントに関する映像・音声メディアファイルや座標情報などの取得

視聴体験可能なイベントの取得

Web360[2]にアクセスすると，はじめに図12のようなイベント選択画面が表示される．この画面に表示されるイベントの各情報は，アプリケーションからRDFストアに向けて図13のSPARQLクエリを発行し取得される．

あるイベントに関する映像・音声メディアファイルや座標情報などの取得

イベント選択画面から1つのイベントを選択してタップあるいはクリックすると，図11のような視聴画面に遷移する．視聴画面では視点に対して音声可視化オブジェクトがそれぞれ正しい配置でレンダリングされる必要があるため，360°動画や各音声のメディア以外に，それらの座標情報が必要となる．また，映像・音声はそれぞれ独立したメディアファイルを用いるため映像と音声の同期処理が必要になる．同期のためにそれぞれのメディアファイルの長さの情報なども必要である．このように，視聴画面をレンダリングするためには選択されたあるイベントに対して様々な情報が必要であり，Web360[2]では，図14のSPARQLクエリを用いて収集する．

A-Frameを用いた処理概要

A-Frameを用いて，図10の右点線枠の手順で360°動画の投影や音声可視化オブジェクトの描画をおこなう．360°動画はHLSでストリーミング配信され，A-Frameのa-videosphereオブジェクトを用いて，カメラの座標を基点に広がる球面内部に映像を投影する．HLSはSafariやEdge，一部のモバイル端末用ブラウザではネイティブサポートされているが，ChromeやFirefox，IEなどサポートしていないブラウザも多く存在するため，hls.js[*13]を

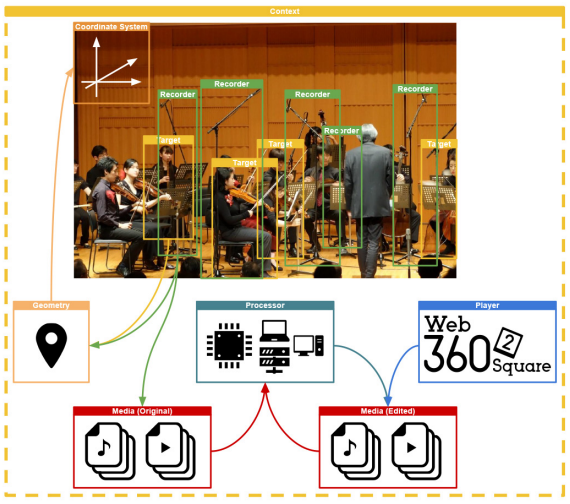

図7　SDM Ontology Version 2に準拠するデータ作成の概要．メディアの収録や編集・加工について，SDM Ontology Version 2が定義するクラスやプロパティを用いて整理・分類し，RDFデータを作成する．

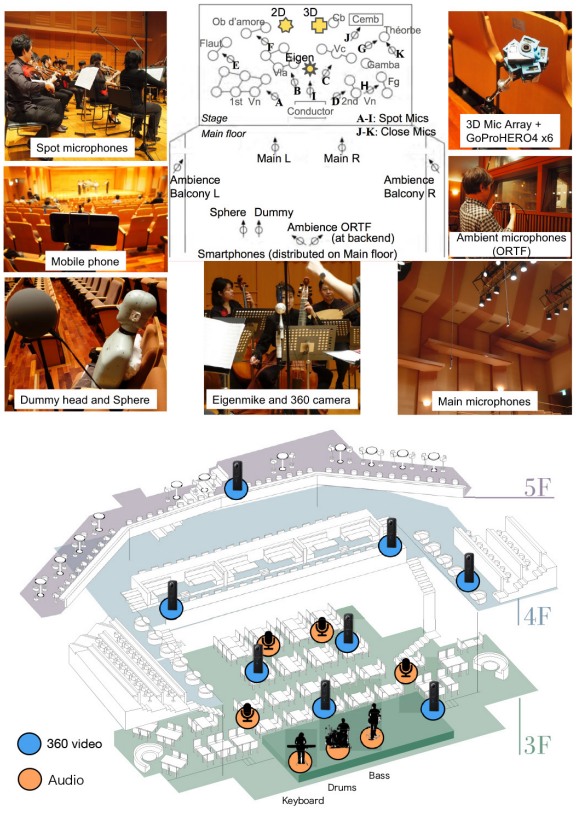

図8　慶應義塾大学コレギウム・ムジクム古楽アカデミーのコンサート(上)，および，Musilogue Bandのジャズセッション(下)の収録機器の配置を表す[55]．

＊13　https://github.com/video-dev/hls.js/

用いて対応した．音声の可視化はWeb Audio のAnalyserノードから提供される各音声の周波数領域データを用いて表現している．

また，視聴者のインタラクティブな操作は主にA-Frameが担う処理を介して入力される．視聴者の頭部動作は端末に搭載されたジャイロセンサに検知され，A-Frameのカメラオブジェクトに入力されることで視聴角度が変更される．タッチ操作はA-Frameでレンダリングしたオブジェクトにタッチすることで入力として認識され，各音声のON/OFF状態を変更することができる．

Web Audio APIを用いた処理概要

図10の左点線枠に示すように，Web AudioのAudioContext上で各ノードを鎖状に繋いでいき，視聴位置・角度とON状態の各音声オブジェクトから聴こえるべき音声をリアルタイムレンダリングする．はじめに各音源が1つにまとめられたAudioSpriteのファイルデータから各音源を抽出し，Buffer Sourceノードに入力する．次のGainノードでは，OFF状態の音声のゲインを0にすることで音声のON/OFF状態を表現する．映像はある視聴位置に対して視聴角度と各オブジェクトの絶対座標に応じて，A-Frameが自動的に適切な相対位置を計算しながらレンダリングするが，Web Audio上で音声オブジェクトは視聴角度の変動を自動追従できないため，カメラの視聴角度と絶対座標から相対座標を随時計算してPannerノードに受け渡すことで音声の追従性を実現する．例えば，視聴角度に対して音声オブジェクトが右側に存在すれば右側で音が鳴っているような音声を生成することができる．また，各音声の周波数領域データはAnalyserノードで計算され，A-Frameの音声の可視化処理に用いる．以上の処理までは各音源ごとにされ，最後にマスターゲインとなるGainノードやDynamicsCompressorノードでまとめられバイノーラル出力される．

```
# prefix definition
@prefix sdm: <http://sdm.hongo.wide.ad.jp/resource/> .
@prefix sdmo: <http://sdm.hongo.wide.ad.jp/sdmo/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix geom: <http://data.ign.fr/def/geometrie#> .

# ...

# an instance of microphone coordinate
sdm:Kmc160110AudioRecorderGeometry1
    a sdmo:Geometry ;

    # coordinate values
    geom:coordX "3.05"^^xsd:double ;
    geom:coordY "1.0"^^xsd:double ;
    geom:coordZ "-7.33"^^xsd:double ;

    # link to a microphone instance
    sdmo:geometryOf sdm:Kmc160110AudioRecorder1 ;

    # link to a coordinate system instance
    sdmo:measuredBy sdm:Kmc160110CoordinateSystem1 .

# ...
```

図9　素材データ収録をSDM Ontology Version 2に準拠するように記述したRDFデータ．例として収録機器配置に関する記述の一部を抜粋している．データはTurtle記法で記述されている．

音声の可視化

音声可視化オブジェクトは音声の周波数領域データに対応する棘状オブジェクトをもち，対応する周波数データの大きさによって長さや色を変化する．この棘状オブジェクトは[67]のアルゴリズムを参考に球面上に可能な限り均等に配置している．ただし，[67]では球面の極周辺における分布のずれの補正に関する言及があるが，配置に関してそこまでの精度を必要としないことから補正項の導入はおこなっていない．音声可視化オブジェクトがもつ棘の数は32としているため，図15に示す対応する音声の周波数領域データを32個の値に変換する処理が必要である．

Web AudioのAnalyserノードでは，時間領域データやFFT（Fast Fourier Transform，高速フーリエ変換）による周波数領域データを取得することができ，AnalyserNode.getByteFrequencyData()メソッドを用いて周波数データを取得する．周波数データの取得は音声可視化オブジェクトを構成するA-Frameコンポーネントのtickハンドラを用いて毎秒20回程度のオーダーで繰り返しおこなわれる．Web Audioのデフォルトのサンプリング周波数はOSやブラウザに依存する値で，一般には44,100Hzまたは48,000Hzであり，AnalyserNode.getByteFrequencyData()メソッドによってナイキスト周波数22,050Hzまたは24,000Hz以下の周波数データが得られるが，極端な高周波数帯・低周波数帯にはあまりデータが存在せず，有効な値が含まれる周波数の範囲は限られていることが多い．そこで，一度でも有効な値が現れ

たことがある周波数の上限・下限をとり，この範囲を有効周波数帯とする．有効周波数帯は再生開始から周波数データが取得される度に更新される．有効周波数帯の中に含まれるデータを周波数の高さの順に可能な限り要素数が均等な32グループに分けて，グループ内で平均値を計算する．得られた32グループそれぞれの平均値は音声可視化オブジェクトの32本の棘の長さと色に変換されて描画される．周波数の高さ順と音声可視化オブジェクトの棘の配置順を対応させて描画してみたところ，近接する周波数グループ同士に相関があるために部分的に棘が盛り上がるような不格好な形状となったため，周波数の32グループと音声可視化オブジェクトの32本の棘の配置の対応関係はランダムに定まるようにした．また，OFF状態の音声可視化オブジェクトの棘の色はグレーであり，ON状態では平均値が大きくなるにつれて青，水色，緑，黄，オレンジ，赤と変化していく．

負荷軽減

音源1つにつき32本の棘状オブジェクトをもつ音声可視化オブジェクトを生成し，さらにそれらがすべて独立に長さや色を変化させるため，デバイスがおこなうべき計算処理は非常に大きくなる．使用するデバイスの計算能力によっては，レンダリングのカクつきが起こりQoE（Quality of Experience）の低下につながる可能性が高い．Web360[2]では，以下の3つの負荷軽減のための対策を導入している．

1. 音声可視化処理のFPS（Frames Per Second）の低減
   A-Frameコンポーネントのtickハンドラはデフォルトでは毎秒60回から120回程度のオーダーで実行される

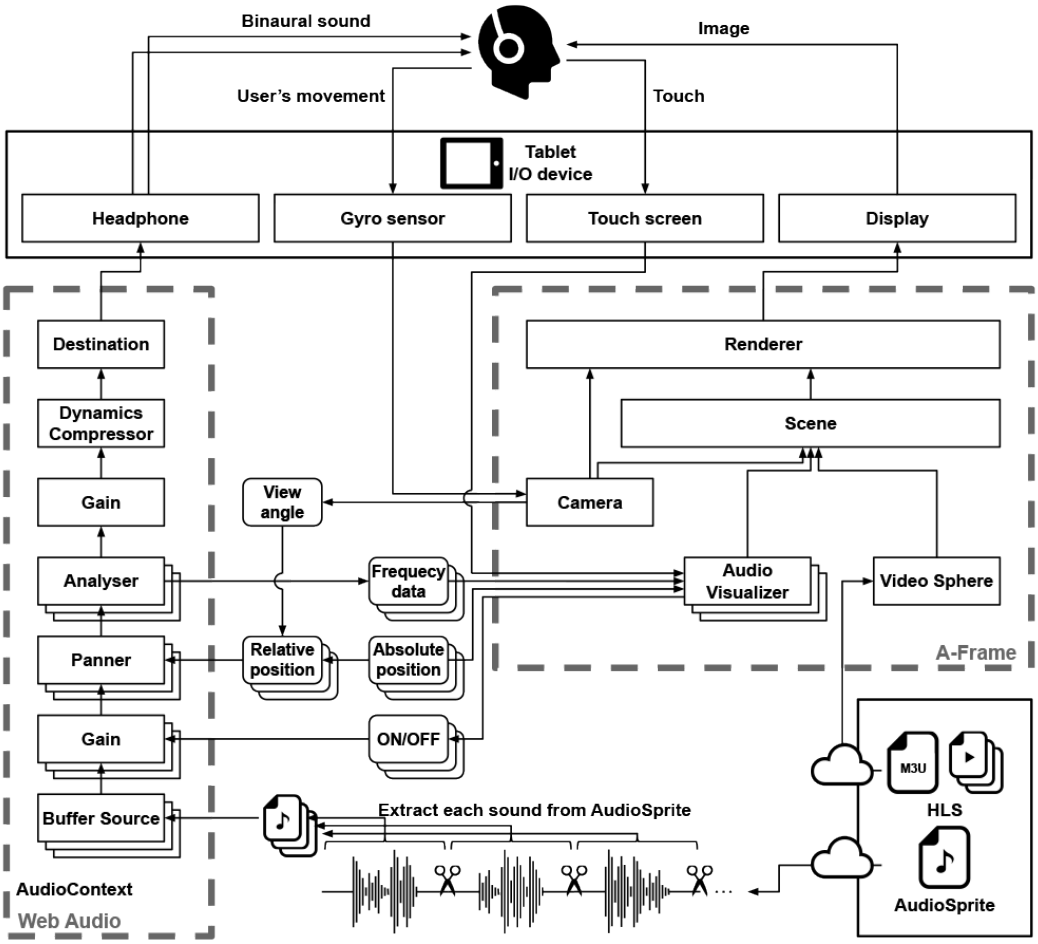

図10　Web360[2]の実装概要．
Web Audio APIを用いた処理（左点線枠）とA-Frameを用いた処理（右点線枠）を連携して実装している

が，音声可視化オブジェクトにおけるtickハンドラは毎秒20回程度のオーダーで実行されるように実行回数を低減している．様々なオーダーを試してみたが，音声可視化オブジェクトが十分滑らかに更新されるためには毎秒20回程度実行される必要があると主観的に判断した．

2. 死角にある棘状オブジェクトの非表示
音声可視化オブジェクトがもつ棘状オブジェクトは中心から全方位に均等配置で伸びているため，視聴位置

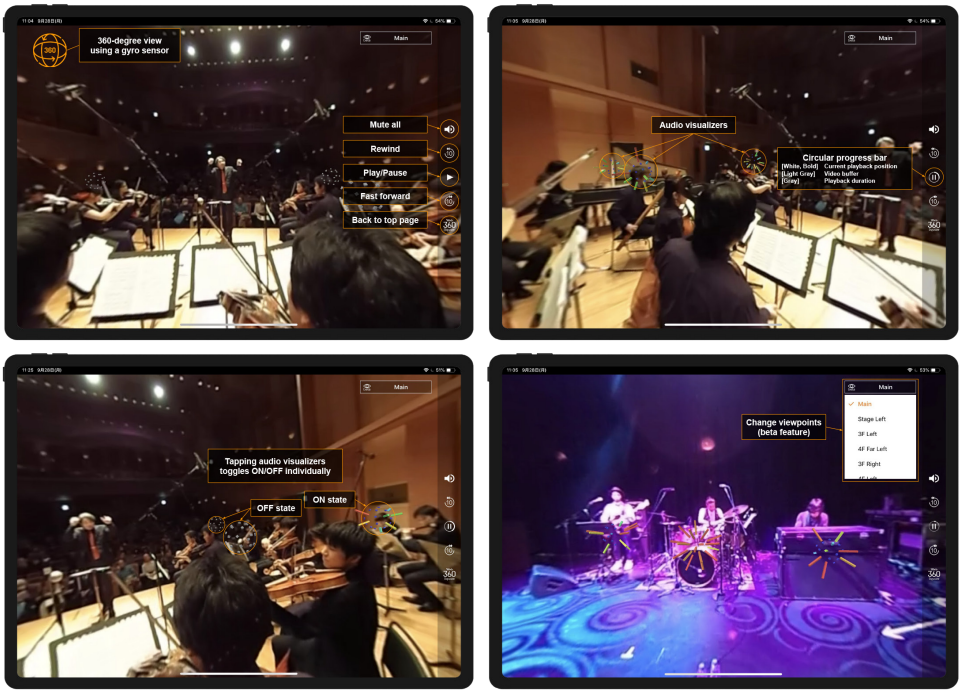

図11　視聴体験中のWeb360²のスクリーンショット．画面右側にコントロールパネルがあり，再生や一時停止，早送り，巻き戻しなどの基本操作ができる．ジャイロセンサ搭載デバイスでは，動作にあわせて視聴角度が変動する．音声は個別管理されていて，タップ(クリック)操作によってそれぞれ独立にON/OFFを切り替えることができる．視聴位置の変更操作は実験的な機能である．

図12　Web360²のイベント選択画面のスクリーンショット．イベントの情報は図13のSPARQLクエリによってRDFストアから動的に取得する．

```
PREFIX sdm: <http://sdm.hongo.wide.ad.jp/resource/>
PREFIX sdmo: <http://sdm.hongo.wide.ad.jp/sdmo/>
PREFIX schema: <http://schema.org/>

SELECT DISTINCT ?event ?eventName ?eventDate ?eventPlaceName ?eventPlaceAddress WHERE {
  VALUES ?playerClass {sdmo:Player sdmo:DataPlayer sdmo:AudioPlayer sdmo:VideoPlayer sdmo:CompositePlayer}
  ?player
    a ?playerClass ;
    schema:name "Web360Square" ;
    sdmo:plays ?event .
  ?event
    a sdmo:SDMEvent ;
    schema:name ?eventName ;
    schema:startDate ?eventDate ;
    schema:contentLocation ?eventPlace .
  ?eventPlace
    a schema:Place ;
    schema:name ?eventPlaceName ;
    schema:address ?eventPlaceAddress .
}
```

図13　Web360²におけるイベント情報取得クエリ

の反対方向に伸びている棘は常に視聴者の死角にある．死角にある棘状オブジェクトを図16のように非表示にすることで，32本の棘のうちレンダリングすべき棘はおおよそ20本程度になり計算処理を軽減できる．

3. 音声可視化オブジェクトの球体セグメント数の削減

音声可視化オブジェクトは中心に透明な球体をもち，この球体にタップ(クリック)による音声のON/OFF切り替え操作を紐付けている．球体自体は透明であるため球体に滑らかさは不要であり，セグメント数を削減することでポリゴン構造を荒くすることができる．ただし，あまりに構造が荒いとタップ(クリック)の反応範囲に違和感が現れるため，A-Frameのa-sphereオブジェクトのデフォルトセグメント数の半数に設定している．

```
PREFIX sdm: <http://sdm.hongo.wide.ad.jp/resource/>
PREFIX sdmo: <http://sdm.hongo.wide.ad.jp/sdmo/>
PREFIX geom: <http://data.ign.fr/def/geometrie#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX schema: <http://schema.org/>

SELECT DISTINCT ?playerClass ?contentUrl ?eventTime ?viewLabel ?startAt ?endAt ?x ?y ?z ?eulerDegX ?eulerDegY ?eulerDegZ ?eulerOrder WHERE {
  ?player
    a ?playerClass ;
    schema:name "Web360Square" ;
    sdmo:plays sdm:Billboard170126Event1 ;
    sdmo:inputFrom ?appMedia.
  VALUES ?appMediaClass {sdmo:Media sdmo:DataMedia sdmo:AudioMedia sdmo:VideoMedia sdmo:CompositeMedia}
  ?appMedia
    a ?appMediaClass ;
    schema:contentUrl ?contentUrl ;
    sdmo:inputFrom ?processor .
  VALUES ?plocessorClass {sdmo:Processor sdmo:Generator sdmo:Converter sdmo:Analyzer sdmo:CompositeProcessor}
  ?processor
    a ?plocessorClass ;
    sdmo:inputFrom ?originalMedia .
  OPTIONAL {
    ?appMedia
      sdmo:hasMediaEvent ?mediaEvent .
    ?mediaEvent
      a sdmo:MediaEvent ;
      sdmo:hasMedia ?originalMedia ;
      sdmo:eventTime ?eventTime .
  }
  VALUES ?originalMediaClass {sdmo:Media sdmo:DataMedia sdmo:AudioMedia sdmo:VideoMedia sdmo:CompositeMedia}
  ?originalMedia
    a ?originalMediaClass ;
    sdmo:inputFrom ?recorder ;
    sdmo:startAt ?startAt ;
    sdmo:endAt ?endAt .
  OPTIONAL {
    ?originalMedia
      rdfs:label ?viewLabel .
  }
  VALUES ?recorderClass {sdmo:Recorder sdmo:DataRecorder sdmo:AudioRecorder sdmo:VideoRecorder sdmo:CompositeRecorder}
  ?recorder
    a ?recorderClass ;
    sdmo:geometryAt ?geometry .
  ?geometry
    a sdmo:Geometry ;
    geom:coordX ?x ;
    geom:coordY ?y ;
    geom:coordZ ?z .
  OPTIONAL {
    ?geometry
      sdmo:eulerDegX ?eulerDegX ;
      sdmo:eulerDegY ?eulerDegY ;
      sdmo:eulerDegZ ?eulerDegZ ;
      sdmo:eulerOrder ?eulerOrder .
  }
}
```

図14　Web360[2]における視聴情報取得クエリ

映像と音声の同期処理

映像・音声のメディアファイルはそれぞれ独立しているため同期をとりながら再生する必要がある．映像の再生時間はJavaScriptのHTMLMediaElement.currentTimeプロパティによって取得・設定することができる．一方，音声についてはWeb Audio APIに読み取り専用のAudioContext.currentTimeプロパティが用意されているが，これは音声の再生時間ではなくWeb Audio のAudioContextが生成されてからの経過時間を示す．音声が停止中であってもAudioContext.currentTimeプロパティの値は単調に増加し続けるため，システム内では音声の再生・停止を管理するWeb Audio APIのAudioBufferSourceNode.start()メソッドおよびAudioBufferSourceNode.stop()メソッドが呼び出される際にAudioContext.currentTimeプロパティの値を記録するなどして音声の再生時間を算出している．

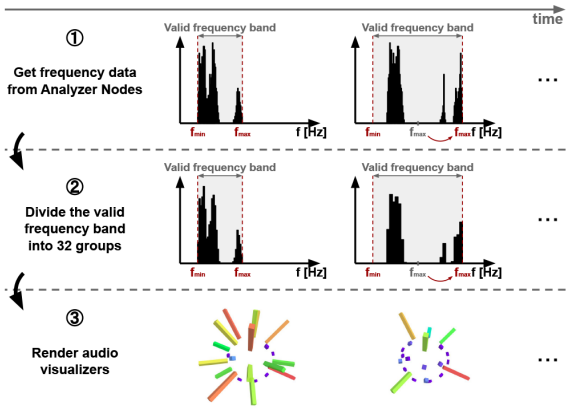

図15　音声を可視化する処理．1)周波数データを取得して有効周波数帯を更新する．2)有効周波数帯を可能な限り要素数が均等な32 グループに分けて，平均値を求める．3）32個の平均値と音声可視化オブジェクトの棘を対応させてレンダリングする．以上，1)から3)までの手順を毎秒20回程度のオーダーで繰り返しおこなう．

再生開始時やループ再生により先頭に戻った際には算出された音声の再生時間をもとに映像の再生時間を強制同期させる．また，視点を管理するA-Frameのカメラオブジェクトがもつtickハンドラによって毎秒60回から120回程度のオーダーで現在の音声の再生時間を算出し，映像の再生時間との差分を管理する．差分値に対して閾値を設定し，閾値を超えたときに映像の再生時間を音声の再生時間に強制的に同期させる対応は可能ではあるが，同期処理が入るたびに映像再生の滑らかさが失われ，QoEを損なってしまう．そこで，現行バージョンでは差分の閾値を0.1秒として，閾値よりも映像が遅れたら映像の再生速度を2倍にし，閾値よりも映像が先行したら映像の再生速度を0.5倍にすることで，映像と音声の再生時間のずれが徐々に閾値以内に収束するような比較的制約のゆるい同期をとっている．再生速度の変調は多少の違和感を生じてしまうが，映像再生の滑らかさを保つことができる．映像の再生速度はJavaScriptのHTMLMediaElement.playbackRateプロパティによって設定する．ここで，映像と音声が，映像の速度変調によるゆるい同期では時間を要するほど大きくずれてしまった場合，視聴者に与える違和感も大きくなってしまうため，やむを得ず映像の再生時間を音声の再生時間に強制的に同期させる．強制同期を発動する再生時間差分の閾値は1秒としている．

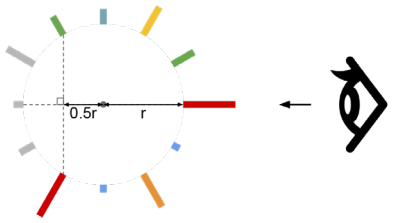

図16　死角にある棘状オブジェクトを非表示にする処理の概要．視聴位置から音声可視化オブジェクトの中心へと向かうベクトルを法線ベクトルとするような平面を音声可視化オブジェクトの中心から半径の半分だけ後方に置く．棘状オブジェクトの根本がこの平面よりも後方にある場合，死角にあると判定して非表示にする．

視聴位置の移動

視聴位置の移動は実験段階のβ機能である．現行バージョンでは，再生中の視聴位置以外の360°動画は読み込まず，視聴位置移動の操作があってから動画の読み込みが始まるためネットワーク環境に依存した遅延が起こる．この問題はすべての視聴位置の動画を事前に読み込んでおくことで解決することができるが，視点位置の数に比例して通信量が大きくなってしまう．このように視点移動の滑らかさと通信量はトレードオフである．

音声可視化オブジェクトをはじめとする視聴オブジェクトを正しい位置にレンダリングするために，映像・音声の座標情報などを取得している．360°動画を内部に投影するA-Frameのa-videosphereオブジェクトは常に視点を管理するカメラオブジェクトを中心とする性質がある．視聴位置の移動の実装の方法として2つの方針が考えられる．

1. カメラオブジェクトを映像の座標まで移動して，音声可視化オブジェクトは取得した座標情報通り配置する

2. カメラオブジェクトを固定して，音声可視化オブジェクトを映像の座標情報を原点とする相対座標で配置する

前者はカメラオブジェクトのみを移動すればよく単純であるが，映像の位置情報だけでなく姿勢情報も含めてカメラオブジェクトに適用する必要がある．カメラオブジェクトの姿勢が過度に傾いている場合，ジャイロセンサによる視聴角度操作が滑らかにおこなわれないことがあったため，Web360[2]では，カメラオブジェクトを固定する後者の音声可視化オブジェクトの相対座標配置を採用している．

3.2　LODチャレンジ

LODチャレンジ[*14]はLOD（Linked Open Data）の普及促進を目的に2011年から毎年開催されているイベントである．LODチャレンジ2020ではアイデア部門，アプリケー

＊14　https://2020.lodc.jp/

ション部門，データセット部門，データ分析・可視化部門，基盤技術部門の5つの部門について，2020年7月1日から10月18日まで作品募集しており，SDMコンソーシアムからはLODに興味・関心のある人々を中心に我々の活動を周知してもらう目的で，アプリケーション部門にWeb360[2]，データセット部門にSDM Ontology Version 2準拠のオープンデータを提出した．データセット部門へ提出したオープンデータについて，SDM Ontology Version 2による記述の構造的正確さや表現力が評価され，最優秀賞およびスポンサー賞の受賞に至った(図17)．

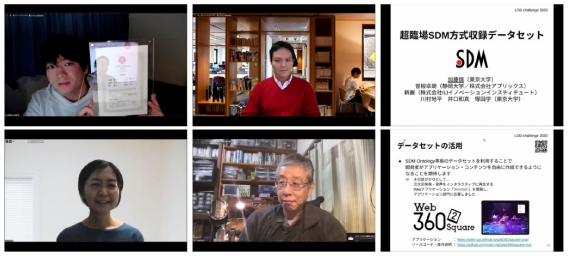

図17　授賞式の様子

第４章　BIM共通基盤とアプリケーション

この章では，施設のBIM化のための初めの一歩として行っている，3Dモデルの作成について触れた後，作成した3Dモデルに付与するメタデータやそれらデータを外部から参照するためのAPIについて活動報告を行う．最後に，共同研究先であるCIMXと共に製作している工場の可視化ツールについて説明を行う．

4.1　3Dマップ作成

学内には各建物の図面があり，一意の共通IDで部屋を指定することができるが，混雑度の情報提示のみならずさまざまな建物の管理情報を一元的に管理できるBIM(Building Information Modeling)が望まれている．その一環として，コンピュータ上に現実と同じ建物と立体モデルを構築するための3次元マップを作成する．この節では，3Dモデルの作成について触れていく．三次元データ作成にMatterport[*15]というクラウドサービスを使用している．Matterportは3Dスキャニングが行え，専用カメラを用い，撮影した場所をサービス上で三次元データ，点群を作成できるクラウドサービスである．これを用い，東京大学工学部二号館，エービー白河事業所，またNTTCOMスマートシティラボについて撮影を行った．

4.1.1　東京大学キャンパスマップ

Matterportを用い，東京大学工学部二号館のB1Fから13Fまでを撮影した．ただし，4F，10F以外の階では階段からエレベーターホールまでの短い区間のみを撮影した．4Fでは，それらに加えて講義室や実験室を撮影し，10F

図18　東京大学工学部二号館4F

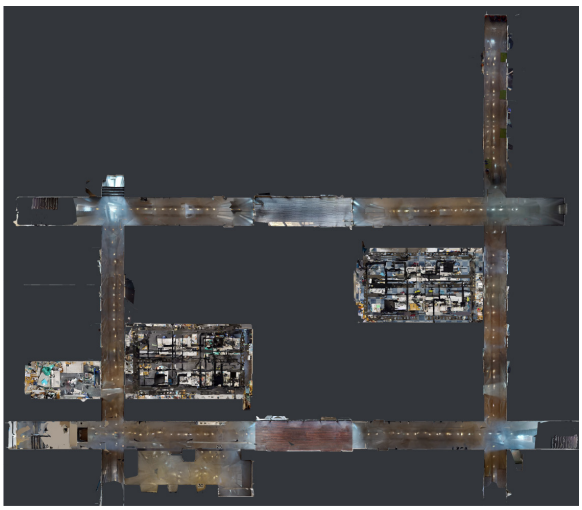

図19　東京大学工学部二号館10F

＊15　https://matterport.com/

では，研究室を撮影した．図18は，撮影を行った工学部二号館4Fのデータを，図19は，撮影を行った工学部二号館10Fのデータを上部から見た画像である．4Fの撮影には約5時間（撮影回数:350回），10Fの撮影には約2時間（撮影回数:120回）かかった．撮影した講義室のデータの一部を図20に示す．このように，Matterportで生成できるファイルは，簡単なメッシュデータとテキスチャデータである．これらの3Dモデルを用いて，3Dマップを構築していく．撮影に使用したMatterportPro2というカメラでは，一度に35mの範囲を撮影できるのだが，図18を見てもらえばわかるように，長距離の撮影であると，誤差が蓄積してしまい，図の上部と下部で閉ループが構築できないという問題が発生してしまっている．

4.1.2　エービー白河事業所マップ

BIMの活用分野の一つにファクトリーオートメーションが挙げられる．本節では，工場内の情報の可視化等に活用するため作成した3Dモデルについて説明する．撮影を行なった場所はエービー白河事業所で，ここではプラスチック金型制作などが行われている．工場内には大型の機械などが多く，オクルージョンが多く発生したため，撮影には多くの地点を要した．撮影に要した時間は4時間半程で，139地点での撮影情報を用いた．に工場を上から見た画像を示す．また，工場内には環境センサやLidarセンサが複数箇所に設置されており，温度，湿度，照度，騒音や機械の電力といった情報を取得することが出来る．

図20　241講義室

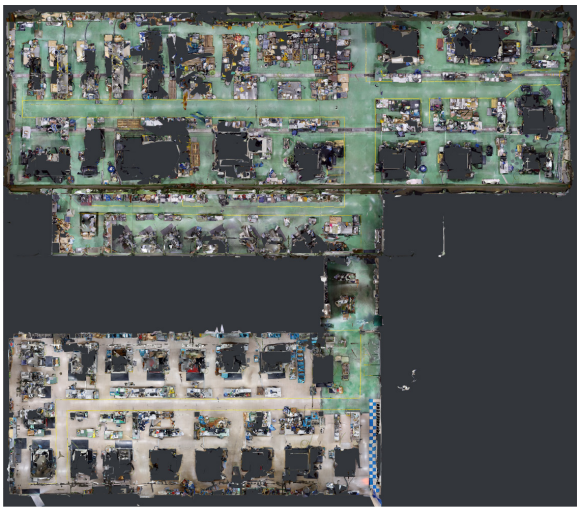

図21　エービー白川事業所

4.1.3　NTTCOMスマートシティラボのマップ

東大グリーンICTプロジェクト（GUTP）[*16]にてディジタルツインアプリケーション開発のためのワークフローの検討および実証実験が進められている．ディジタルツインアプリケーション開発の最初の段階として，今回NTTCOMスマートシティラボにて撮影を行なった．この撮影では，Matterport Pro2に加え，BLK360も用いられた．両者で撮影をしえられた点群の比較では，主に密度に差があり，物体の形状などはBLK360の方が精度が上であった．例えば，床に貼られたテープが，BLK360では一直線となっていたが，Matterport Pro2では途中で途切れるなどしていた．一方で，今回撮影を行なった規模

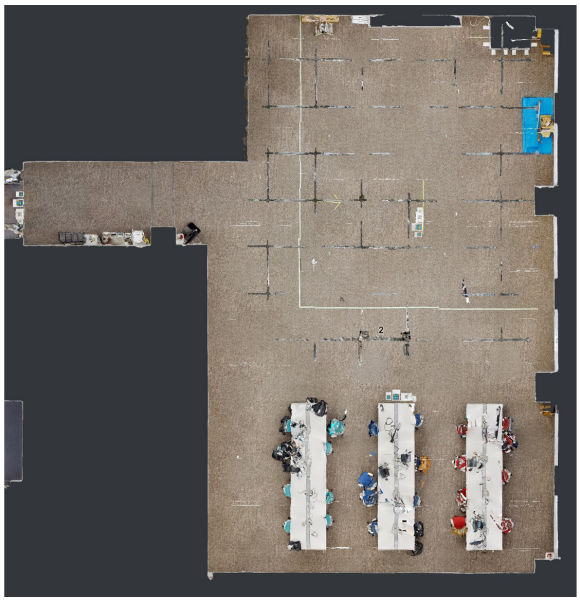

図22　NTT Com実証実験ラボ

＊16　https://www.gutp.jp/

では，部屋全体の精度としてはそれほど大きな差は出なかった．今回の撮影に要した時間は1時間ほどで，撮影地点は48箇所であった．今後，この3Dモデルを用いてディジタルツインアプリケーションの開発を進めていく．

4.2　3DマップAPIとオントロジ

4.1.1にて作成した東京大学工学部2号館の3Dマップに建物情報などをメタデータとして付与し，外部から情報を取得するAPIを作成した．

4.2.1　オントロジ

工学部2号館の内の位置関係は，Building Topology Ontology（BOT）[68]を用いて記述した．BOTは建物の構造情報を表すオントロジーであり，今回は主にキャンパス，建物，フロア，部屋の位置関係を記述している．また，各部屋に対して，部屋の名称，ID，glbファイルのパスを与えている．

4.2.2　3DマップAPI

前節で記述したメタデータ及び部屋の画像を取得することのできるAPIをDocker内にFastAPIを用いて作成した．このAPIは，キャンパス内の混雑具合を表示するアプリケーションのMOCHA[*17]などと連携することを想定している．3Dマップからの画像の取得はpythonを用いた手法とUnityによる手法の2種類の手法を用いて実装を行なった．Pythonを用いた実装では，VM上にXvfbによる仮想ディスプレイを作成し，pygameを用いてobjファイルを読み込み，指定された視点からの仮想ディスプレイのキャプチャを行なっている．視点の向き，視点の位置，部屋はPOSTリクエストにより指定している．

Unityを用いた実装では，FastAPIを通したPOSTリクエストからUnityへアクセスする．POSTリクエストには，上部から撮影するか斜部から撮影するかといった情報やどの部屋を撮影するかといった情報を含んでいる．POSTリクエストを受け取った後は，Xvfbを用いて仮想ディスプレイを作成し，ビルド済みのUnityアプリケーションデータを用いて3Dモデルを配置し，キャプチャを行い画像ファイルを生成する．最後にその画像ファイルをBase64形式でユーザに返却するといった構成になっている．

4.3　Factory Automation関連アプリケーション

株式会社CIMXと共にFactory Automation（FA）について，Matterportや各種センサを用いて行えることを模索している．この節では，FAの初めの一歩として，現在作成している工場の可視化ツールについて説明していく．この可視化ツールでは，4.1.2で紹介したエービー白川事業所と提携しており，3Dモデルと機械や通路に設置したIoTセンサを利用している．

4.3.1　工場可視化ツール

可視化ツールはUnityを用いて作成しており，図23に示したUIを持つアプリケーションである．これは，鹿島建設株式会社が作成したリアルタイム現場管理システム「3D K-Field」[*18]を参考にし作成しているが，これに加えてBIMのような詳細なメタ情報が含まれていない3Dモデルからもこの可視化ツールを利用できる簡便なシステムを目指している．工場では，いくつかの区画に分かれており，このアプリケーションでは，Matterportを用いて作成した3Dモデルを与えられた空間情報に応じて，分割し表示することが出来る．そして，その3Dモデルの上に，Unity上で自動生成された各種センサを示すオブジェクト（図23上では，赤いオブジェクトで示されている）やLiDARセンサで取得した点群データ24を重ねて表示する．これに加えて，環境センサから取得した温度情報を利用して，温度ヒートマップを同空間上に描画すること（図26）や，同

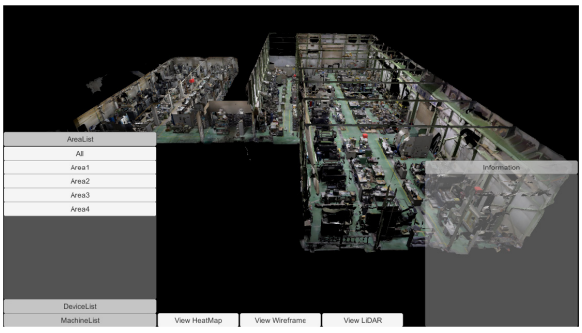

図23　工場可視化ツール

＊17　https://mocha.t.u-tokyo.ac.jp/
＊18　https://www.kajima.co.jp/news/press/202010/28a1-j.htm

モデルの描画方法を変更し，空間上でのセンサデバイスの位置を見やすくしたワイヤーフレーム表示(図27)などが行えるようになっている．このような3Dでの描画のみならず，実際に工場内に設置された各種センサにアクセスし，温度，湿度，照度，騒音，電力，信号灯などと言ったデータをリアルタイムに取得，表示することができる(図25)．

4.3.2 REST API

この各種センサにアクセスするためのAPIは株式会社CIMXが作成しており，REST APIの形式で構築されている．このAPIの一部を図28に示す．このAPIでは，APIのバージョンとセンサの種類，センサーのIDと取得時間(直近，指定した30分間など)にて取得する情報を指定することが出来る．アプリケーションでは，起動と同時にAPIを使用するため，与えられたユーザでログインし，その後に各種センサ情報を取得し，アプリケーション終了時にログアウトする形で利用している．

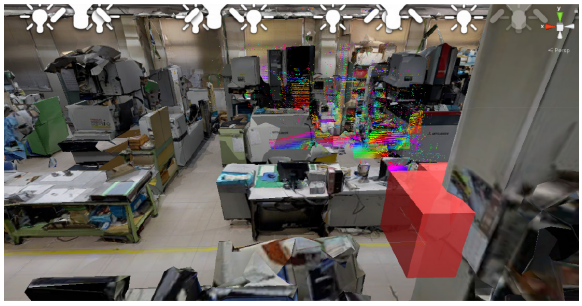

図24　点群データの可視化

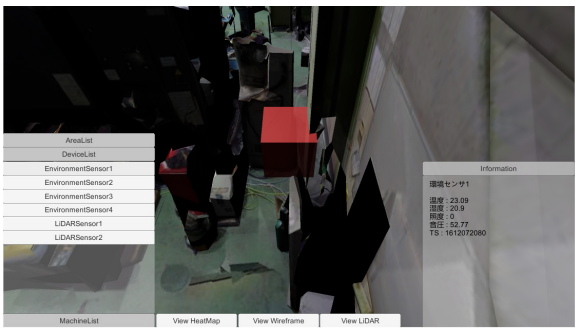

図25　点群データの可視化

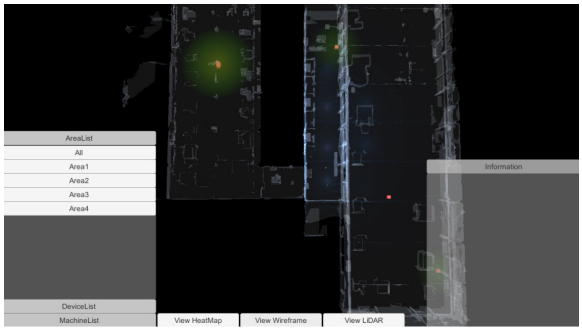

図26　温度ヒートマップの表示

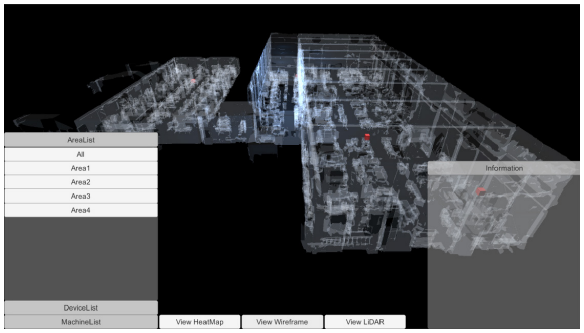

図27　ワイヤーフレーム表示

/api/v0_0/{env}/{sensor_id}/{interval}/{start_time}/{finish_time}

パラメータ

sensor_id : センサの識別 ID
start_time : データ取得開始時刻のタイムスタンプ
finish_time : データ取得開始時刻のタイムスタンプ

env	temp	温度
	humd	湿度
	light	照度
	noise	騒音

interval	minute	1 分毎の測定値
	halfHour	30 分の平均値
	hour	60 分の平均値
	day	1 日の平均値
	month	1 ヶ月の平均値

例)
/api/v0_0/temp/1/minute/1607572800/1607659200

応答

項目	内容	型
CUST	顧客 ID	str
area_id	（対象建造物内を分割した）エリアの ID	str
sensor_id	各センサの識別 ID	str
DEV	センサの MAC アドレス	str
unit	測定値の単位	str
posX	センサの X 座標	float
posY	センサの Y 座標	float
posZ	センサの Z 座標	float
TS	データ計測時刻	int
val	センサ測定値	float
status	応答のステータス	str

図28　工場可視化ツールREST API

4.3.3　利用方法

このアプリケーションは，WebGLやiOSなどと言ったスマート端末向けにデプロイ可能となっており，ユーザが株式会社CIMXが用意したサーバにデプロイされたWebアプリやiPadなどを用いて閲覧することを想定している．このアプリケーションを用いて，工場内の情報を現在の状態はもちろん，履歴から過去の情報を可視化することや，温度ヒートマップやLiDARからの点群情報を用いて動的に変化するデータを3D空間上で閲覧することが出来る．

4.3.4　今後について

4.3.1で記述したように，詳細なメタ情報を含まない3Dモデルからもこのシステムを利用可能にすることを一つの目的としている．そのため，今回エービー事業所をモデルとしてこのツールを作成しているが，今後4.1.3で記述したNTTCOMスマートシティラボや美術館などといった様々な場所をモデルとして，このツールが転用可能であることを示していく予定である．

第５章　まとめ

本報告書では，2014年より開始したSDMコンソーシアムで進めて来た，視聴空間サービスのソフトウェア制御による研究で，2020年度に行なった活動について報告した．

SDMコンソーシアムでは，ソフトウェア処理による視聴空間の制御，映像音声を制御するネットワーク機器，インタラクティブ・ユーザインターフェイス，SDMプラットホームを利用したコンテンツ作成など，共同研究活動に参加するパートナーを募集しています．

ご協力をよろしくお願いいたします．