**ISO/IEC JTC 1/SC 24/WG 9 "Augmented reality continuum concepts and reference model"**
Convenorship: **KATS**
Convenor: **Kim Gerard Jounghyun Mr**

# Information Model for Mixed and Augmented Reality (MAR) Contents Part 3: Live actor and entity

| Document type | Related content | Document date | Expected action |
|---|---|---|---|
| Meeting / Working documents for discussion | Meeting: VIRTUAL 21 Jul 2021 | 2021-11-03 | |

Reference number of working document: ISO/IEC JTC 1/SC 24 N 0000

Date: 2021-07-21

Reference number of document: ISO/IEC CD 0000

Committee identification: ISO/IEC JTC 1/SC 24/WG 9

Secretariat: Charles A. Whitlock

# Information technology — Computer graphics, image processing and environmental data representation —

# Information Model for Mixed and Augmented Reality (MAR) Contents Part 3: Live actor and entity



Document type: International Standard
Document subtype: subtype
Document stage: (20) Committee
Document language: E

STD Version 1

# Contents

## 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.

ISO/IEC 18040 was prepared by Joint Technical Committee ISO/IEC JTC 1, Information technology, Subcommittee 24, Computer graphics, image processing and environmental data representation.

## Introduction

Mixed and Augmented Reality (MAR) refers to spatial coordination of components, systems, and services which enabling the connection between physical real world and virtual objects working relatively. This is a conceptual way that provides the rich experience based on realism in field of computer graphic. In order to reach the goal of implementation, it requires several steps to generate a completed MAR system.

In the purpose of this standard, the architectural concept is to concentrate on Mixed and Augmented Reality and to describe the Live Actor and Entity (LAE) in MAR content. Through the LAE representation, system is needed to be designed and applied the MAR conceptions with the specific requirements such as sensors, context analyser, computational simulation, and display system. Since the researches of machine learning growing rapidly, the decision of choosing this technology is a powerful way to force the performance of system implementation. Above all is to facilitate LAE representing and controlling in MAR system. Therefore, the enhancement of integrating LAE representation in MAR is to consider on information model and implementation properties for LAE contents.

There are requirements to achieve in the concept of representing LAE in MAR scene/contents. The following listed requirements used to describe the main objectives of the document of entire LAE-MAR conception.

- Demonstrate the LAE-MAR system architecture
- Define the models and properties of LAE
- Define definitions of node of LAE information model

Information technology — Computer graphics, image processing and environmental data representation —

# Information Model for Mixed and Augmented Reality(MAR) Contents Part 3: Live actor and entity

## 1. Scope

This document has an objective to extend the previous research project and existing standard for improving the LAE information models on Mixed and Augmented Reality scene/contents description. This extension enhancing the capabilities of LAE-MAR system more reliable and putting system in advance stage of development.

(1) Improving LAE-MAR system working more effectively

(2) Allowing deep learning techniques to involve in system process

(3) Using Virtual Reality (VR) technology to extend the immersive experience to user

(4) Interaction ability between LAE and MAR models

(5) Standardization of using LAE-MAR system with defined nodes structure

## 2. Normative references

Since this is the extensional document, it relies on the MAR Reference model (ISO/IEC 18039), and live actor and entity representation in mixed and augmented reality(ISO/IEC 18040)

ISO/IEC 18039, *Information technology — Computer graphics, image processing and environmental data representation — Mixed and Augmented Reality Reference Model, January 2017*

ISO/IEC 18040, *Information technology — Computer graphics, image processing and environmental data representation — Live Actor and Entity Representation in Mixed and Augmented Reality, January 2017*

## 3. Terms and definitions

This standard document is based on the improvement of terms and definitions defined in ISO/IEC 18040, *technology -- Computer graphics and image processing –Live actor and entity representation in Mixed augmented reality*. Therefore, the following terms and definitions apply.

### 3.1 Description of technical terms as already defined in the other related standards

#### 3.1.1. Augmentation

Virtual object data (computer-generated, synthetic) added onto or associated with target physical object data (live video, a physical world image) in a MAR scene. Equally applies to physical object data added onto or associated with target virtual object data.

[ISO/IEC 18039:2016, definition 3.1]

#### 3.1.2. Augmented object

An object with augmentation.

### 3.1.3. Augmented reality system

Type of mixed reality system in which virtual world data is embedded and/or registered with the representation of physical world data.

[ISO/IEC 18039:2016, definition 3.2]

### 3.1.4. Augmented virtuality system

Type of mixed reality system in which physical world data is embedded and/or registered with the representation of virtual world data.

[ISO/IEC 18039:2016, definition 3.3]

### 3.1.5. Display

Device by which rendering results are presented to a user. It can use various modalities, such as visual, auditory, haptics, olfactory, thermal, motion, etc. In addition, any actuator can be considered a display if it is controlled by a MAR system.

[ISO/IEC 18039:2016, definition 3.4]

### 3.1.6. Dynamic object

An object which can be translated, rotated, and scaled in a physical world or a virtual world.

### 3.1.7. Feature

Primitive geometric elements (e.g., points, lines, polygons, colour, texture, shape, etc.) and/or attributes of a given (usually physical) object used in its detection, recognition, and tracking.

[ISO/IEC 18039:2016, definition 3.5]

### 3.1.8. Geographic coordinate system

[ISO/IEC 18040:2019, definition 3.1.8]

A coordinate system which is provided by sensor devices for defining a location of LAE.

### 3.1.9. Head mounted display (HMD)

[ISO/IEC 18039:2019, definition 3.5]

A device which displays stereo views of virtual reality, such as Samsung Gear VR, Oculus Rift, Google Cardboard, etc. It has two small displays with lenses and semi-transparent mirrors which can adapt to the left and right eyes.

### 3.1.10. Live actor and entity (LAE)

A representation of a living physical or real object, such as a human being, animal, or bird, in the MAR content or system. A live actor can be animated, moved, and interacted with virtual objects in a MAR scene by capturing gesture from a camera. Entity refers to 3D objects and entities that exist in MAR content.

[ISO/IEC 18040:2019, definition 3.5]

### 3.1.11. LAE recognizer

A MAR component that recognizes the output from a LAE capturer and a LAE sensor, then generates MAR events based on conditions indicated by the content creator.

### 3.1.12. LAE capturer

A MAR component that captures a LAE in a virtual world and a physical world, which includes depth cameras, general cameras, 360º cameras, etc. A LAE's information will be processed by a LAE recognizer and LAE tracker to extract background or skeleton.

### 3.1.13. LAE sensor

A device that returns values related to a detected or measured condition or property related to a LAE. A LAE sensor may be an aggregate of LAE sensors.

[ISO/IEC 18039:2016, definition 3.20]

### 3.1.14. LAE tracker

A MAR component (hardware and software) that analyses signals from LAE capturers and sensors and provides some characteristics of a tracked LAE (e.g., position, orientation, amplitude, profile).

### 3.1.15. MAR event

An event which is triggered by the detection of a condition relevant to MAR content and augmentation (e.g. detection of a marker).

[ISO/IEC 18039:2016, definition 3.6]

### 3.1.16. MAR execution engine

A collection of hardware and software elements that produce the result of combining components that represent, on the one hand, the physical world and its objects, and on the other hand, those that are virtual, synthetic, and computer generated.

[ISO/IEC 18039:2016, definition 3.7]

### 3.1.17. MAR experience

The human visualization of and interaction with a MAR scene.

[ISO/IEC 18039:2016, definition 3.8]

### 3.1.18. MAR scene

The observable spatiotemporal organization of physical and virtual objects. It is the result of a MAR scene representation being interpreted by a MAR execution engine. A MAR scene has at least one physical object and one virtual object.

[ISO/IEC 18039:2016, definition 3.9]

### 3.1.19. MAR scene representation

A data structure that arranges the logical and spatial representation of a graphical scene, including the physical and virtual objects that are used by the MAR execution engine to produce a MAR scene.

[ISO/IEC 18039:2016, definition 3.10]

### 3.1.20. Mixed and augmented reality system

Term synonymous with *mixed reality system*[1].

[ISO/IEC 18039:2016, definition 3.11]

[1] The word "augmented" is often used together with the word "mixed".

### 3.1.21. Mixed reality continuum

Spectrum spanning physical and virtual realities according to a proportional composition of physical and virtual data representations (originally proposed by Milgram et al. [1]).

[ISO/IEC 18039:2016, definition 3.12]

### 3.1.22. Mixed reality system

A system that uses a mixture of representations of physical world data and virtual world data as its presentation medium.

[ISO/IEC 18039:2016, definition 3.13]

### 3.1.23. Marker

In the context of a MAR system, a marker consists of metadata embedded in a MAR background that specifies the location of a superimposed object.

[ISO/IEC 18039:2016, definition 3.27]

### 3.1.24. Movable volume

A volume in which a LAE is movable in a physical world or in a virtual world.

### 3.1.25. Natural feature

A feature that is not artificially inserted for the purpose of easy detection/recognition/tracking.

[ISO/IEC 18039:2016, definition 3.14]

### 3.1.26. Physical Camera coordinate system

A coordinate system which is provided by a camera for capturing LAE(s) in physical world.

### 3.1.27. Physical object

A physical object that is designated for augmentation with virtual data representation.

[ISO/IEC 18039:2016, definition 3.15]

### 3.1.28. Physical reality

Term synonymous with the physical world itself or a medium that represents the physical world (e.g., live video or a raw image of the physical world).

[ISO/IEC 18039:2016, definition 3.16]

### 3.1.29. Physical world

Spatial organization of multiple physical objects.

[ISO/IEC 18039:2016, definition 3.17]

### 3.1.30. Point of interest

A single target location or a collection of target locations. Aside from location data, a point of interest is usually associated with metadata, such as an identifier and other location specific information.

[ISO/IEC 18039:2016, definition 3.18]

### 3.1.31. Physical coordinate system

A coordinate system that enables locating a LAE and is controlled by a geospatial coordinate system sensing device.

### 3.1.32. Spatial registration

The establishment of the spatial relationship or mapping between two models, typically between a virtual object and a target physical object.

[ISO/IEC 18039:2016, definition 3.21]

### 3.1.33. Static object

An object which cannot be translated, rotated, or scaled in a physical world or a virtual world.

### 3.1.34. Target image

A target object represented by a 2D image.

[ISO/IEC 18039:2016, definition 3.22]

### 3.1.35. Target object

A target physical object designed or selected to allow detection, recognition, and tracking (and, finally, augmentation).

[ISO/IEC 18039:2016, definition 3.23]

### 3.1.36. Virtual actor and entity

A virtual reality representation of a LAE. The virtual actor and entity are obtained by a 3D capturing technique and can be reconstructed, transmitted, or compressed in the MAR scene. A virtual actor and entity can be captured in one place or transmitted to another place in real time using holography technology.

### 3.1.37. Virtual object

A computer-generated entity that is designated for augmentation in association with a physical object data representation. In the context of MAR, it usually has perceptual (e.g., visual, aural) characteristics and, optionally, dynamic reactive behaviour.

[ISO/IEC 18039:2016, definition 3.25]

### 3.1.38. Virtual world or environment

Spatial organization of multiple virtual objects, potentially including global behaviour.

[ISO/IEC 18039:2016, definition 3.26]

### 3.1.39. World coordinate system

A universal system in computer graphics that allows model coordinate systems to interact with each other.

## 3.2. Abbreviated terms

### 3.2.1. HMD

Head mounted display

### 3.2.2. LAE

Live actor and entity

### 3.2.3. LAE-MAR

Live actor and entity representation in mixed and augmented reality

### 3.2.4. MAR

Mixed and augmented reality

[ISO/IEC 18040:2017]

### 3.2.5. VAE

Virtual actor and entity

### 3.2.6. VR

Virtual reality

### 3.2.7. VTG

Virtual transform group

### 3.2.8. SM

Spatial Mapper

### 3.2.9. EM

Event Mapper

# 4. Concepts

## 4.1. Overview

The system mainly focuses on Live Actor and Entity (LAE), which is a presentation of physical actors or objects appearing in MAR scene/content and has the abilities to work interactively between real-world and virtual environment. Actor or Entity performs as a character who can be animated, moved, or transformed based on defined mapping algorithm. In addition, actors can demonstrate with many different activities like interaction, which an actor can interact with virtual objects in MAR scene. Since previous work, the actor is captured form real-world target object by advanced functional camera like depth camera. However, on this document, there is a reliable improvement, which is using deep learning technique to capture a target object. Moreover, the information getting from cameras or sensors are used to be analyzed, tracked, and embedded into MAR scene. The LAE representation is not only performed as human, but it could be defined as different objects such as cat, dog, bird, and etc.

To facilitate the accessibility of Virtual Reality (VR) technology in order to allow the user experiencing the immersive world, the MAR scene is designed as 3D virtual world, that can be described in sense representation. For LAE representation in MAR scene can be rendered as 2D or 3D according to camera or deep learning model. Many examples that LAE-MAR system can be applied to such as TV talk show studio, news broadcast studio, education services, virtual surgical operations, games, and any kinds of demonstration performing as 3D virtual scene. Importantly, the system has possibility to run and render the scene/content in real-time process. Interestingly, the user can experience with exciting scene displayed in another virtual world and be intractable to virtual objects like in real actions by using Head Mounted Display (HMD) device and its controllers. Also, LAE representation can be represented in MAR scene as a Virtual Actor and Entity (VAE).

## 4.2. LAE-MAR system

Definitely, the process of system is described based on standard document. However, to be easily understandable, the summarized architecture has been designed as below. The important role of the entire system is to connect between physical world and virtual world working perfectly with high-performed simulator and to make real display and actions to user.

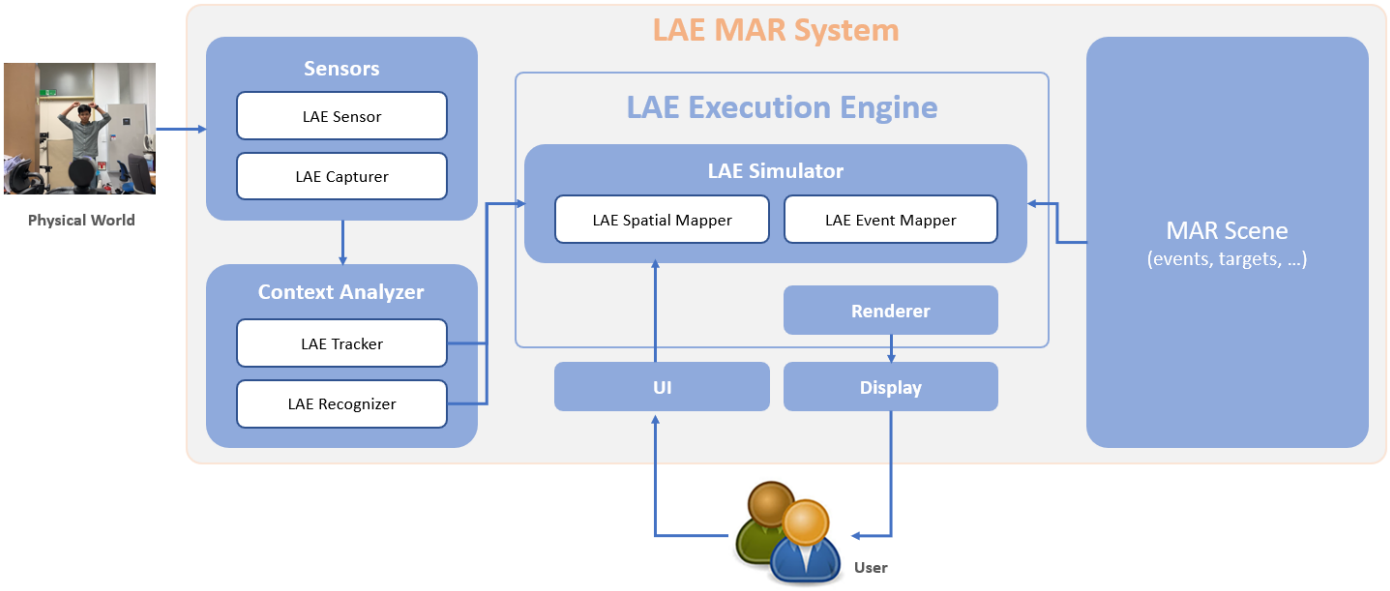



*Figure 1, Representing the whole architecture of LAE-MAR system*

For LAE representation in the MAR scene, the system process is working consequently in order to transmit or pass the information through every phase like sensors or content analyzer. Next, mappers are mainly handling full services to connect between receiving information and defined environment scene. Finally, the system provides a smoothly relatively display to users.

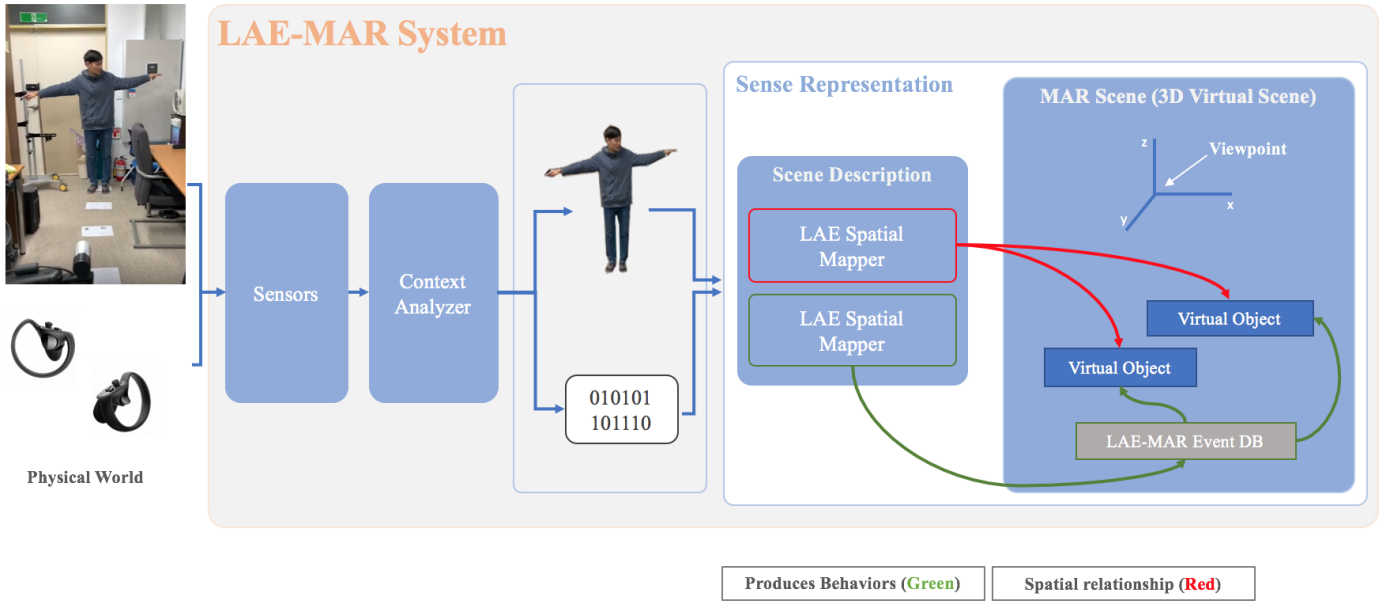
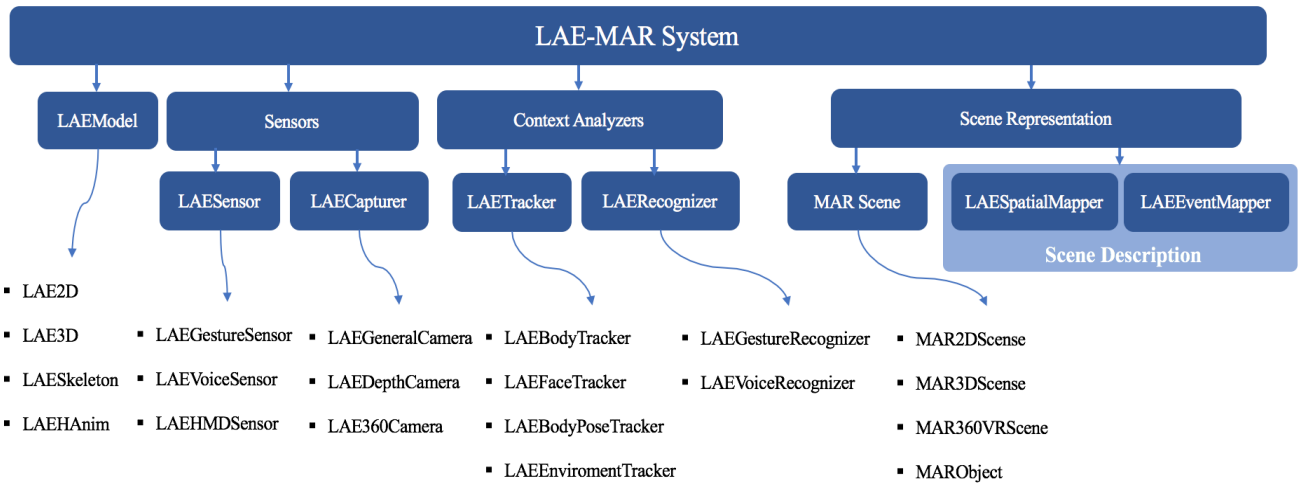



*Figure 2: This is how the system processing with captured information and handling it to final display scene*

### 4.3. LAE-MAR components

There are some updates on previous document and also components are extended from standard model of LAE. Since this is the improvement of implement of the project, there are some properties, data types and information model defined differently. By enabling deep learning techniques to be involved in the system, the LAE representation is described in another way over previous defined variables. The following Figure shows the levels of components which are used to implement the LAE representing in MAR scene.



### 4.3.1. General

In overall, the system constructed by different parts known as components which used to function separately to handle the process of allowing the virtual world objects to perform as same as any objects in physical world. There are various components proposed in this system so each part and relation of entire process is briefly described below.

### 4.3.2. Relation of components

In LAE System, there are several components defined to make a complete system that allows the relation between real-word object and virtual object working respectively. The following diagram shows the relation of each components in order to wholly understand how real world and virtual world are mapped.

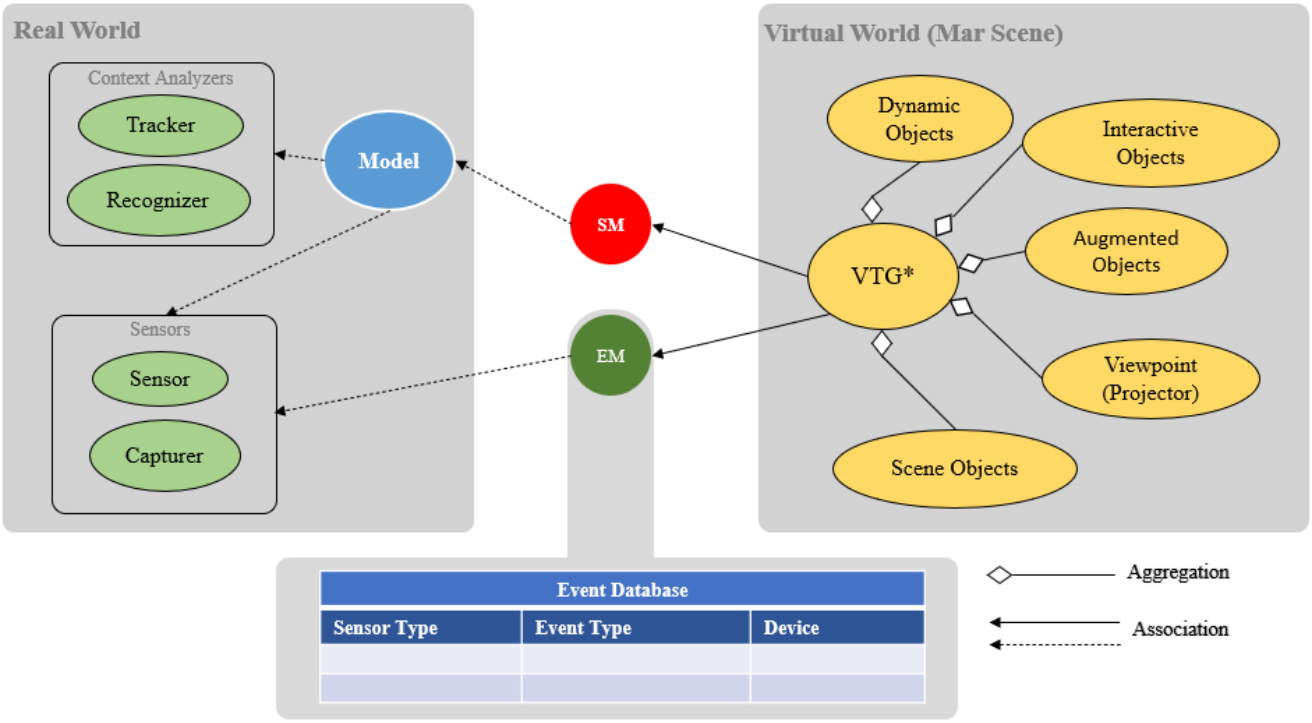



*Figure 4: Connection between real and virtual world in LAE-MAR system*

### 1.1.1. LAE capturer

Basically, the system necessarily needs to access directly to the connected camera that it can be applied with any types of camera such as depth camera, general camera, 360º camera, etc. Sets of images consequently captured and controlled by LAE capturer component for providing service to other components like LAE tracker and LAE recognizer.

### 1.1.2. LAE sensor

By receiving information from the real world, the system is equipped with sensors to make access on different information and its target is on sensing information from attached sensors and used the sensing data for a specific behaviour or event.

### 1.1.1. Context analyzer

This is one of important aspects that considered as a component in charge of analysing, evaluating or learning from sensing data or captured data. This context analyzer consists of LAE tracker and LAE recognizer.

#### 1.1.1.1. LAE tracker

As an extended component from context analyzer, LAE tracker purposely identified as a component to track the needed information from captured data or sensing data based on the target applied. The tracker can be existed with many different types of tracking techniques. For instance, the LAE tracker can be assumed to track the object from image or to track the voice from audio.

#### 1.1.1.2. LAE recognizer

The process of LAE recognizer is fundamentally similar to tracker but it is responsible on recognizing sensing information and map it to the events defined in scene description to perform in virtual scene. Through the recognizer, the manners or events in virtual scene can be determined in many different ways according to sensing data mapped.

### 1.1.2. LAE model

In this context, the model is used as abstract object which is composed of single or several component(s), sensor or context analyzers, in order to create a correct model for mapping it into a virtual object. Actually, in LAE system, the model can be made up by different sensor and context analyzer as needed. Also, there are several available models and it can be created following by the pre-defined models. Each model has to serve different purpose.

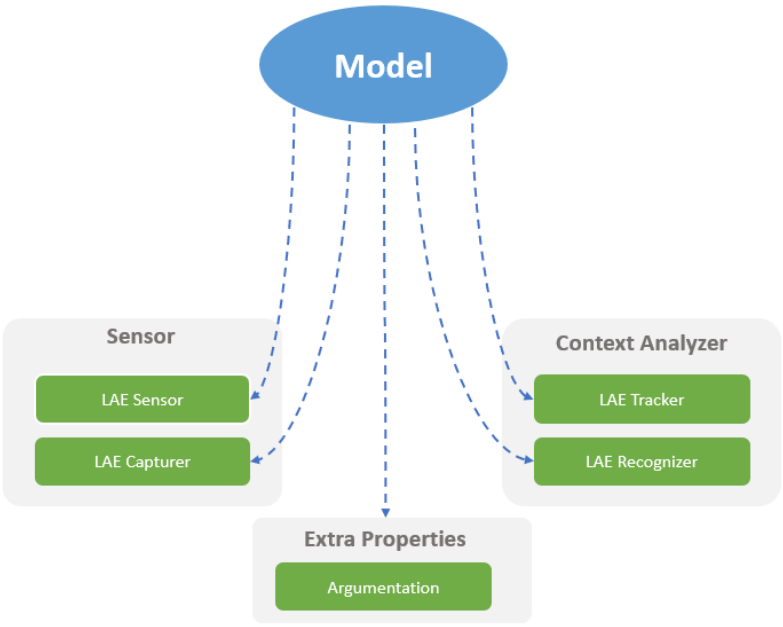



*Figure 5: The construction of a model in LAE-MAR system*

### 1.1.3. Scene representation

The mappers considered as potential components since their main functionality is to map between the real-world objects and virtual scene objects.

#### 1.1.3.1. Scene description

#### 1.1.3.2. LAE spatial mapper

The Spatial Mapper is an association class among Transform Group classes that explicitly specifies spaces or objects are spatially related within the virtual scene structure. Always, one object is designated to be the target object to be augmented.

Here below are functionalities of Spatial Mapper

- Mapping between real world to virtual world
- Position
- Rotation
- Scale

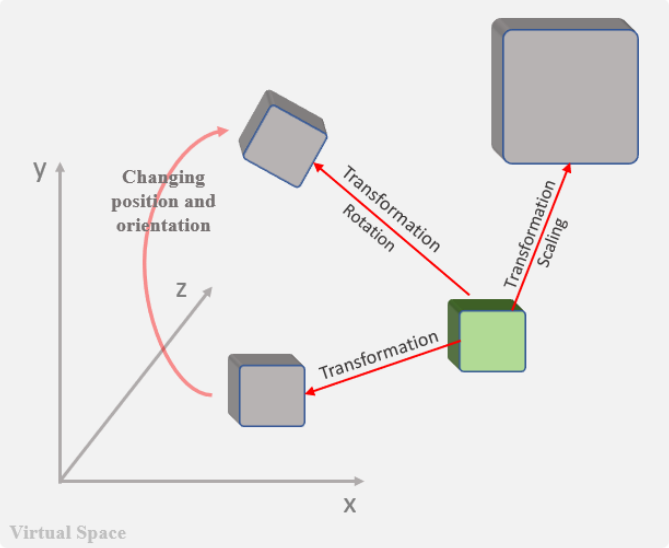



*Figure 6: The functions of LAE spatial mapper*

#### 1.1.3.3. LAE event mapper

Like any interactive systems, behavior and its simulation are driven by various events and data as produced by the system components through sensor-based environment monitoring and user interaction. that all of this produces the events and data stream need to drive the behavior or action of the object(s).

Here below are functionalities of Event Mapper

- Mapping between real world to virtual world
- Behavior in virtual scene performs relatively to the information defined in the **Event Database**

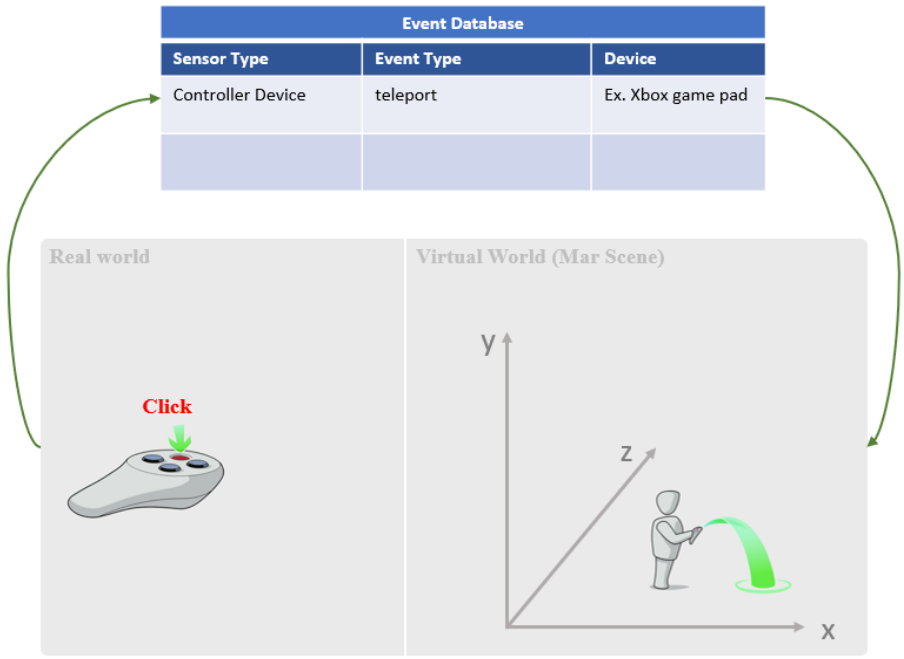



*Figure 7: The functions of LAE event mapper*

#### 1.1.3.4. MAR scene

Designing the entire virtual environment with static or dynamic virtual objects can be organized by MAR scene. In MAR scene, it has several types of components such as dynamic object, augmented object, scene object, interactive object and viewpoint.

##### 1.1.3.4.1. Mar scene components

Designing a decent virtual scene, there should be many various objects available to create the environment dynamically and flexibly. Each component stands in different level and performs in the way of its design.

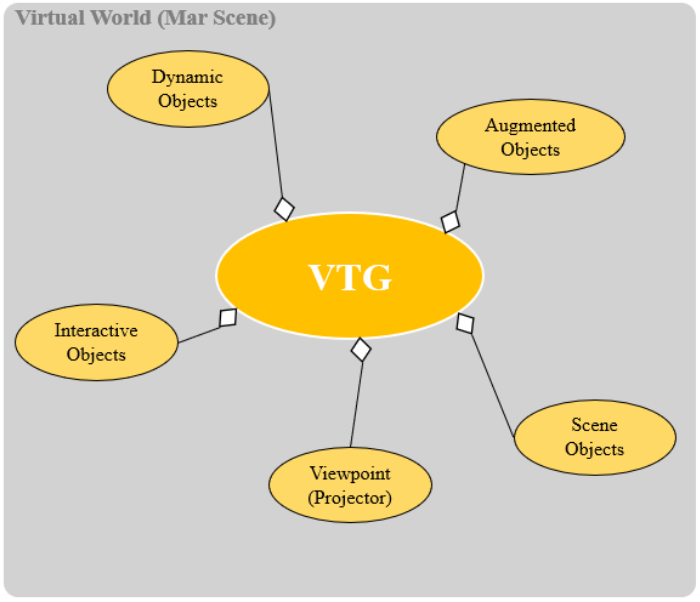



*Figure 8: Model components in mar scene*

**Virtual Transform Group** is an extendable parent class used to represent the group of virtual objects.

**Scene Object** is a stand-still object which has no possibility to be updated or modified.

**Dynamic Object** is a subclass of object of VTG that object can be performed dynamically depending on the definition of mapped LAE model.

**Interactive Object** is also an object inherited from VTG, but the object is dynamically changed or updated. Moreover, the object is interactable by other targeted objects.

**Augmented Object** is an object with augmentation that is designed in a particular feature of making object in virtual scene more interesting like animation, effect, or any behaviours.

**Viewpoint/Projector** is a projection of camera to capture the story in the virtual scene. There can be more than one projector used to project in entire scene.

#### 1.1.3.5. MAR scene object

By just allowing a possibility to the system to decorate the virtual environment, the system can use MAR scene objects to construct the whole virtual world by any available 3D models designed in scene. In runtime, the MAR scene object cannot be dynamically updated, but this object model is specifically used for the fixed object model which has no constraint to any actions or events.

#### 1.1.3.6. MAR dynamic object

As an extendable object, MAR object gives the permission to the system to freely create the pre-defined models in virtual scene. Pre-defined model is considered as an essential privilege that system can demonstrate the different kinds of model which perform different ways and purpose.

#### 1.1.3.7. MAR interactive object

Interactive object is also used in MAR scene to design the virtual world. Through pre-defined live actor model, the virtual objects can be interactable in a purpose of making live actor to interact directly with any interactable objects in form of events or actions.

#### 1.1.3.8. MAR augmented object

The functionality of augmented object is that the system allows to illustrate the augmenting and animating object model which the effects can be shown corresponding to the augmented object defined in the system.

### 4.4. Scene graph

In order to facilitate the implement of how to design/create LAE system, the scene graph has been defined simply. In the scene graph, every component is considered as Node, which are declared in form of HTML for basic understanding by just using element ID for linking relation between Nodes.

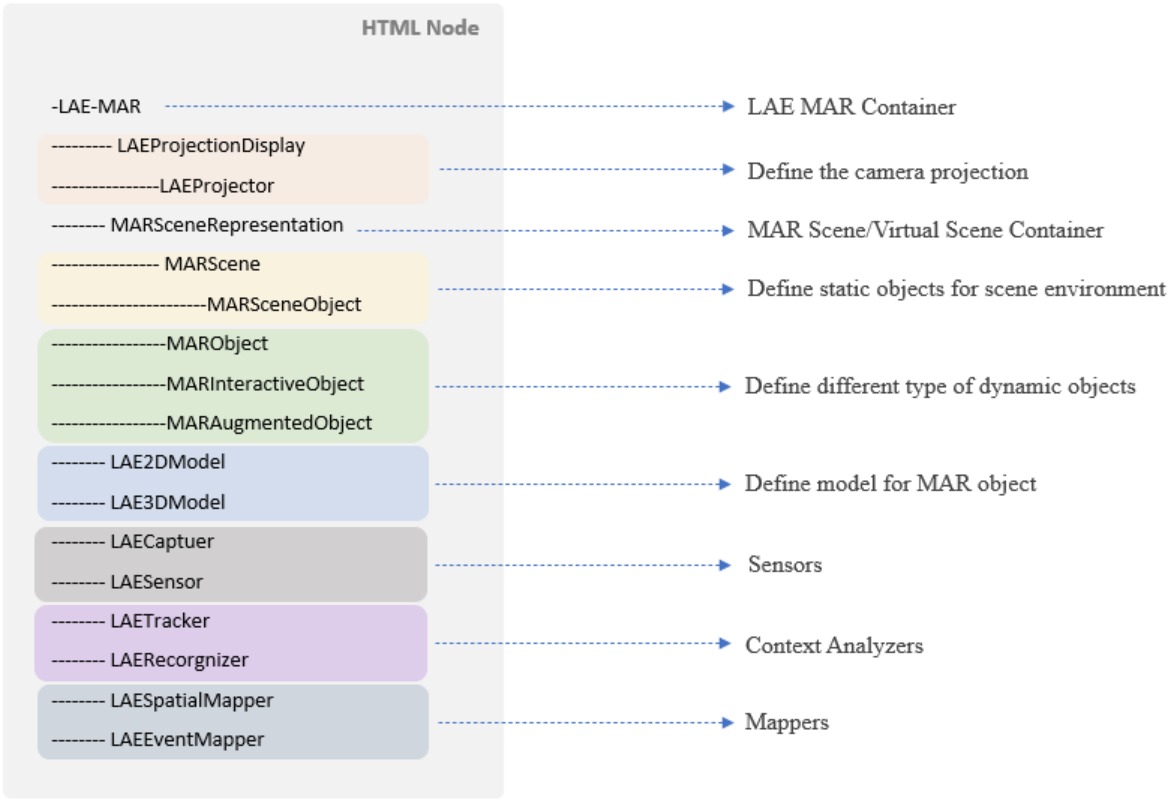



*Figure 9: Scene graph designed in LAE-MAR system*

## 5. LAE sensor

### 5.1. Overview

Sensor is a hardware (and optionally) software component able to measure specific physical properties. In this context, a sensor is used to detect, recognize and track the target physical object or information to be augmented. In this case, it is called a “pure sensor”. In this study, the LAESensor connects directly to the sensors or devices. Thus, the LAE sensor node stores the pre-processed sensor information collection according to the sensor type. Basically, this LAE sensor module can be designed to receive data of various sensors such as latitude, longitude, camera orientation, touch, acceleration, speed, movement, depth, or distance. For example, the system aims to obtain sensing/device information of the Head Mounted Display (HMD) context and HMD controller devices. The HMD display context data of a VR device can be used for rendering a stereo screen in the VR screen for an immersive experience. Then, the HMD controller sensors of button, accelerometer, and gyroscope are used to represent the user's hand tracking in 3D scene. The user's virtual arm is driven by the exchange or awareness of the HMD controller, which is an important part of supporting user interaction. Basically, the context Analyzer modules can generally access the output of the LAE sensor for use in LAE identifiers.

### 5.2. Node definition

**Table 1 – A prototypical object specification for Sensor**

| **LAESensor** | | | |
|---|---|---|---|
| **Name** | **Type/Valid range** | **Access Modifiers** | **Default** |
| LAESensor() | LAESensor* | Public | Function |
| id | String | Public | Specified by user's definition |
| enable | Bool | Public | True |
| rawData | Any | Private | Null |
| type | String | Private | Null |
| setRawData() | Void | Public | Function |
| getData() | Any* | Public | Function |

— **Property Description**

- LAESensor() : Constructor
- id : Identifier
- enable : Enable the sensor
- rawData: Sensed data
- type : Indicating the particular sensor type
- setRawData() : Set information for the sensor class
- getData() : Get accessible variable/data from sensor class

## 6. LAE capturer

### 6.1. Overview

Capturer is a component extended from Sensor designed specially to capture real-world objects/environment. The capturer is made up to handle over connected camera which its properties can be configurable according to the required information.

### 6.2. Node definition

**Table 2 − A prototypical object specification for Capturer**

| **LAECapturer** | | | |
|---|---|---|---|
| **Name** | **Type/Valid range** | **Access Modifiers** | **Default** |
| LAECapturer () | LAECapturer* | Public | Function |
| id | String | Public | Specified by user's definition |
| data | Image | Private | Null |
| cameraId | Number | Private | 0 |
| enabled | Bool | Private | True |

| | | | |
|---|---|---|---|
| type | String | Private | Capturer |
| resolutionWidth | Number | Private | 512 |
| resolutionHeight | Number | Private | 512 |
| mode | String | Private | RGB |
| setImageData() | Void | Public | Function |
| getImageData() | Image* | Public | Function |

— **Property Description**

- LAECapturer() : Constructor
- id : Identifier
- data : Captured data
- cameraId : Specify the capturer device
- enabled : Boolean value indicating whether to enable this Capturer
- type : Indicating the particular sensor type. Since this node is specifically designed for capturer, it is automatically defined as a type of capturer sensor
- resolutionWidth : Adjusting width resolution
- resolutionHeight : Adjusting height resolution
- mode : Specify image mode, if available
- setImageData() : Set information for the capturer class
- getImageData() : Get accessible variable/data from capturer class

# 7. Context analyzer

## 7.1. Overview

Providing system to understand real-world information, the context analyzer is needed to be defined in two forms, LAE tracker and LAE recognizer. Both of them are taking care of analysing and tracking the real data and translate it to understandable data for supporting behaviours in in virtual world.

## 7.2. Components

### 7.2.1. LAE Tracker

#### 7.2.1.1. Overview

Tracker is a subclass defined as a component to track the target information based on settings or leaning. In this part, tracker is every useful to apply any kind of smart learning techniques. It can be used to track the target information getting from one or many sensors and used it to serve the purpose in virtual scene. In the LAE capture module, there are possibilities of different types of cameras depending on the data requirements. For example, if we need to implement the LAE-MAR system in the 360º environment, this module must obtain information from the 360º camera. Since this module handles camera information, any method can be applied to obtain camera data that is useful for the use of other modules. However, we write programs to read video and audio information from common cameras in this context. Basically, we recommend using the general camera. The reason for using a general camera is because it is convenient for any user whose computer has a webcam installed. LAE can be extracted using machine learning techniques instead of the built-in algorithms of sophisticated cameras. In addition, the resolution, image resolution, or camera ID properties for requesting video / audio information can be assigned in the objects of the LAE capturer node. As a result of LAE capturer we can get any type of data according to the camera information such as video frame, depth information, skeleton, sound, etc. All information is accessible for other modules such as LAE sensor, LAE tracker or LAE recognizer.

### 7.2.1.2. Node definition

**Table 3 − A prototypical object specification for LAE Tracker**

| **LAETracker** | | | |
|---|---|---|---|
| **Name** | **Type/Valid range** | **Access Modifiers** | **Default** |
| LAETracker() | LAETracker* | Public | Function |
| id | String | Public | Null |
| data | Any | Private | Null |
| type | String | Private | Null |
| enabled | Bool | Private | True |
| deeplabModelSrc | Object | Private | Null |
| originalImg | Image | Private | Null |
| maskImg | Image | Private | Null |
| chromakeyingImg | Image | Private | Null |
| hmrModelSrc | Object | Private | Null |
| joints | Array<vector2> | Private | [ ] |
| verts | Array<vector3> | Private | [ ] |
| joints3D | Array<vector3> | Private | [ ] |
| normals | Array<vector3> | Private | [ ] |
| processImageSegmentation() | Object | Private | Function |
| process3DReconstruction() | Image | Private | Function |
| set() | Void | Public | Function |
| get() | Any* | Public | Function |

— **Property Description**

- LAETracker () : Constructor
- id : Identity
- data : Tracked data (depends on the sensor/input)
- type : Track a specific target information from sensor/input data
- enabled : Boolean value indicating whether to enable this Tracker
- deeplabModelSrc : Path of pre-train image segmentation model to be used
- originalImg : A variable for the original image from the camera

- maskImg : Mask image generated from image segmentation model
- chromakeyingImg : Final output after filtering the live actor body
- hmrModelSrc : Path of pre-train 3D reconstruction model to be used
- joints : A variable to store 2D joints of a body from an image
- verts : The vertices information for a predicted body model
- joints3D : 3D joints of a body for skeleton behaviors
- normals : Compute the normal for a 3D model to reflex with light
- processImageSegmentation() : A function to run the pre-trained image segmentation model and do perdition
- process3DReconstruction() : A function to run the pre-trained HMR model and do perdition
- set() : Set information for the tracker class
- get() : Get accessible variable/data from tracker class

### 7.2.2. LAE Recognizer

#### 7.2.2.1. Overview

Recognizer is working similarly to tracker, but it has potential to analyse the information from sensors. Also, the Recognizer can be used to apply the machine learning/deep learning where is a tool to recognize things as needed. The recognizer and sensor are related process. By obtaining the sensing data, the recognizer takes responsibility to understand and translate that data that can be readily prepared for mapper functions. Particularly, the recognizer tries to understand the data of sensors. For instance, the system uses the LAE recognizer to manage the user interaction. By using this LAE recognition device to manage user interaction, the system uses this module to recognize/read HMD controller data information received from the LAE sensor. Since the LAE recognition module is set to recognize target data, there must be filters to observe different types of data. Targets for this module can be device/sensor API information or event handlers. For the event handler target data, this is useful for the interactive system, which is a basic module is implemented to enable LAE interaction. In addition, the LAE recognition module takes care of the display of the HMD. After reviewing the display pop-up information of the HMD, the system can render the stereo view directly to the VR device based on the information provided. In the web application, the system can log in to access information from VR or HMD devices.

#### 7.2.2.2. Node definition

**Table 4 − A prototypical object specification for LAE Recognizer**

| **LAERecognizer** | | | |
|---|---|---|---|
| **Name** | **Type/Valid range** | **Access Modifiers** | **Default** |
| LAERecognizer() | LAERecognizer* | Public | Function |
| target | Any | Private | Null |
| data | Any | Private | Null |
| type | String | Private | Null |
| enabled | Bool | Public | True |
| filter | Void | Public | Function |
| targetHandler | Void | Public | Function |
| setRawData() | Void | Public | Function |
| getData() | Any* | | Function |

— **Property Description**

- LAERecognizer() : Constructor
- target : Target object template information to be recognized
- data : Specified by sensed data applied to be recognized (depends on the Recognizer type)
- type : Recognize a specific target information from sensor/input data
- enabled : Boolean value indicating whether to enable this Recognizer
- filter : Function filtering or post-processing the sensed and recognized data
- targetHandler : Telling that the target is activated
- setRawData : Set information for the recognizer class
- getData : Get accessible variable/data from recognizer class

# 8. LAE model

## 8.1. Overview

This LAE Model is a parent class for constructing LAE model in the MAR scene. This parent class is used to create an LAE model which is considered as a virtual object that its physical assets consist of many different information of sensors.

## 8.2. Node definition

**Table 5 − A prototypical object specification for LAEModel::LAE-MARNode**

| **LAEModel** | | | |
|---|---|---|---|
| **Name** | **Type/Valid range** | **Access Modifiers** | **Default** |
| LAEModel () | LAEModel | Public | Function |
| id | String | Public | Null |
| originalImg | Image | Private | Null |
| audio | Audio | Private | Null |
| type | String | Public | Null |
| rotation | Vector3 | Public | (0, 0, 0) |
| position | Vector3 | Public | (0, 0, 0) |
| scale | Vector3 | Public | (1, 1, 1) |
| getImgData() | Image* | Public | Function |
| getAudioData() | Audio* | Public | Function |
| setData() | Void | Public | Function |

— **Property Description**

- LAEModel() : Constructor
- id: Identifier
- originalImg: A variable for the original image from the camera
- audio: A variable for the original audio from the camera if it exists
- type: Specify the model type

- rotation: A matrix for rotation in the scene
- position: A matrix for a position in the scene
- scale: A matrix for scale in the scene
- getImgData(): A function to access the segmented image
- getAudioData(): A function to access the audio
- setData(): Set the information from a camera

## 8.3. 2D LAE model

### 8.3.1. Overview

On the physical side, the abstract LAE2DModel node is made up of the required sensors, the camera, the tracking technique, and the recognition technique to generate as a transmission texture on a plane, mapping with a virtual object designed to represent as an actor in 3D space. Initially, the actual LAE can be obtained from the image captured by many different camera options. Despite using the advanced camera, we only used a general camera to capture the actual LAE. Together with the original image, we implemented the image segmentation technique to predict the target object, the human body. We mapped the original image to the intended result to extract the LAE and remove unnecessary background. Next, use the extracted LAE image as a bounding box texture, whose depth value must be zero. Thus, we receive a real-time texture in a box that can be added to the 3D scene with spatial information. In addition, the system not only takes care of the image but also processes the audio from the camera, if it exists.

### 8.3.2. Node definition

**Table 6 − A prototypical object specification for 2DLAEModel::LAEModel::LAE-MARNode**

| **2DLAEModel** | | | |
|---|---|---|---|
| **Name** | **Type/Valid range** | **Access Modifiers** | **Default** |
| 2DLAEModel () | 2DLAEModel | Public | Function |
| id | String | Public | Null |
| originalImg | Image | Private | Null |
| maskImg | Image | Private | Null |
| chromakeyingImg | Image | Private | Null |
| type | String | Public | Null |
| rotation | Vector3 | Public | (0, 0, 0) |
| position | Vector3 | Public | (0, 0, 0) |
| scale | Vector3 | Public | (1, 1, 1) |
| processImageSegmentation() | Void | Private | Function |
| processAudio() | Void | Private | Function |
| bodyFilter() | Void | Private | Function |
| mappingTexture() | Void | Private | Function |
| getImgData() | Image* | Public | Function |
| getAudioData() | Audio* | Public | Function |
| setData() | Void | Public | Function |

| | | | |
|---|---|---|---|
| additionalproperties | Array[any] | Public | Null |
| setAdditionalProperties(any[]) | Void | Public | Function |

— **Property Description**

- 2DLAEModel() : Constructor
- id : Identifier
- originalImg : A variable for the original image from the camera
- maskImg : Mask image generated from Deeplab model
- chromakeyingImg : Final output after filtering the live actor body
- type : Specify the model type
- rotation : A matrix for rotation in the scene
- position : A matrix for a position in the scene
- scale : A matrix for scale in the scene
- processImageSegmentation() : Function to execute the model for image segmentation
- processAudio() : Process the audio if exists
- bodyFilter() : Filter the body from the original image with the segmented mask
- mappingTexture() : A function to map a virtual object with texture
- getImgData() : Function to access the segmented image
- getAudioData() : Function to access the audio
- setData() : Set the sequences of image/audio as the input
- additionalproperties : In case, LAE model requires more properties so it can be defined as needed
- setAdditionalProperties(any[]) : Return the list/array of required properties

## 8.4. 3D LAE model

### 8.4.1. Overview

This 3D LAE model is used to build a model in the form of 3d object. Because it is extended from LAE Model, this node has the same properties as its parent class. On the physical side, the LAE3DModel abstract node composes the sensors, camera, tracking technique, and recognition technique needed to generate as a constructed 3D model, mapping with a virtual object to represent a live actor in 3D space. First of all, we have the LAE image captured by a general camera to build a virtual 3D LAE. Using the original image data, the system uses a 3D reconstruction technique to reconstruct a complete 3D mesh of the LAE. By using this technique, we can get some information like 3D joints, vertices, and joints after the prediction phase. Vertices and 3D joints are used to build a 3D mesh model for a 3D scene. After we have a 3D mesh built, we can add it to the scene and to spatial information such as position, rotation, and scale. For audio information, it is the same as 2D LAE. The system not only takes care of the image but also processes the audio from the camera, if it exists.

### 8.4.2. Node definition

**Table 7 − A prototypical object specification for 3DLAEModel::LAEModel::LAE-MARNode**

| **3DLAEModel** | | | |
|---|---|---|---|
| **Name** | **Type/Valid range** | **Access Modifiers** | **Default** |
| 3DLAEModel() | 3DLAEModel | Public | Function |
| id | String | Public | Null |

| | | | |
|---|---|---|---|
| type | String | Public | Null |
| joints | Array<vector2> | Private | [ ] |
| verts | Array<vector3> | Private | [ ] |
| joints3D | Array<vector3> | Private | [ ] |
| normals | Array<vector3> | Private | [ ] |
| faces | Array<vector3> | Private | [ ] |
| rotation | Vector3 | Public | (0, 0, 0) |
| position | Vector3 | Public | (0, 0, 0) |
| scale | Vector3 | Public | (1, 1, 1) |
| Process3DReconstruction() | Void | Private | Function |
| processAudio() | Void | Private | Function |
| threejsModelConverter() | 3DModel* | Private | Function |
| setData() | Void | Public | Function |
| getModelData() | 3DModel* | Public | Function |
| getAudioData() | Audio* | Public | Function |
| additionalproperties | Array[any] | Public | [ ] |
| setAdditionalProperties (any[]) | Void | Public | Function |

— **Property Description**

- 3DLAEModel () : Constructor
- id : Identifier
- type : Default type is 3DLAE (3d-live-actor)
- joints : A variable to store 2D joints of a body from an image
- verts : The vertices information for a predicted body model
- joints3D : 3D joints of a body for skeleton behaviors
- normals : Compute the normal for a 3D model to reflex with light
- faces : The faces of vertices
- rotation : A matrix for rotation in the scene
- position : A matrix for a position in the scene
- scale : A matrix for scale in the scene
- process3DReconstruction() : Function to execute the model for constructing a 3D model from 2D image
- processAudio() : Access the audio if exists
- threejsModelConverter() : Translate the joints3d and vertices to a 3D model
- setData() : Set the sequences of 3d model data/audio as the input
- getModelData() : A function to access the constructed model
- getAudioData() : A function to access the audio
- additionalproperties : In case, LAE model requires more properties so it can be defined as needed
- setAdditionalProperties(any[]) : Return the list/array of required properties

## 9. LAE spatial mapper

### 9.1. Overview

In order to bring real object into virtual scene, there must be a functional node to map the objects between the real world to virtual world. Through this technique, LAE Spatial Mapper is defined as an interface to map the two objects, real and virtual object, working relatively. Moreover, this node mainly focuses on spacing context where position, rotation or scale can be defined for mapping real objects into virtual scene. In the LAE spatial mapper, there are two main functions to consider. Firstly, the LAE spatial mapper can be a connection between real LAE and virtual LAE. The real LAE is an abstract model that stores the various LAE information, just as the virtual LAE is a 3D virtual object modelled to support as a representation of LAE in the MAR scene. The connection between physical and virtual objects can be made by the ID assigned in the spatial mapper node. Secondly, the LAE spatial mapper can provide spatial information such as rotation, position, and scale to the virtual LAE in the MAR scene. Through this, the LAE object can be located anywhere in the scene.

### 9.2. Node definition

**Table 8 − A prototypical object specification for LAESpatialMapper::LAE-MARNode**

| **LAESpatialMapper** | | | |
|---|---|---|---|
| **Name** | **Description** | **Type/Valid range** | **Access Modifiers** |
| LAESpatialMapper() | LAESpatialMapper | Public | Function |
| id | String | Public | Null |
| laeModel | String | Public | Null |
| marObject | String | Public | Null |
| position | Vector3 | Public | (0, 0, 0) |
| rotation | Vector3 | Public | (0, 0, 0) |
| scale | Vector3 | Public | (1, 1, 1) |
| mappingSpatialInfo() | Void | Private | Function |
| getData() | Any* | Public | Function |
| setData() | Void | Public | Function |

— **Property Description**

- LAESpatialMapper () : Constructor
- id : Identifier
- laeModel : Specify the LAE model which used to map to MAR object
- marObject : Specify the MAR object which used to map to LAE model
- position : Tell where the virtual object should be located at
- rotation : Tell how the virtual object should be rotated
- scale : Mapping the spatial information with a virtual object
- mappingSpatialInfo() : Mapping the spatial information with a virtual object
- getData() : Return the list/array of required properties

- setData() : Set the properties to the Spatial mapper

## 10. LAE event mapper

### 10.1. Overview

LAE Event Mapper works in the same level of LAE Spatial Mapper. The LAE Event Mapper is responsible for events. The events are all defined in database which describes the action of objects. By receiving the information from sensors, the information can be analysed or defined to perform an action of objects according to database defined. On the other hand, the LAE event mapper node is created primarily to allow user interaction on the system. The input of this module is subsequently obtained from the LAE sensor and the LAE recognizer. Therefore, the LAE can perform some activities to carry out the events, which the interactive system can recognize. To do that, each event must be defined in a database, which describes the action of each object. Therefore, the LAE event mapper node is only used to attach the callback() function to the virtual interactive object.

### 10.2. Node definition

**Table 9 − A prototypical object specification for LAEEventMapper::LAE-MARNode**

| **LAEEventMapper** | | | |
|---|---|---|---|
| **Name** | **Type/Valid range** | **Access Modifiers** | **Default** |
| LAEEventMapper() | LAEEventMapper | public | Function |
| id | String | Public | Null |
| type | String | Public | Null |
| data | Any | Private | Null |
| targetVirtualObject | IneractiveVirtualObject | Public | Null |
| getIntersection() | Void | Private | Function |
| eventHandler() | Callback | Public | Function |
| initialHandler() | Callback | Public | Function |
| getData() | Any* | Public | Function |
| setData() | Void | Public | Function |

— **Property Description**

- LAESpatialMapper() : Constructor
- id : Identifier
- type : Specify the event type
- data : Sensing data for being used to handle the event
- targetVirtualObject : An object that listens to the event trigger
- getIntersection() : A ray-caster function for recognizing the intersection
- eventHandler() : Handle the event depending on the action and target object; return the target
- initialHandler() : Handle the event at the initial stage; return the target
- getData() : A function to access the data

- setData() : Set the data for recognizer module

## 11. MAR scene

### 11.1. Overview

MAR Scene is one of the MAR Scene Representation children. In this node, the class has been designed for constructing the fixed virtual environment. The MAR scene is a parent of the scene object. The transform matrix of scene object is under control of MAR scene. Since this node is designed for material designs of a virtual environment, the transform matrix cannot be modified in runtime.

### 11.2. Node definition

**Table 10 − A prototypical object specification for MARScene::TransformGroup::LAE-MARNode**

| **MARScene** | | | |
|---|---|---|---|
| **Name** | **Type/Valid range** | **Access Modifiers** | **Default** |
| MARScene () | MARScene | Public | Function |
| autoUpdate | Boolean | Private | False |
| children | [VirtualObject] | Private | [ ] |
| addChild() | Void | Public | Function |
| removeChild() | Void | Public | Function |
| removeAllChild() | Void | Public | Function |
| getChildren() | [VirtualObject] | Public | Function |
| getData() | Any* | Public | Function |

— **Property Description**

- MARScene() : Constructor
- autoUpdate : The renderer checks every frame if the scene and its objects need matrix updates
- children : Store the children node, which also represents virtual scene objects
- addChild() : Add a child to this scene node
- removeChild() : Remove a child from this scene node
- removeAllChild() : Remove all children node
- getChildren() : Obtain all containing node, children
- getData() : A function to access the scene representation data

## 12. MAR scene object

### 12.1. Overview

MarSceneObject is used to define a fix object which the object cannot be transformed, rotated or scaled during the running time since it is used to decorate the scene environment. In another hand, the object properties cannot be changed or updated when system running. Basically, this model just stores the 3D model file and embeds it into the scene with spatial information like position, scale, and rotation. In particular, the transformation matrix is deactivated after the system starts. Since the model has a static shape, it works mainly for material design. For example, the model uses the 3D model file to design the manufacturing environment, building, fixed table, and so on.

## 12.2. Node definition

**Table 11 – A prototypical object specification for MARSceneObject::TransformGroup::LAE-MARNode**

| **MARSceneObject** | | | |
|---|---|---|---|
| **Name** | **Type/Valid range** | **Access Modifiers** | **Default** |
| MARSceneObject() | MARSceneObject | Public | Function |
| id | String | Public | Null |
| hidden | Boolean | Public | False |
| transparent | Number | Public | 1 |
| type | String | Public | Null |
| 3DObject | 3Dobject | Public | Null |
| position | Vector3 | Private | (0, 0, 0) |
| scale | Vector3 | Private | (1, 1, 1) |
| rotation | Vector3 | Private | (0, 0, 0) |
| matrixAutoUpdate | Boolean | Private | False |
| receiveShadow | Boolean | Private | True |
| castShadow | Boolean | Private | True |
| userData | Object | Public | Null |
| initialSpatialInfo() | void | Public | Function |
| getData() | Any | Public | Function |

— **Property Description**

- MARSceneObject() : Constructor
- id : Identifier
- hidden : Hide the object in a virtual scene
- transparent : The opacity of the object model
- type : Types of a model (Gltf, Cube, Cone, etc.)
- 3DObject : A virtual object used for a virtual scene
- position : Object’s local position. Default (0, 0, 0)
- scale : Object's local scale
- rotation : Object's local rotation
- matrixAutoUpdate : Default is false. recalculate the matrix world property
- receiveShadow : Default is false. It allows the model can receive the shadow
- castShadow : This allows having cast shadow
- userData : Store the additional information for some particular data
- initialSpatialInfo() : Define the initial information of the spatial mapper
- getData() : A function to access object data

## 13. MAR dynamic object

### 13.1. Overview

In the LAE system is such an object used to behave differently based on model designed. The object can be performed in many different contexts depending on pre-designed models. The system allows the mar object model to perform various behaviours. Therefore, this model can represent the physical object dynamically based on the information provided. Information from manufacturing machines or sensors can be used to change/animate the imported 3D object model with attached animation clips. The properties of the service node are designed to play the animation with a 3D model with an animation clip. For example, this is very useful for representing the real-world object or sensor state by jumping to an exact time in the animation. In addition, the system can update the spatial information at any time, so the update of the model transformation matrix is always set to true.

### 13.2. Node definition

**Table 12 – A prototypical object specification for MARDynamicObject::MARObject::VirtualTG::TransformGroup::LAE-MARNode**

| **MARDynamicObject** | | | |
|---|---|---|---|
| **Name** | **Type/Valid range** | **Access Modifiers** | **Default** |
| MARDynamicObject() | MARScene | Public | Function |
| id | String | Public | Null |
| hidden | Boolean | Public | False |
| transparent | Number | Public | 1 |
| type | String | Public | Null |
| 3Dobject | 3Dobject | Public | Null |
| position | Vector3 | Private | (0, 0, 0) |
| scale | Vector3 | Private | (1, 1, 1) |
| rotation | Vector3 | Private | (0, 0, 0) |
| deltaTime | Number | Public | 0.5 |
| timeAt | Number | Public | 0 |
| loop | Boolean | Public | False |
| matrixAutoUpdate | Boolean | Private | True |
| receiveShadow | Boolean | Private | True |
| castShadow | Boolean | Private | True |
| initialSpatialInfo() | void | Public | Function |
| updateSpatialInfo() | void | Public | Function |
| userData | Object | Public | Function |
| getData() | Any | Public | Function |

— **Property Description**

- MARObject() : Constructor

- id: Identifier
- hidden: Hide the object in a virtual scene
- transparent: The opacity of the object model
- type: Types of a 3D model (Gltf, Cube, Cone, etc.)
- 3Dobject: A virtual object used for a virtual scene
- position: Object's local position. Default (0, 0, 0)
- scale: Object's local scale
- rotation: Object's local rotation
- deltaTime : The time scale of action
- timeAt: The local time of the action (Jump to an exact time in an animation)
- loop: An infinite number of repetitions
- matrixAutoUpdate : Default is True. recalculate the matrix-world property
- receiveShadow : Default is false. It allows the model can receive the shadow
- castShadow : This allows having cast shadow
- initialSpatialInfo() : Define the initial information of the spatial mapper
- updateSpatialInfo() : A function to update spatial info in runtime
- userData : Store the additional information for some particular data
- getData() : A function to access the dynamic object data

## 14. MAR interactive object

### 14.1. Overview

The MAR Interactive Object is subclass of MAR object that used to make an object interactable. It means object properties can be changed or updated dynamically depending on the defined actions. In order for the user to play with virtual objects, the interactive model is designed as a model to receive the interaction and manage itself to handle the event. For example, the manual tracking virtual object can represent a game controller device in the 3D scene. Tracking hand motions and motion are dependent on device sensing information such as accelerometer, gyroscope, and button information. In fact, the spatial information of this interactive virtual object is under the control of the interactive system, which means that user interaction can update the transformation matrix of an interactive virtual object.

### 14.2. Node definition

**Table 13 – A prototypical object specification for MARInteractiveObject::MARObject::VirtualTG::TransformGroup::LAE-MARNode**

| **MARInteractiveObject** | | | |
|---|---|---|---|
| **Name** | **Type/Valid range** | **Access Modifiers** | **Default** |
| MARInteractiveObject () | MARInteractiveObject | Public | Function |
| id | String | Public | Null |
| hidden | Boolean | Public | False |
| transparent | Number | Public | 1 |
| type | String | Public | Null |
| 3Dobject | 3Dobject | Public | Null |
| fixed | Boolean | Public | False |

| | | | |
|---|---|---|---|
| position | Vector3 | Private | (0, 0, 0) |
| scale | Vector3 | Private | (1, 1, 1) |
| rotation | Vector3 | Private | (0, 0, 0) |
| matrixAutoUpdate | Boolean | Private | True |
| receiveShadow | Boolean | Private | True |
| castShadow | Boolean | Private | True |
| initialSpatialInfo() | Void | Public | Function |
| updateSpatialInfo() | Void | Public | Function |
| getIntersection() | Object | Public | Function |
| userData | Object | Public | Null |
| getData() | Any | Public | Function |

— **Property Description**

- MARInteractiveObject() : Constructor
- id : Identifier
- hidden : Hide the object in a virtual scene
- transparent : The opacity of the object model
- type : Types of a 3D model (Gltf, Cube, Cone, etc.)
- 3Dobject : A virtual object used for a virtual scene
- fixed : Whether to update spatial information
- position : Object's local position. Default (0, 0, 0)
- scale : Object's local scale
- rotation : Object's local rotation
- matrixAutoUpdate : Default is True. recalculate the matrix world property
- receiveShadow : Default is false. It allows the model can receive the shadow
- castShadow : This allows having cast shadow
- initialSpatialInfo() : Define the initial information of the spatial mapper
- updateSpatialInfo() : A function to update spatial info in runtime
- getIntersection() : Whether it is intersected with the user's controller devices
- userData : Store the additional information for some particular data
- getData() : A function to access the interactive object data

## 15. MAR augmented object

### 15.1. Overview

Another subclass of MAR object is MAR Augmented Object which enables to make object special in order to perform such as an informative augmentation. The new augmented object is defined as a transform group to control the children's object. Additionally, this model is beneficial for data display, graph display, and setup controls. Because the augmented model is designed to contain the child object, the augmented object is formed as a parent object to control the other virtual models. For example, the augmented object has two different types of child elements, the text model is used to

display the data, and the button models are used to listen to user interaction. Basically, the augmented object applied drag control on an interactive panel object to move the panel and children around the environment. Since this augmented model requires different types of models as children, we can create more external virtual objects as panel properties to support the requirement. The node object in this model has the properties to access, add, and remove children on demand. Surprisingly, the spatial information of all child nodes must be managed by a transformation group of a parent, a model of augmentation.

### 15.2. Node definition

**Table 14 − A prototypical object specification for MARAugmentedObject::MARObject::VirtualTG::TransformGroup::LAE-MARNode**

| **MARAugmentedObject** | | | |
|---|---|---|---|
| **Name** | **Type/Valid range** | **Access Modifiers** | **Default** |
| MARAugmentedObject() | MARAugmentedObject | Public | Function |
| id | String | Public | Null |
| hidden | Boolean | Public | False |
| transparent | Number | Public | 1 |
| type | String | Public | Null |
| 3Dobject | 3Dobject | Public | Null |
| children | [VirtualObject] | Private | [ ] |
| position | Vector3 | Private | (0, 0, 0) |
| scale | Vector3 | Private | (1, 1, 1) |
| rotation | Vector3 | Private | (0, 0, 0) |
| matrixAutoUpdate | Boolean | Private | False |
| receiveShadow | Boolean | Private | True |
| castShadow | Boolean | Private | True |
| initialSpatialInfo() | Void | Public | Function |
| updateSpatialInfo() | Void | Public | Function |
| addChild() | Void | Public | Function |
| removeChild() | Void | Public | Function |
| removeAllChild() | Void | Public | Function |
| getChildren() | [VirtualObject]* | Public | Function |
| userData | Any | Public | Null |
| getData() | Any* | Public | Function |

— **Property Description**

- MARAugmentedObject () : Constructor
- id : Identifier
- hidden : Hide the object in a virtual scene

- transparent : The opacity of the object model
- type : Types of a 3D model (Gltf, Cube, Cone, etc.)
- 3Dobject : A virtual object used for a virtual scene
- children : Store the children node containing in this augmented group
- position : Object's local position
- scale : Object's local scale
- rotation : Object's local rotation
- matrixAutoUpdate : Default is True. It recalculates the matrix world property
- receiveShadow : Default is True. It allows the model can receive the shadow
- castShadow : It allows having cast shadow
- initialSpatialInfo() : Define the initial information of the spatial mapper
- updateSpatialInfo() : A function to update spatial info in runtime
- addChild() : Add a child to this augmented object group
- removeChild() : Remove a child from the augmented object group
- removeAllChild() : Remove all children from the augmented object group
- getChildren() : Obtain all containing object, children
- userData : Store the additional information for some particular data
- getData() : A function to access the object data

## 16. Scene representation

### 16.1. Overview

MAR Scene Representation plays in an important part to construct the entire scene virtually. It performs as a parent class which requires child classes to create or design the scene environment in virtual space. Also, the LAE scene representation node plays an important role in the virtual construction of the entire scene. Furthermore, it is designed to cover the LAE node and simulate them in virtual objects under the control of the LAE event mapper and spatial mapper. In the other word, the representation of the scene LAE that represents as placeholders for the virtual scene. It acts as a practical structure that combines physical object nodes and physical object nodes.

### 16.2. Node definition

**Table 15 − A prototypical object specification for MARSceneRepresentation::LAE-MARNode**

| MARSceneRepresentation | | | | |
|---|---|---|---|---|
| **Name** | **Type/Valid range** | **Access Modifiers** | **Default** | **Description** |
| MARSceneRepresentation() | MARSceneRepresentation | public | Function | Constructor |
| id | String | Public | Null | Identifier |
| autoUpdate | Boolean | Private | True | The renderer checks every frame if the scene and its objects need matrix updates |
| background | Any | Public | Null | It can be set to a color or texture |
| children | [VirtualObject] | Private | [ ] | Store the children node, which also represents the physical and virtual objects |
| addChild() | Void | Public | Function | Add a child to this scene node |
| removeChild() | Void | Public | Function | Remove a child from this scene node |

| | | | | |
|---|---|---|---|---|
| removeAllChild() | Void | Public | Function | Remove all children node |
| getChildren() | [VirtualObject]* | Public | Function | Obtain all containing node, children |
| getData() | Any* | Public | Function | Access function for the scene representation data |

— **Property Description**

- MARSceneRepresentation(): Constructor
- id : Identifier
- autoUpdate: The renderer checks every frame if the scene and its objects need matrix updates
- background: The scene background can be set as a color or texture
- children: Store the children node, which also represents the physical and virtual objects
- addChild(): Add a child to this scene node
- removeChild(): Remove a child from this scene node
- removeAllChild(): Remove all children node
- getChildren(): Obtain all containing node, children
- getData(): Access function for the scene representation data

## 17. Scene description

### 17.1. Overview

Learning from sensing information and mapping it into the events in virtual scene, mapping the events and sensing data are defined in a database called event database. Taking control over the mapping event, database is designed to connect the sensors, event types and devices together in order to provide specific match of handling process of an event for the LAE event mapper.

### 17.2. Node definition

**Table 16 − A prototypical object specification for LAESceneDescription::LAE-MARNode**

| **LAESceneDescription** | | | |
|---|---|---|---|
| **Name** | **Type/Valid range** | **Access Modifiers** | **Default** |
| LAESceneDescription() | LAESceneDescription | Public | Function |
| targetVirtualObject | IneractiveVirtualObject | Private | Null |
| eventType | String | Public | Null |
| returnInfo | VirtualObject | Private | Null |
| callbackHandler() | Void | Public | Function |
| getData() | Any* | Public | Function |
| setData() | Void | Public | Function |

— **Property Description**

- LAESceneDescription(): Constructor
- targetVirtualObject: A virtual object in MAR scene that is used to handle the event
- eventType: Define the type of event. For example: click, hover, touch, etc.
- returnInfo: Information to be returned with the callback function

- callbackHandler(): Trigger the event of a virtual object
- getData(): A function to access the class data
- setData(): Set the data for the module

## 18. LAE projection display

### 18.1. Overview

The node, which takes a part in projecting scene in virtual scene, is LAE Projection Display. Making a set of projection in virtual scene, this node must be defined to configure the viewpoints to be captured. Since it is a parent node, it has children, which each child node can capture the scene in different position locations or angles.

### 18.2. Node definition

**Table 17 – A prototypical object specification for LAEProjectionDisplay::LAE-MARNode**

| **LAEProjectionDisplay** | | | |
|---|---|---|---|
| **Name** | **Type/Valid range** | **Access Modifiers** | **Default** |
| LAEProjectionDisplay() | LAEProjectionDisplay | Public | Function |
| children | [Projector] | Private | [ ] |
| addChild() | Void | Public | Function |
| removeChild() | Void | Public | Function |
| removeAllChild() | Void | Public | Function |
| getChildren() | [Projector] | Public | Function |
| getData() | Any* | Public | Function |

— **Property Description**

- LAEProjectionDisplay (): Constructor
- children: Store the children node, which also represent the projector object
- addChild(): Add a projector to the scene
- removeChild(): Remove a projector from the scene
- removeAllChild(): Remove all children node
- getChildren(): Obtain all containing node, children
- getData(): A function to access the projection display data

## 19. LAE projector

### 19.1. Overview

LAE Projector is a subclass of LAE Projection Display used to described where the camera should be put or looked at. There are properties applicable to project the scene where considered as an interest view. This LAE projection display node is used to describe the type of projector and information on where the camera should be positioned or viewed. In this system, more than one projector can be added to capture multiple views of the scene. Basically, there are two types of projection such as perspective and orthographic projector. The perspective camera model is the most common forecast mode used in the 3D scene. A mathematical model describes the correspondence between the observed points in the

world and the pixels of the captured image. Also, it describes the transition from 3D points in the world to 2D points in an image. The mode of this projection is designed to mimic the way the human eye sees. Also, orthographic projection is another common mode of scene projection in which the size of an object in the displayed image remains constant regardless of its distance from the camera. Unlike a perspective camera, orthographic projection describes a viewing box with parallel corners rather than a box where both sides meet at the horizontal point of the scene. In addition, there are applicable node properties for projecting the scenario for deployment, where a view of interest is considered.

### 19.2. Node definition

**Table 18 − A prototypical object specification for LAEProjector::LAE-MARNode**

| **LAEProjector** | | | |
|---|---|---|---|
| **Name** | **Type/Valid range** | **Access Modifiers** | **Default** |
| LAEProjector() | LAEProjector | Public | Function |
| id | String | Public | Null |
| type | String | Public | Null |
| control | String | Public | Null |
| left | Number | Public | 0 |
| bottom | Number | Public | 1 |
| width | Number | Public | 1 |
| height | Number | Public | 1 |
| position | Vector3 | Public | (0,0,0) |
| lookAtPostion | Vector3 | Public | (0,0,0) |
| fav | Number | Public | 75 |
| aspect | Number | Public | WinWidth/WinHeight |
| nearDistance | Number | Public | 0.1 |
| farDistance | Number | Public | 100000 |

— **Property Description**

- LAEProjector(): Constructor
- id: Identifier
- type: Type of viewpoint projection
- control: If the projector can be controlled directly by using mouse according to control type
- left: Decide the starting point of view from left side by scale rate of 0 and 1
- bottom: Decide the end point of view from right side by scale rate of 0 and 1
- width: Decide the width by scale rate of 0 and 1
- height: Decide the height by scale rate of 0 and 1
- position: Set the position where the camera should be placed at
- lookAtPostion: Return the position where the camera is focused on
- fav: Camera frustum vertical field of view
- aspect: Camera frustum aspect ratio

- nearDistance: Camera frustum near plane
- farDistance: Camera frustum far plane

## 20. ANNEX

### 20.1. Live actor

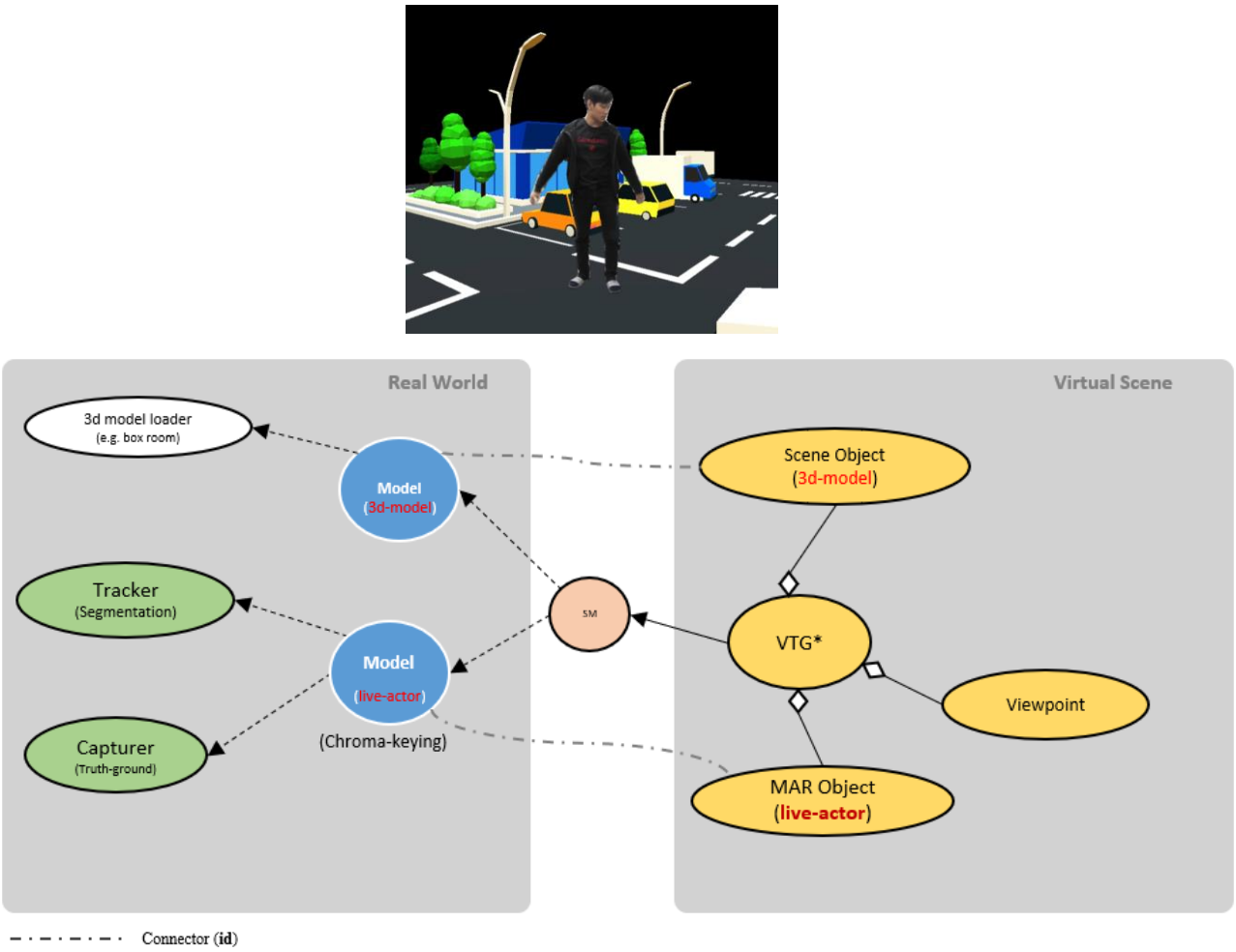



*Figure 10: Scenario example of live actor in LAE-MAR system*

— **Virtual Scene**
  - In the virtual scene, the viewpoint or projector has to be defined since it is useful to capture the story in the scene
  - Also, in the virtual scene, there are two object components required
    - Scene Object is used to load file of 3d model, which is the view of city according to figure above
    - Mar Object created for mapping with live-actor model, which made up in real-world side, through Spatial Mapper (SM)

— **Real world**
  - There two models constructed to map with two virtual objects in virtual scene
    - For “3d-model” model, through SM, it is connected with scene object in virtual scene. This model constructed by “3d-model” loader, which allows model to upload 3d model from file format
    - For “live-actor” model, through SM, it connected with mar object in virtual scene. This model constructed by tracker, which applied the deep learning technique about image segmentation, and capturer, which used to capture sequence of image from real world. Therefore, constructed model produces a chroma-keying live-actor

— Here is the actual implementation of the live actor (LAE).

```
<!-- MAR LAE node -->
<MAR-LAE id="lae" description="LAE Project" isShownFPS="true" isShownConfig="true">
  <MARSceneRepresentation id="mar-scene" type="3d-Scene">
    <!-- MAR object for LAE -->
    <MARObject id="object1" type="2d-live-actor"></MARObject>
  </MARSceneRepresentation>

  <!-- LAE Models -->
  <LAE2DModel id="laemodel1"
              laeCaputurer="laecapture1"
              laeRecognizer="laerecognizer1">
  </LAE2DModel>
  <LAECapturer id="laecapture1"
               cameraType="general-camera"
               cameraId="0"
               resolution="640x640"
               mode="rgb">
  </LAECapturer>
  <LAETracker id="laetracker1" type="chromakeying"> </LAETracker>
  <LAERecognizer id="laerecognizer1"
                 type="oculus-controller">
  </LAERecognizer>

  <!-- LAE Mapper -->
  <LAESpatialMapper id="laespatialmapper1"
                    model="laemodel1"
                    marObject="object1"
                    mappingType="texture"
                    position="1 1 1"
                    scale="1 1 0"
                    rotation="0.01 0.6 0.01">
  </LAESpatialMapper>
</MAR-LAE>
```

### 20.2. Classroom design

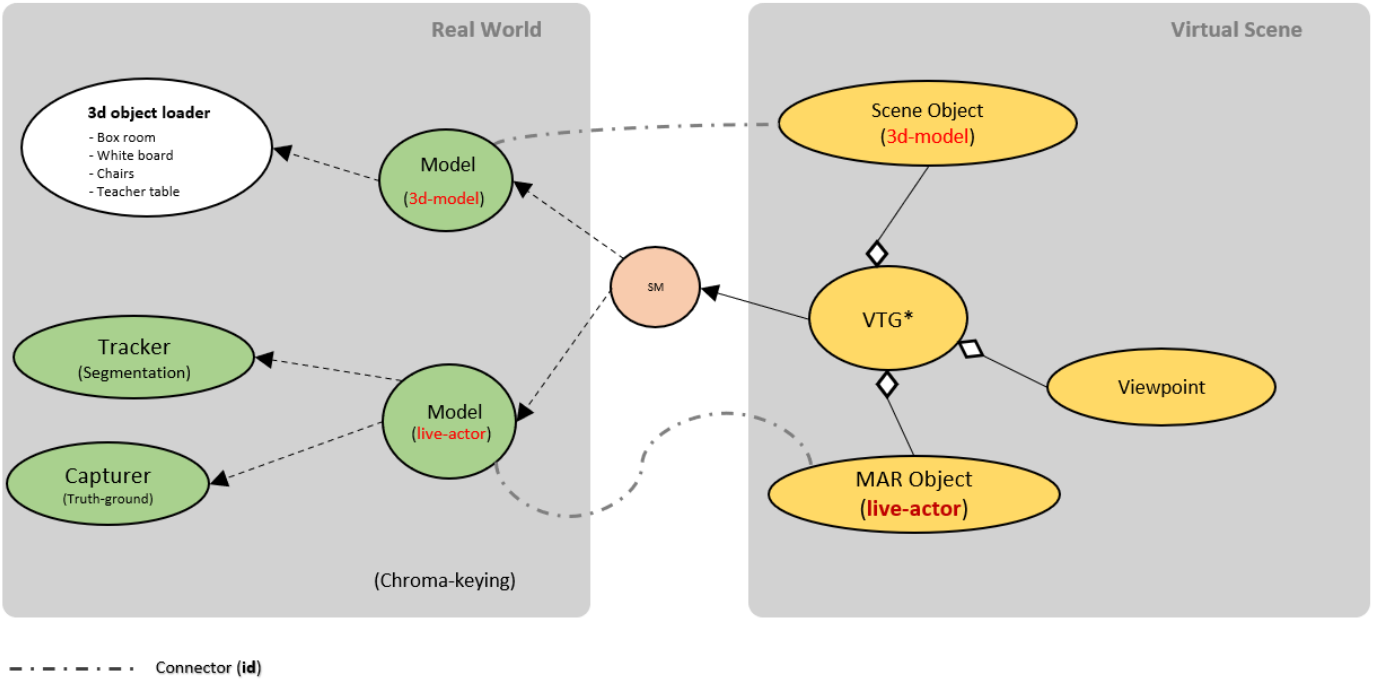



*Figure 11: Scenario example of designing classroom in LAE-MAR system*

— **Virtual Scene**
  - In the virtual scene, the viewpoint or projector has to be defined since it is useful to capture the story in the scene
  - Also, in the virtual scene, there are two types of virtual object have to be created
    - Scene objects are used to load file of 3d models, which are box room, table, chair and whiteboard.
    - Mar Object created for mapping with live-actor model, which made up in real-world side, through SM

— **Real world**
  - There two models constructed to map with two virtual objects in virtual scene
    - For "3d-model" model, through SM, it connected with scene object in virtual scene. This model constructed by "3d-model" loader, which allows to upload 3d models
    - For "live-actor" model, through SM, it connected with mar object in virtual scene. This model constructed by tracker, which applied the deep learning technique about image segmentation, and capturer, which used to capture sequence of image from real world. Therefore, constructed model produces a chroma-keying live-actor

— Here is the actual implementation of the classroom design.

```
<!-- MAR-LAE node -->
 <LAE-MAR id="lae" isShownFPS="true" isShownConfig="true">

  <!-- MAR SceneRepresentation -->
  <MARSceneRepresentation id="mar-scene">
      <MARScene id="3dscene">
          <MARSceneObject id="sceneobject1" type="3d-model"></MARSceneObject>
          <MARSceneObject id="sceneobject2" type="3d-model"></MARSceneObject>
    </MARScene>
  </MARSceneRepresentation>

  <!-- LAE Models -->
  <Model id="laemodel1"
              modelFormat="gltf"
              src="3Dmodels/white-board/white-board.glb">
  </Model>
  <Model id="laemodel2"
              modelFormat="gltf"
              src="3Dmodels/school-chair.glb">
  </Model>

  <!-- LAE Mappers -->
  <LAESpatialMapper id="laespatialmapper1"
                  model="laemodel1"
                  marObject="sceneobject1"
                  position="1 1 -3"
                  scale="0.03 0.03 0.03"
                  rotation="-1.6 0 0">
  </LAESpatialMapper>
  <LAESpatialMapper id="laespatialmapper2"
                    model="laemodel2"
                    marObject="sceneobject2"
                    position="-1 0.5 3"
                    scale="0.3 0.3 0.3"
                    rotation="0 3.1 0">
  </LAESpatialMapper>
</LAE-MAR>
```

## 20.3. Multi-projection

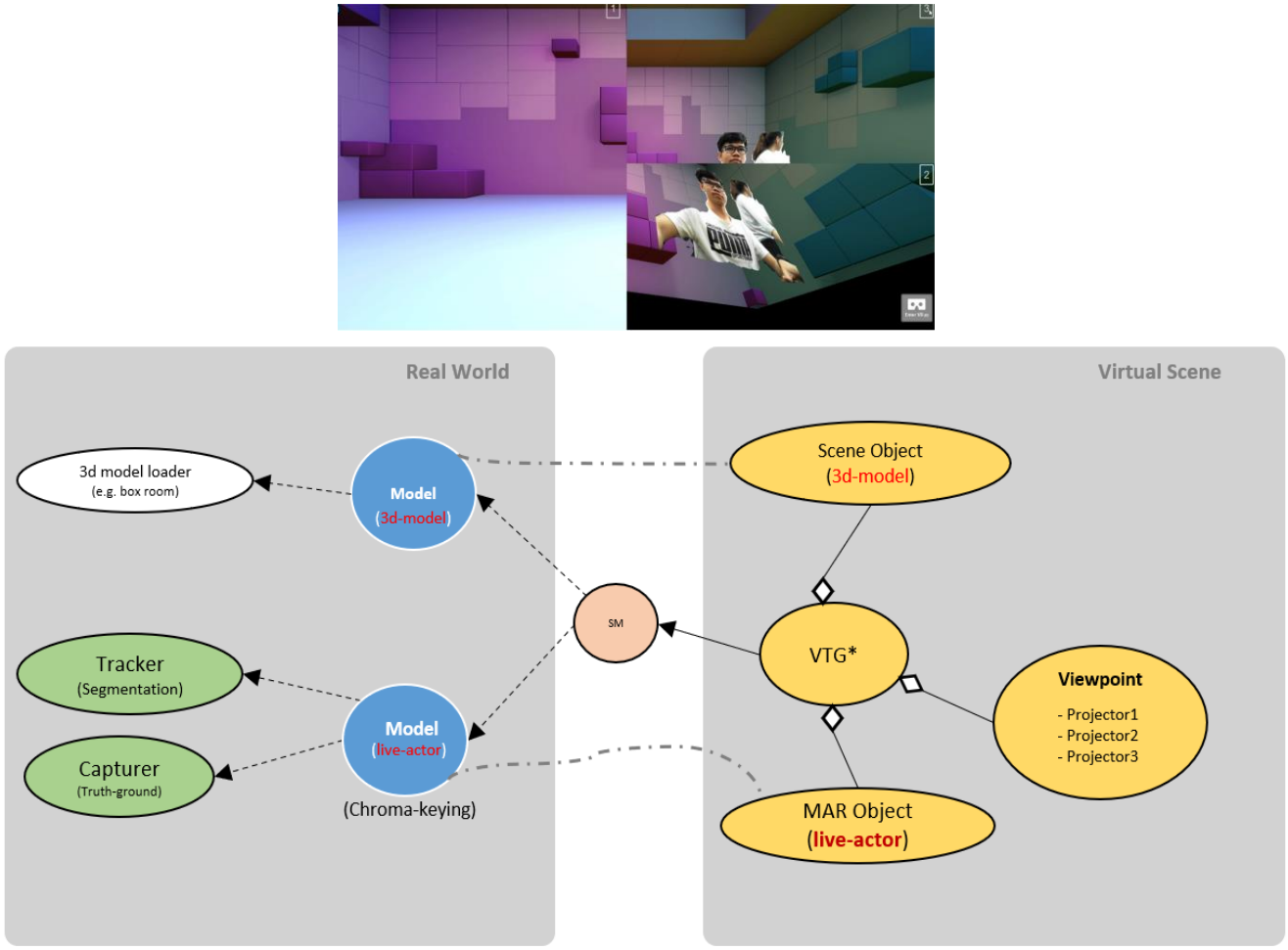



*Figure 12: Scenario example of multi-projection in LAE-MAR system*

— **Virtual Scene**
  - In the virtual scene, the viewpoints or projectors have to be defined since it is useful to capture the story in the scene with three different cameras. Therefore, according on the scenario above, each camera can capture three aspects of view.
  - Also, in the virtual scene, there are two types of virtual object have to be created
    - Scene object is used to load file of 3d model, which is box room
    - Mar Object created for mapping with live-actor model, which made up in real-world side, through SM

— **Real world**
  - There two models constructed to map with two virtual objects in virtual scene
    - For “3d-model” model, through SM, it connected with scene object in virtual scene. This model constructed by “3d-model” loader, which allows to upload a 3d model
    - For “live-actor” model, through SM, it connected with mar object in virtual scene. This model constructed by tracker, which applied the deep learning technique about image segmentation, and capturer, which used to capture sequence of image from real world. Therefore, constructed model produces a chroma-keying live-actor

— Here is the actual implementation of the multiple projection in the scene.

```
<!-- LAE-MAR node -->
  <LAE-MAR id="lae" description="LAE Project" isShownFPS="true" isShownConfig="true">
  <!-- LAE Display -->
  <LAEProjectionDisplay>
    <LAEProjector id="projector1"
                  type="perspective-projector"
                  left="0"
                  bottom="0"
                  width="0.5"
                  height="1.0"
                  position="0 1 3"
                  lookAtPosition="0 0 0"
                  fov="75"
                  nearDistance="0.1"
                  farDistance="100000">
    </LAEProjector>
    <LAEProjector id="projector2"
                  control="orbit"
                  type="perspective-projector"
                  left="0.5"
                  bottom="0"
                  width="0.5"
                  height="0.5"
                  position="0 4 5"
                  fov="75"
                  nearDistance="0.1"
                  farDistance="100000">
    </LAEProjector>
    <LAEProjector id="projector3"
                  type="perspective-projector"
                  left="0.5"
                  bottom="0.5"
                  width="0.5"
                  height="0.5"
                  position="0 2 5"
                  lookAtPosition="0 0 0"
                  fov="75"
                  nearDistance="0.1"
                  farDistance="100000">
  </LAEProjector>
  </LAEProjectionDisplay>
  <!-- MAR SceneRepresentation -->
  <!-- LAE Models -->
  <!-- LAE Mappers-->
</LAE-MAR>
```

### 20.4. Live actor with interaction

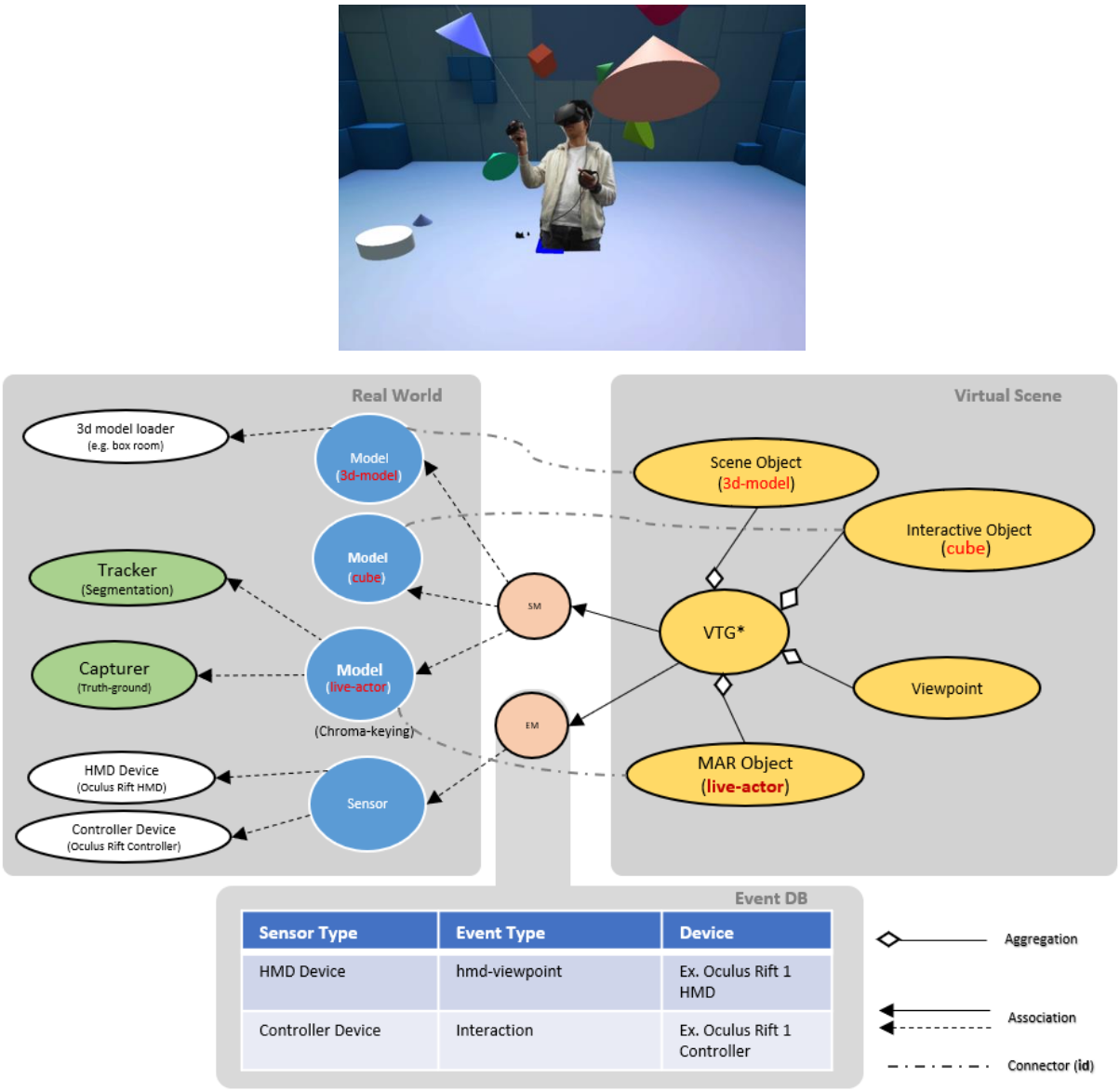



*Figure 13: Scenario example of live actor with interaction in LAE-MAR system*

— **Virtual Scene**
- In the virtual scene, the viewpoint or projector has to be defined since it is useful to capture the story in the scene
- Also, in the virtual scene, there are three types of virtual object have been created
  - Scene Object is used to load file of 3d model, which is box room
  - Interactive Object is made for object, which is interactable
  - Mar Object created for mapping with live-actor model, which made up in real-world side, through SM

— **Real world**
- There three models constructed to map with virtual objects in virtual scene
  - For “3d-model” model, through SM, it connected with scene object in virtual scene. This model constructed by “3d-model” loader, which allows to upload a 3d model
  - “cube” model connected to interactive object in virtual scene. Since it is a built-in model, the cube can be loaded without requiring the model loader.
  - For “live-actor” model, through SM, it connected with mar object in virtual scene. This model constructed by tracker, which applied the deep learning technique about image segmentation, and capturer, which used to capture sequence of image from real world. Therefore, the constructed model produces a chroma-keying live-actor
- Two sensors, HMD device and Controller Device, are mapped by Event mapper. Through the Event Database, HMD device used for “hmd-viewpoint” and Controller Device used for interaction. Therefore, the system allows to use HMD and by controller device, some objects are interactable

— Here is the actual implementation of the interaction of LAE

```
<MAR-LAE id="lae" isShownFPS="true" isShownConfig="false">
    <MARSceneRepresentation id="mar-scene">
        <MARObject id="object1" type="2d-live-actor"></MARObject>
        <MARInteractiveObject id="object2" type="cube"></MARInteractiveObject>
    </MARSceneRepresentation>

    <LAE3DModel id="laemodel1"
                laeCaputurer="laecapture1"
                laeTracker="laetracker1"
                laeRecognizer="laerecognizer1">
    </LAE3DModel>
    <Model id="laemodel2"></Model>
    <LAECapturer id="laecapture1"
          type="general-camera"
          cameraId="0"
          resolution="512x512"
          mode="rgb">
    </LAECapturer>
    <LAETracker id="laetracker1" type="chromakeying"> </LAETracker>
    <LAERecognizer id="laerecognizer1" type="oculus-controller">
    </LAERecognizer>

     <!-- Mappers -->
    <LAESpatialMapper id="laespatialmapper1"
                      model="laemodel1"
                      marObject="object1"
                      position="1 1 1"
                      scale="0.5 0.5 0"
                      rotation="0.01 0.6 0.01">
    </LAESpatialMapper>
    <LAESpatialMapper id="laespatialmapper2"
                      model="laemodel2"
                      marObject="object2"
                      position="1 2 0"
                      scale="1 1 1"
                      rotation="1 1 1">
    </LAESpatialMapper>

    <LAEEventMapper id="laeeventmapper1" marObject="object2" type="click">
    </LAEEventMapper>
  </MAR-LAE>
  <script>
      function onClickEvent(evt, a){
        console.log("I'm fired")
      }

      $('#laeeventmapper1').on( "onclick", onClickEvent);
  </script>
```

## 20.5. Video player

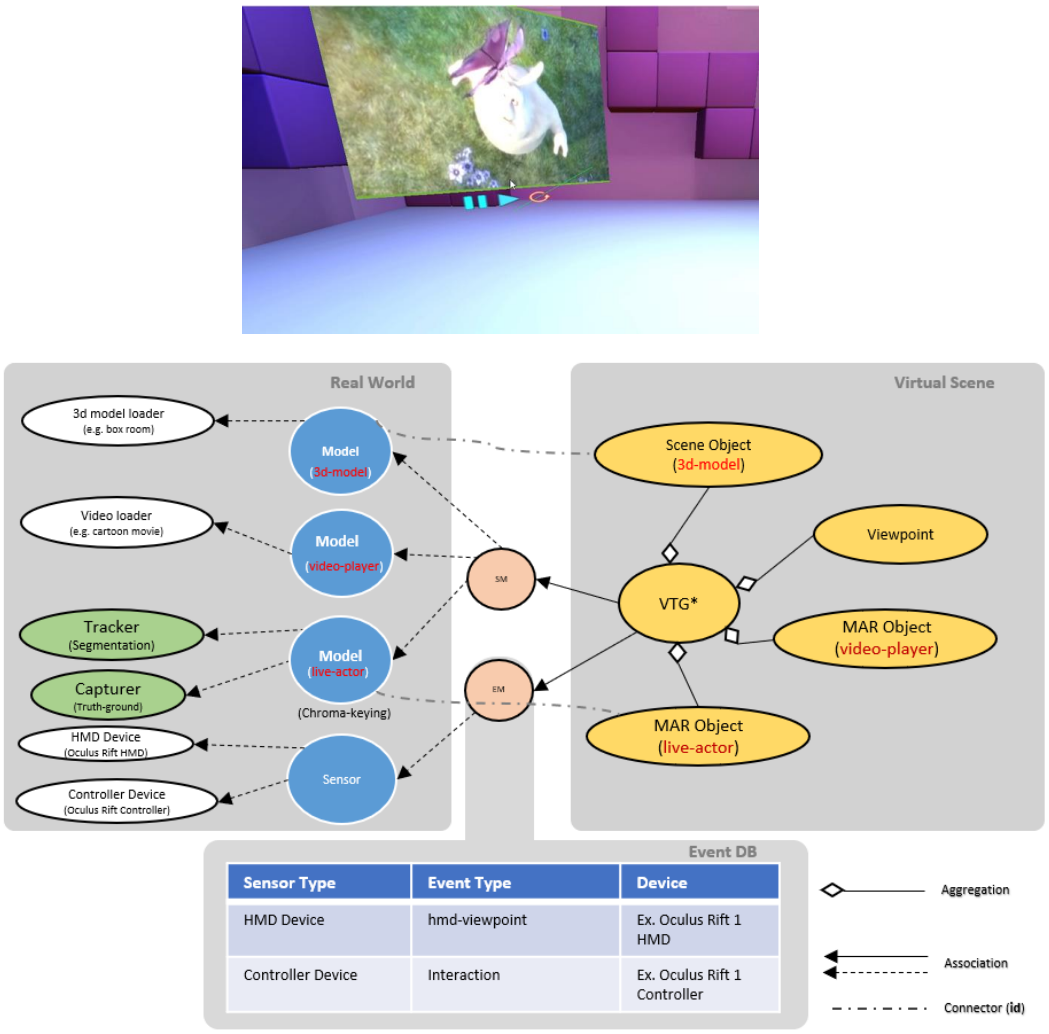



*Figure 14: Scenario example of video player in LAE-MAR system*

— **Virtual Scene**
- In the virtual scene, the viewpoint or projector has to be defined since it is useful to capture the story in the scene
- Also, in the virtual scene, there are three virtual objects, one scene object and two mar objects, have been created
  - Scene Object is used to load file of 3d model, which is box room
  - Mar Object created for mapping with "live-actor" and "video-player" model, which made up in real-world side, through SM

— **Real world**
- There three models constructed to map with virtual objects in virtual scene
  - For "3d-model" model, through SM, it connected with scene object in virtual scene. This model constructed by "3d-model" loader, which allows to upload a 3d model
  - For "live-actor" model, through SM, it connected with mar object in virtual scene. This model constructed by tracker, which applied the deep learning technique about image segmentation, and capturer, which used to capture sequence of image from real world. Therefore, constructed model produces a chroma-keying live-actor
  - For "video-player" model, through SM, it connected with mar object in virtual scene. This model constructed by video loader, which allows to load video file. Also, the video player is interactable and can be controlled by controller device as defined.

- Two sensors, HMD device and Controller Device, are mapped by Event mapper. Through the Event Database, HMD device used for "hmd-viewpoint" and Controller Device used for interaction. Therefore, the system allows to use HMD and by controller device, some objects are interactable

— Here is the actual implementation of the video player.

```
<!-- MAR-LAE node -->
<LAE-MAR id="lae" isShownFPS="true" isShownConfig="true">
    <!-- LAE Display -->
    <!-- MAR SceneRepresentation -->
    <MARSceneRepresentation id="mar-scene">
        <MARAugmentedObject id="panel1" hidden="false">
          <MARObject id="videoobject" type="video-texture"></MARObject>
          <MARInteractiveObject id="playbtn" type="cube" transparent="0.4">
          </MARInteractiveObject>
          <MARInteractiveObject id="stopbtn" type="cube" transparent="0.4">
          </MARInteractiveObject>
        </MARAugmentedObject>
    </MARSceneRepresentation>
    <!-- LAE Models -->
    <Model id="videomodel" src="templates/media/videos/mov_bbb.mp4"></Model>
    <Model id="btn"></Model>

    <!-- Mappers -->
    <LAESpatialMapper id="spatialmapper1" model="videomodel" marObject="videoobject"
      position="-9.8 16.5 -36.7" scale="7.4 3.3 0.5" rotation="0 0 0"> </LAESpatialMapper>
    <LAESpatialMapper id="spatialmapper2" model="btn" marObject="playbtn"
      position="-9.9 12.7 -36.7" scale="1.5 0.8 0.3" rotation="0 0 0"></LAESpatialMapper>
    <LAESpatialMapper id="spatialmapper3" model="btn" marObject="stopbtn"
      position="-11.9 12.7 -36.7" scale="1.5 0.8 0.3" rotation="0 0 0"></LAESpatialMapper>

    <LAEEventMapper id="eventmapper1" marObject="videoobject" type="onvideo"/>
    <LAEEventMapper id="eventmapper2" marObject="playbtn" type="click"/>
    <LAEEventMapper id="eventmapper3" marObject="stopbtn" type="click"/>
</LAE-MAR>

<script>
    function onClickToPlay(evt, a){
      console.log("Play video...")
    }
    function onClickToStop(evt, a){
      console.log("Stop video...")
    }

    $('#eventmapper1').on( "onvideo", (evt, { videoEle })=>{ });
    $('#eventmapper2').on( "onclick", onClickToPlay);
    $('#eventmapper3').on( "onclick", onClickToStop);
</script>
```

## 21. Conformance

[…]

## Bibliography

[…]