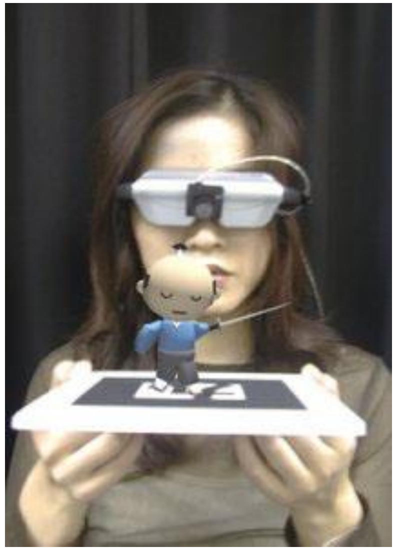
Progress and Updates on Information Model for MAR Contents (ISO 21858)

Jan 23, 2019

Gerard J. Kim
Korea University

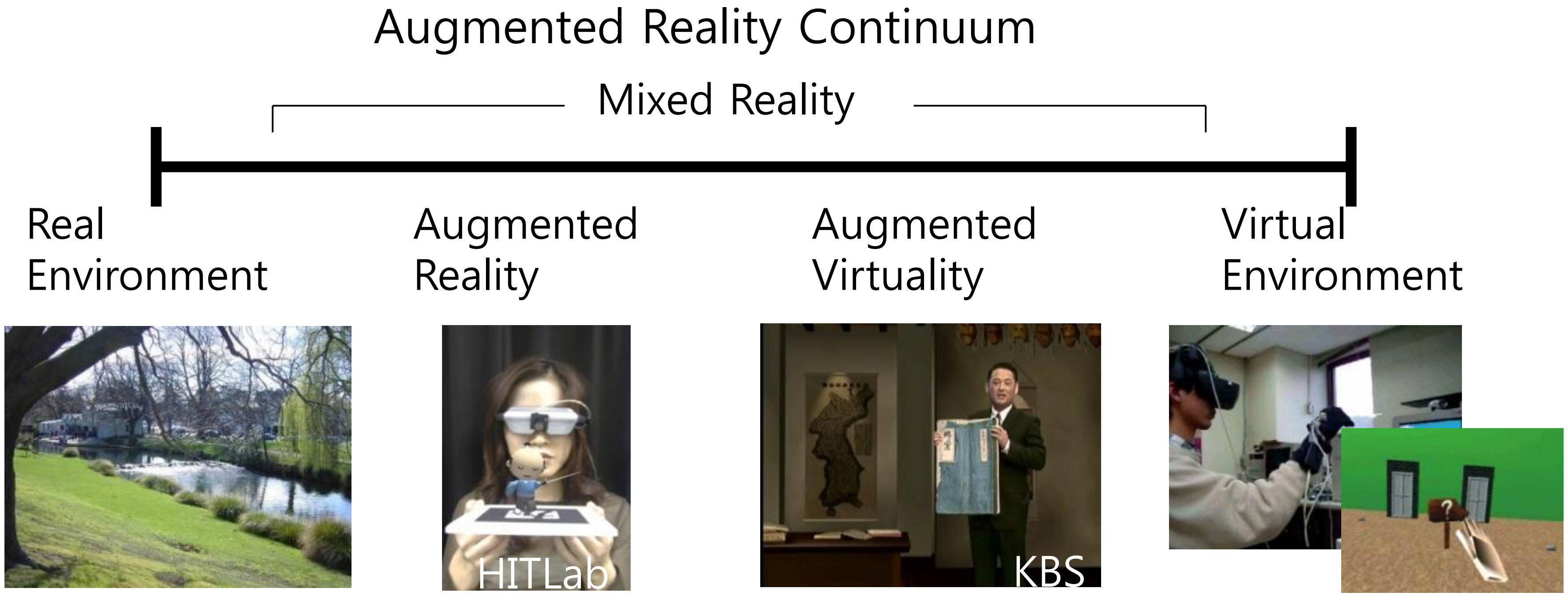
Mixed and Augmented Reality (MAR)

- What is AR (Augmented Reality) ?
  - "Augmented Reality (AR) is a field of computer research which deals with the combination of real-world and computer-generated data." – wikipedia.org



- Key Features of AR [R. Azuma 97]
  - Combines real and virtual images
  - Interactive in Real-Time
  - Registered in 3D Real World

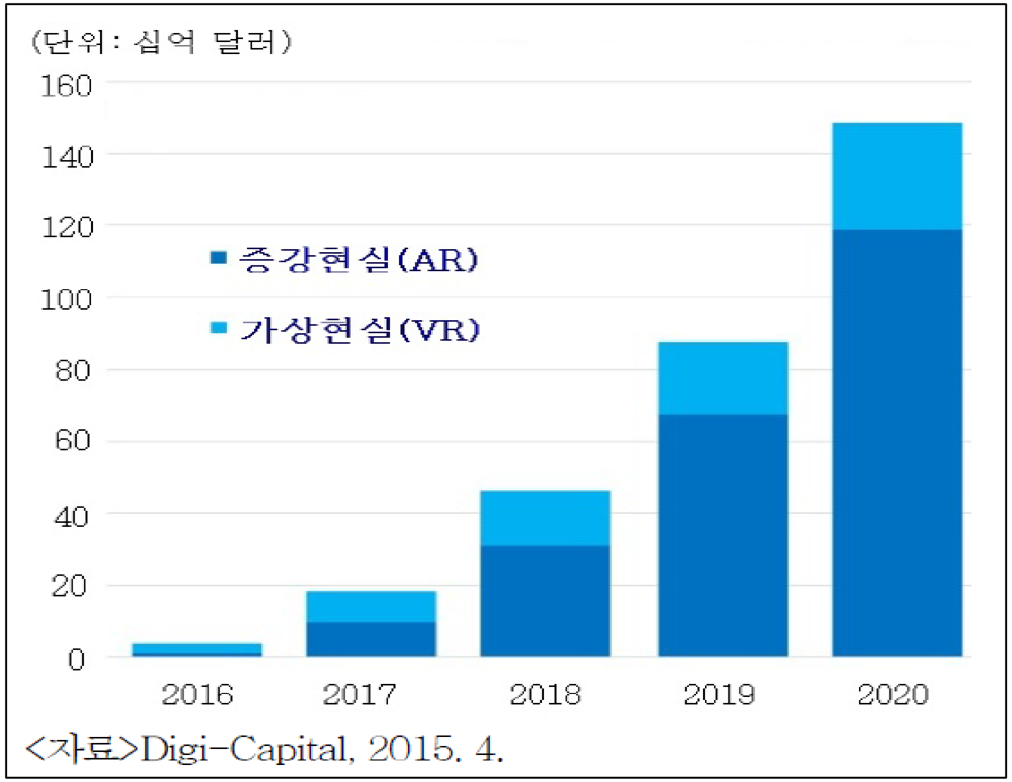
Mixed (and Augmented) Reality Continuum



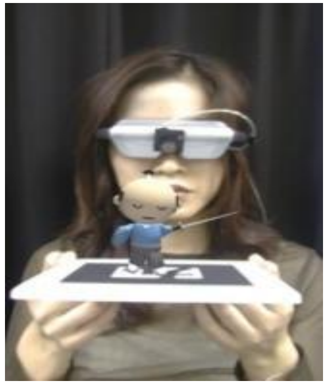
[Paul Milgram's Reality-Virtuality Continuum (1994)]

The Need

- Mixed reality (or augmented reality) has become possible on commodity hardware (e.g. smart phone) and through cloud services - Processing
- Wearable computing
  - Sensors and displays
  - Environment sensors !
- Internet of Things
- Content creation
- Browser + contents model
  - Share contents!



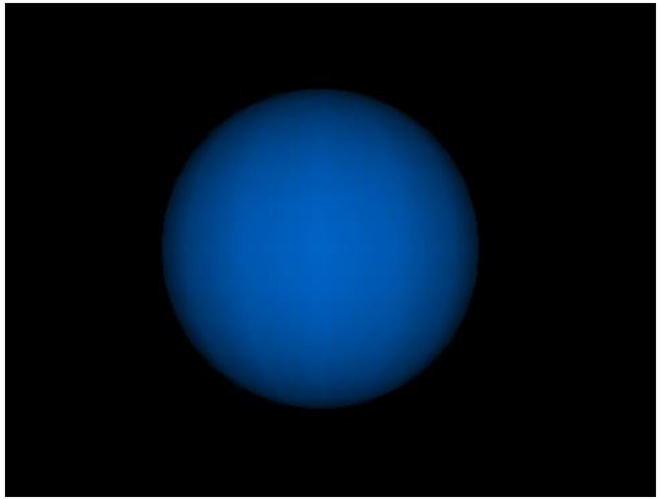
MAR is “implemented” as a VR system



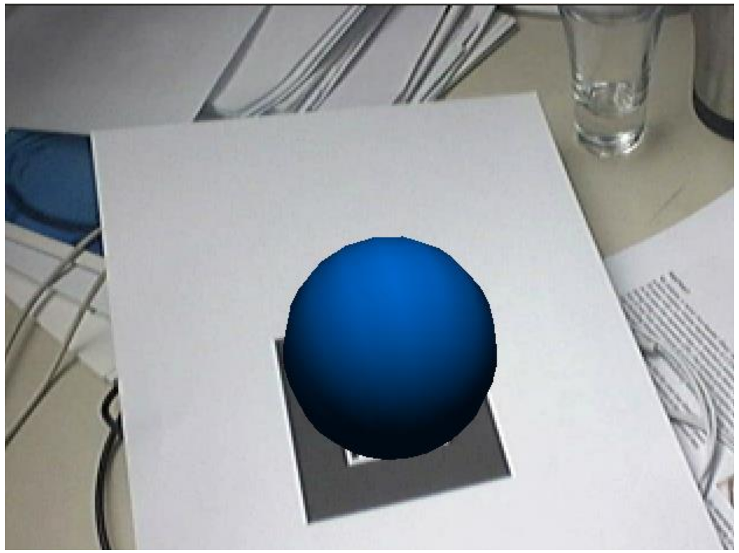
- E.g. Video see through AR
  - Real world is captured as a video
  - Target objects are identified and their spatial information obtained (sensed)
  - A virtual space is created in which the video is put in the background and other synthetic virtual objects are put into this space (using the obtained spatial information) and rendered

- Natural direction
  → Extend current virtual space representation for MAR

Extending contents to be MAR capable!



?



What do we need? *Mix virtual and real*

HTML/document in real (e.g. video)

Video in virtual

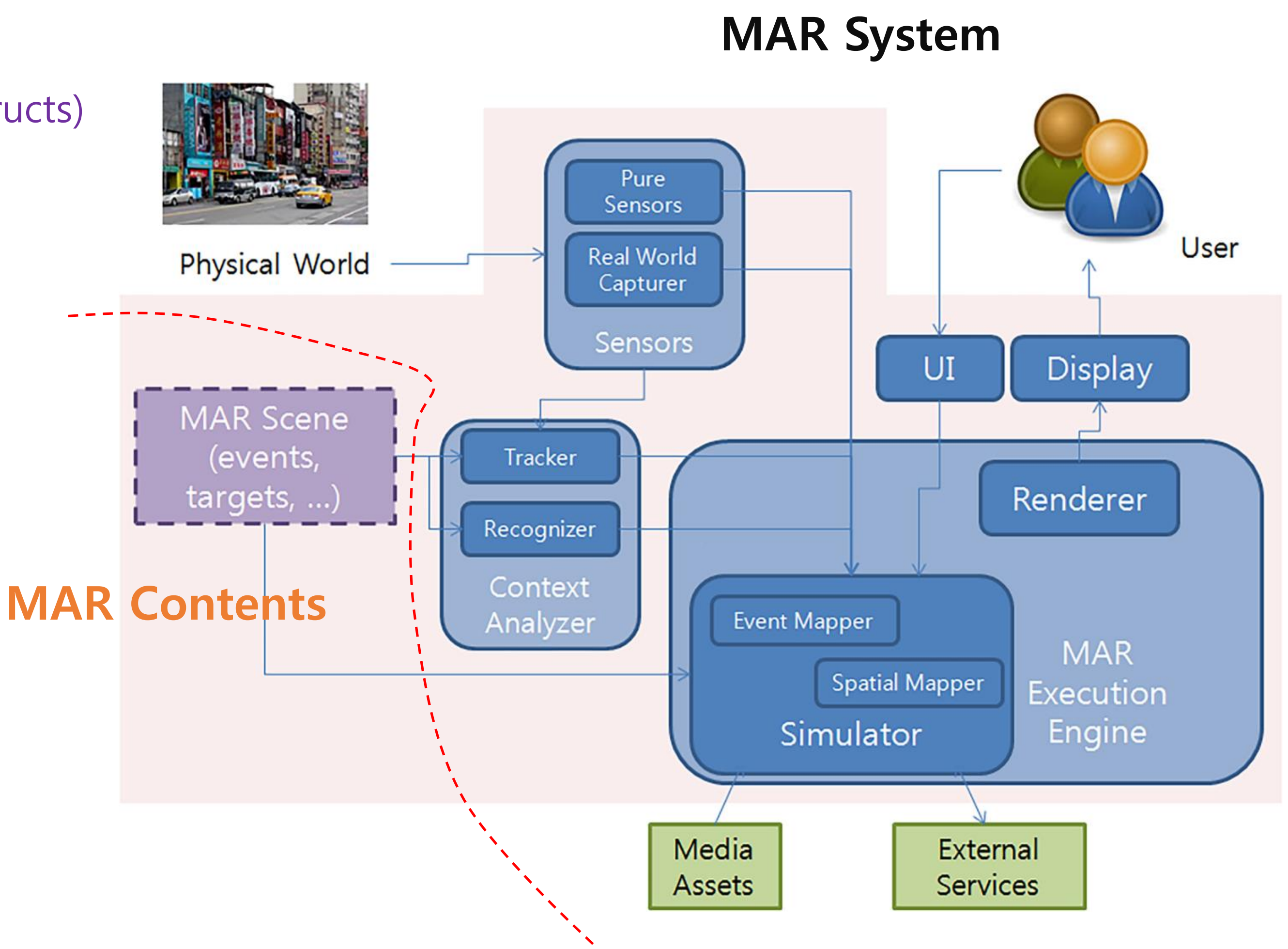
Real (e.g. image) in Real (e.g. image)

Virtual in HTML/document (virtual)

…

Approach for MAR content model – Component based

- Identify chunks of information needed to represent various MAR contents and system classes (AR/AVR)
  - Define functionality or content type by mix and match (association)
    - Real objects
    - Virtual objects (use existing constructs)
    - MAR scene structure (use/modify existing constructs)
      - Real – virtual association
      - Mutual placeholder designation
      - Registration
    - MAR events and behaviors
      - Augmentation information and their style
    - Sensors and real world capture
      - Backdrop world representation



- Abstract out details
  - Easy to use and understand
  - Minimize any system specific scripting/programming

Status – ISO/IEC AWI 21858: "Information model for MAR contents"

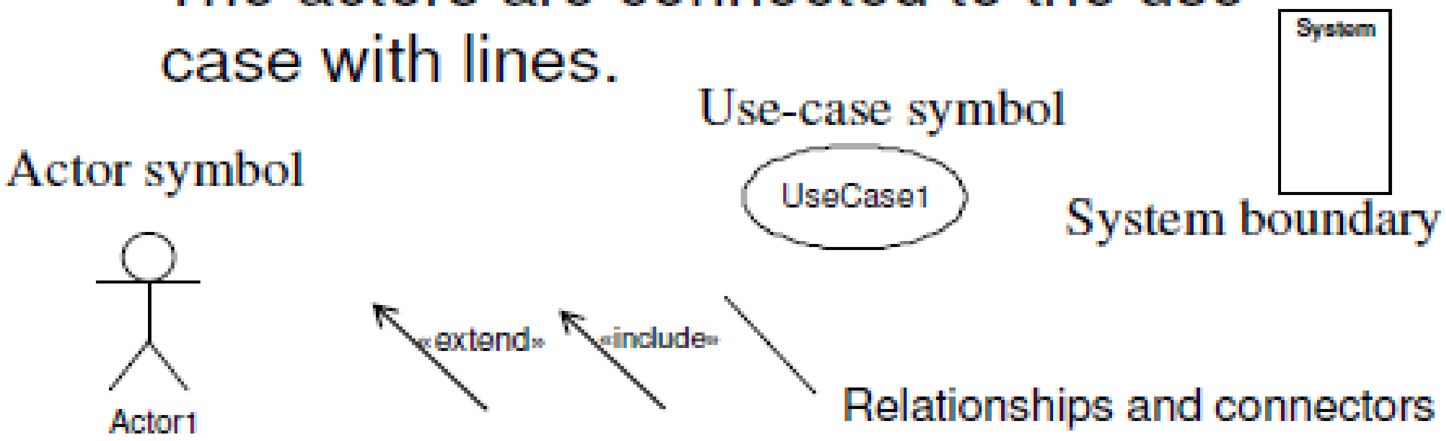
- Approved as a new work item proposal
  - April, 2016 (N3808 / N3809)
- Target date: 2018-08-08 → 2018-12-12?
- Working on the CD document – being delayed …
  - Component identification
  - Object-class diagram (UML)
  - Detailed information/object modeling
    - Attributes and data type
  - Use cases
  - Implementation

Related works

- Most MAR systems implemented as a single application using programming libraries (e.g. AR Toolkit)
- Separation of contents and browser – started with location based AR (Wikitude, Layar, …)
  - ARML (Augmented Reality Markup Language) allows defining geographical points or landmarks of interest and associate GPS coordinates and simple augmentation contents
  - Adopted as a standard for the Open Geospatial Consortium
- X3D: Extended nodes to support e.g. video see-through based AR, such as the live video background, extended camera sensor nodes
  - MPEG: Application format for video augmented content (ARAF)
- InstantReality, AWE, Google ARCore … : Declarative scene description + Scripted AR functionality
- Still limited
  - Not comprehensiveness
  - Lacks sufficient abstraction
  - Lacks clean modularization requiring lengthy and complicated script programming

UCD Components

- The use case itself is drawn as an oval.
- The actors are drawn as little stick figures.
- The actors are connected to the use case with lines.



UML Fundamentals, Ernest Cachia, 2004

UML Use-Cases (UCs not UC Diagrams UCDs)

Definition: *"A set of sequences of actions a system performs that yield an observable result of value to a particular actor."*

Use-case characteristics:

- Always initiated by an actor (voluntarily or
- involuntarily);
- Must provide discernible value to an actor;
- Must form a complete conceptual function.

(conceptual completion is when the end observable value is produced)

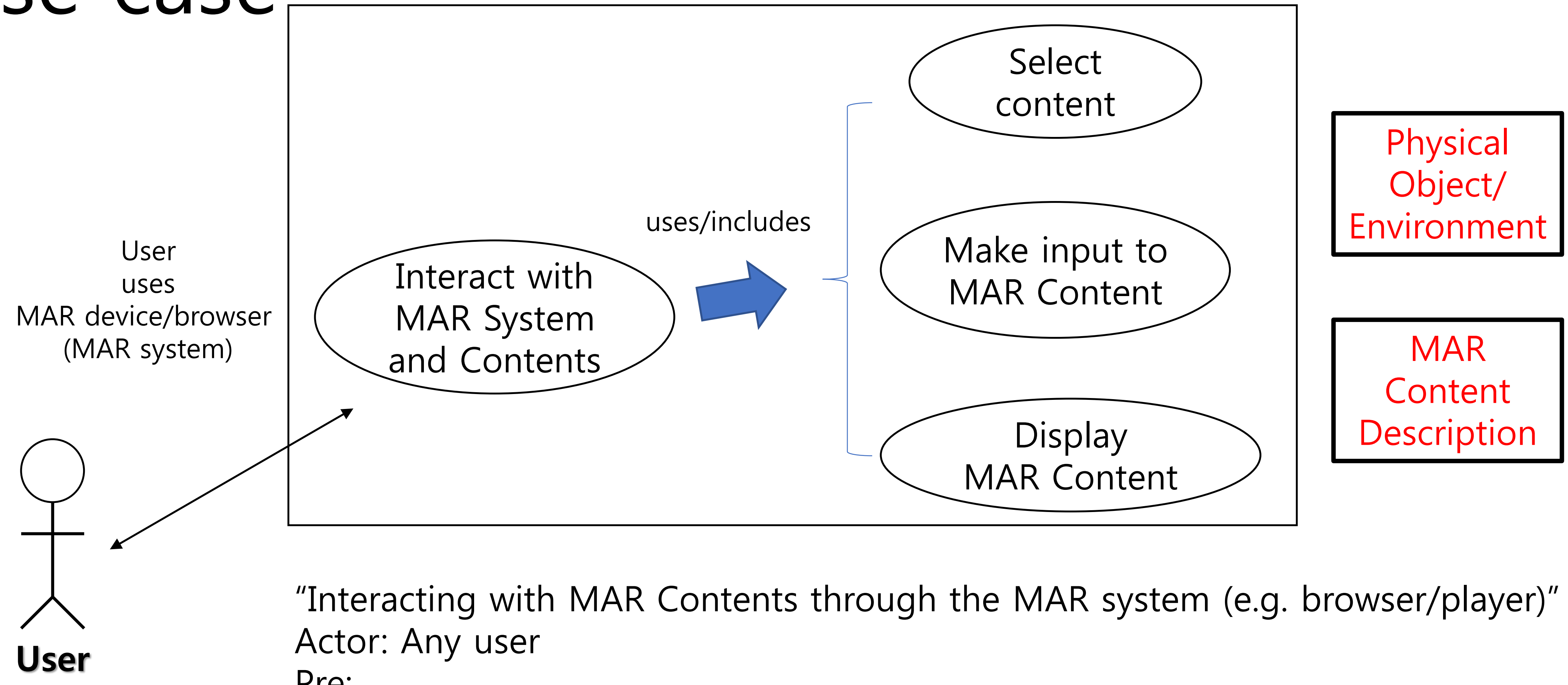
UCD Relationships (1/2)

- Association relationship
- Extend relationship

- Include relationship

- Generalisation relationship

Use case



"Interacting with MAR Contents through the MAR system (e.g. browser/player)"

Actor: Any user

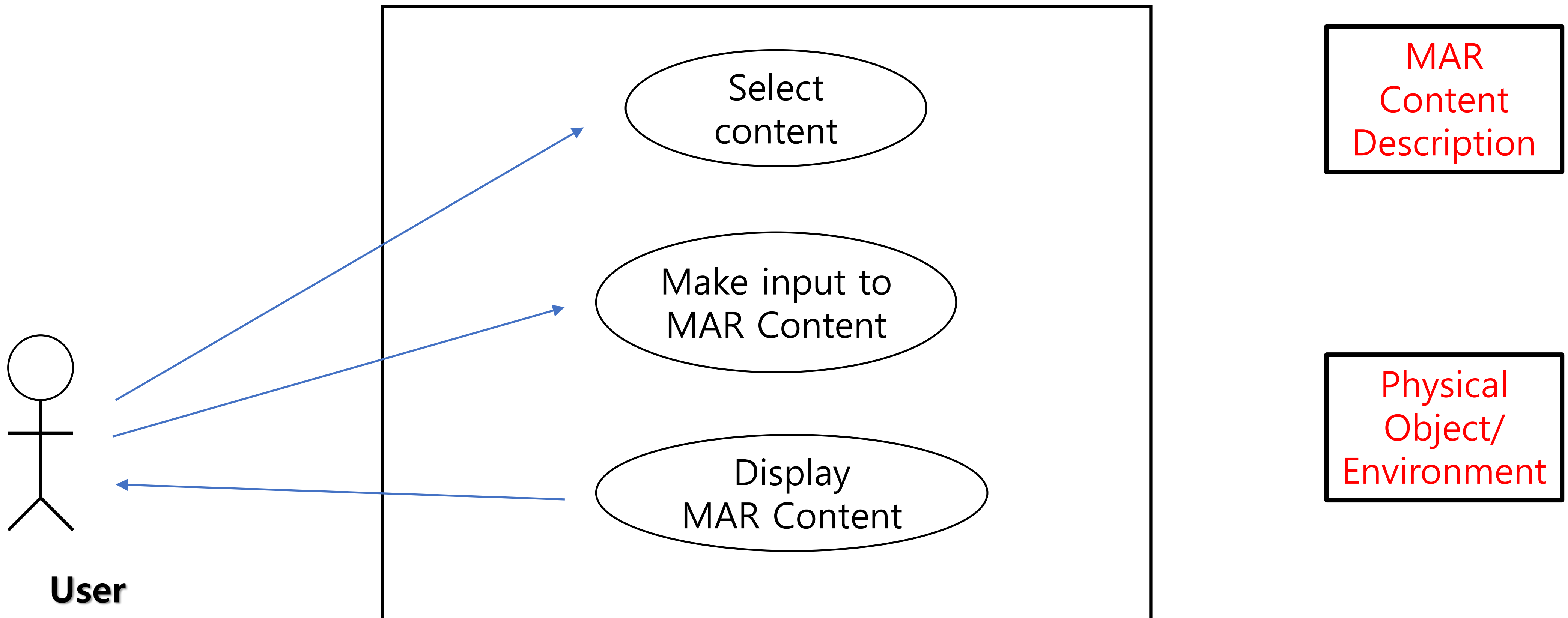
Pre:

- User has MAR device that is equipped with MAR browser
- MAR content description is given/selected by the user

Post:

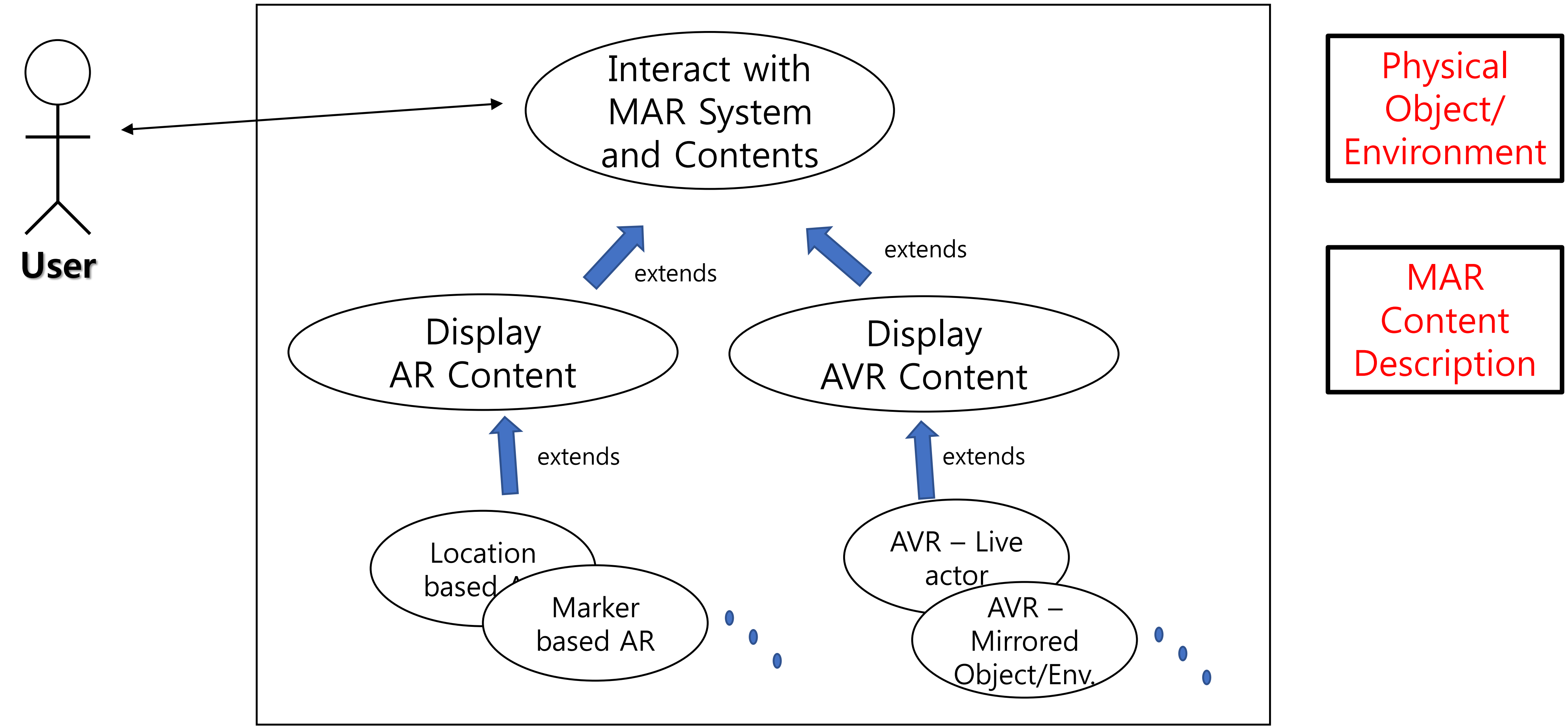
- User sees through the device/browser augmented physical environment according to the MAR content description

Use case


Select content
Make input to MAR Content
Display MAR Content
User
MAR Content Description
Physical Object/ Environment

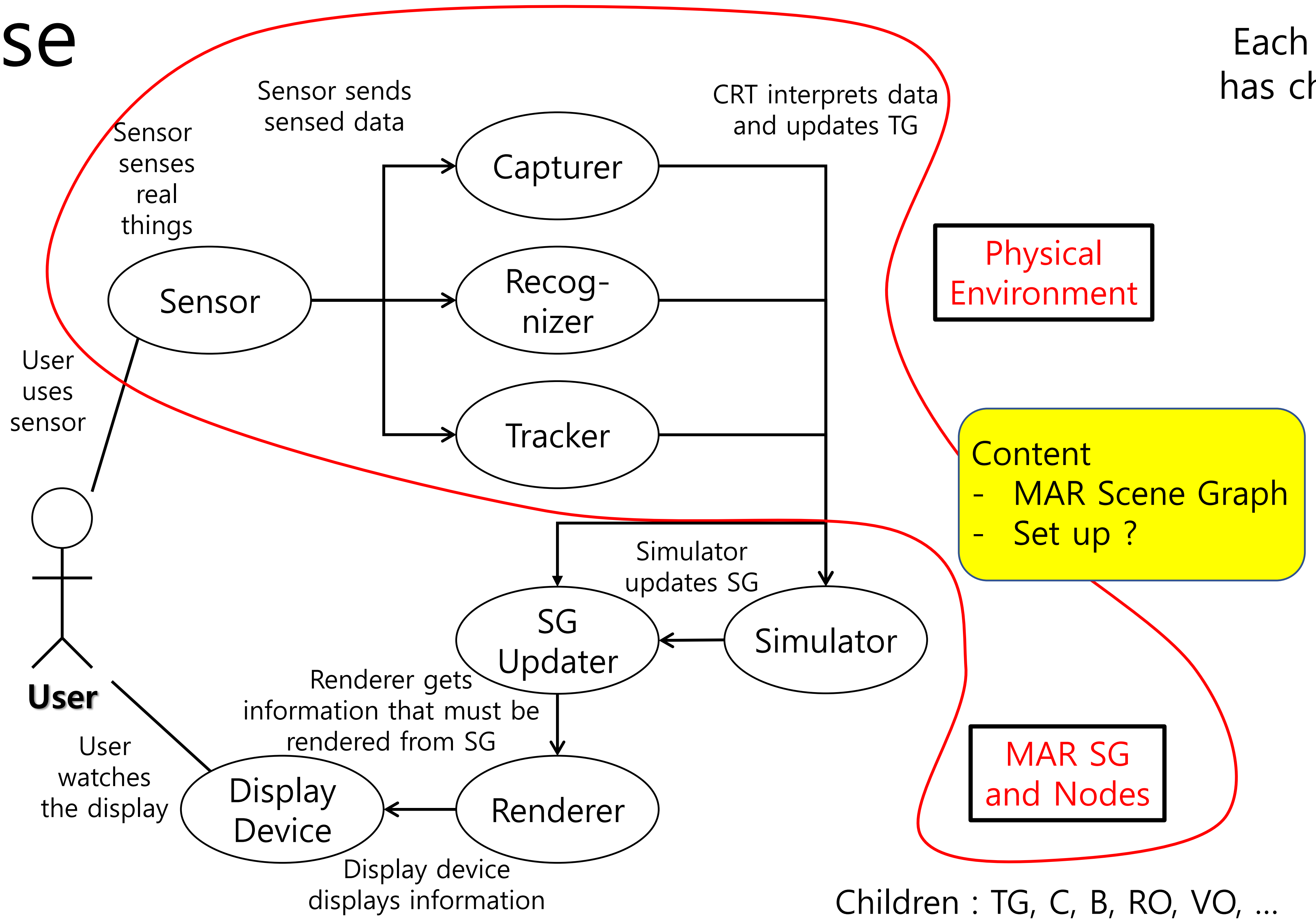
Use case

User
uses
MAR device/browser
(MAR system)



Use case

Each circle has children



Children : TG, C, B, RO, VO, …

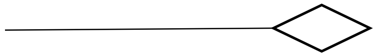
MAR Content/Scene

- Represented as an hierarchical and graphical organization of objects (nodes) in the “mixed and augmented reality” scene
- Nodes represent:
    - All nodes are subclasses of the abstract MARSGNode
    - Objects in the scene (virtual and real)
        - Those that are purely computational/functional
        - Those that have appearances and to be rendered in different modalities (e.g. visual, aural, tactile, haptic, …)
    - Coordinate systems and spatial relation/grouping
    - (Explicit) Registration between real objects and virtual objects
    - Logical/Spatial grouping

MARSGNode			
Access type	Data type	Attribute/Method name	Explanation
private	string	id	a unique identifier for reference
private	MARSCNode[]	parent	parent nodes (usually, there is only one parent)
private	MARSCNode[]	childrenNodes	a list of or array of one or more children nodes, also of the MARSGNode (or its subclass) type
private	Cube	bounding-box	A bounding box specification of for the object this node represents in th e MAR scene (optional).
public	MARSCNode	MARSCNode()	MARSCNode constructor
public	void	init()	abstract initializing method for the MARSCNode class
public	string	getId()	return the string id of this node
public	void	setId(string id)	set the id of this node
public	void	addChild(MARSCNode child)	add a child to this node of MARSGNode type (or its subclass)
public	void	removeChild(MARSCNode child)	remove a child to this node of MARSGNode type (or its subclass), if it exi sts.
public	void	removeAllChild()	remove all children nodes, if any
public	MARSCNode[]	getChildren()	return the list/array of children nodes
public	Cube	getBoundingBox()	recompute and update the bounding box for this node considering all th e sub-objects to this node and update the attribute bounding-box
public	MARSGNode[]	getParent()	return the list/array of parent nodes

Relations (connections among nodes)

- Aggregation (depicts a classifier as a part of, or as subordinate to, another classifier – ibm.com)
  - Between parent and children TransformGroup



    - Spatial placeholder and relationship
    - Logical/Spatial grouping
      - A group node consists of its children conceptually who all share the same transform (real or virtual)
      - E.g. a group may contain a virtual object, bounding box, all sharing the same transform (or coordinate system)
- Association and Dependency $\longrightarrow$ $\longleftrightarrow$

  (objects of one classifier connect and can navigate to objects of another classifier – ibm.com)
  - One node's attribute value refers/changes information from another node through named attribute
    - Can be one directional or bidirectional

A $\longrightarrow$ B

B is navigable (accessible) from A
A can change B

MAR system

- The system that take the MAR content (selected by the user), other user input (as occurring during user interaction with the content) and simulates and displays/presents the interactive content to the user
- Assume that there is an MAR system with the following components according to the MAR Reference Model (18039)
  - Real capturer
  - Recognizer
  - Tracker
  - MAR simulation engine
  - Display/Renderer
- MAR Contents has "relationships" (aggregation and association) to the underlying MAR System (see later slides), e.g.
  - Sensors
  - Capturer → Object nodes, Behavior nodes, …
  - Tracker → TG nodes
  - Recognizer → Behavior nodes, …
  - MAR Scene → Simulation engine

CRT

TransformGroup (TG)

- Specifies coordinate system and spatial relationship with respect to a reference parent coordinate system (or TransformGroup)
  - Translation, Rotation and Scaling
- TG represents a particular spatial placeholder in the given environment
  - Aggregation relationship with parent TG
  - If there is no explicit parent, the parent is the assumed root TG
- Two subclasses
  - RealTG – a spatial placeholder in the physical space
    - Assume that there exists a corresponding root TG node
  - VirtualTG – a sptial placeholder in the virtual space
    - Assume that there exists a corresponding root TG node
  - In principle, there exists a "Registration" association class between/among heterogeneous TG's
    - Explicitly represents the "augmentation" e.g. between RTG and VTG
    - Explicitly represents the merging of heterogeneous worlds, e.g. among separately constructed VTG's, separate RTG's, …
- Also represents the notion of a group (aggregation) of different object information that shares a common coordinate system

- TransformGroup may be implemented in a different way (e.g. as separate but related/associated classes/objects)

(Spatial) Registration

- Association class among/between TG's (real or virtual)
- Explicitly represents the "augmentation" e.g. between RTG and VTG
  - In principle, we choose not to omit the explicit representation (superfluous?)
- Explicitly represents the merging of heterogeneous worlds, e.g. among separately constructed VTG's, separate RTG's, …
- May specify the method of registration, if needed
  - Usual scaling, rotation and translation by computation is omitted ?
- Actual spatial transformation is contained in the associated TG's
- Registration can exist between TG's and MAR system (e.g. CRT) whose values may affect TG and need adjustment (e.g. sensor registration – part of the content?)

(Event/data) Mapper

- Event/Data: Particular type of "data/event" that is used to drive MAR simulation/behaviors
  - Data: Any piece of information with a value and occupy memory location
  - Flows between MAR system (e.g. sensor, CRT) and contents
- In X3D, for events are just any data values of attributes that can propagate through routes (or through association)
- Here we assume that different events and data types exist
  - Event: Object existence, Object pose/location, User interaction (touch, gesture, click), Context (time, identity, location) user defined, …
  - Data: Tracking information (Object pose/location), Sensor data

- Event/data Mapper
  - Association class among/between event generator (e.g. sensor, CRT) and its user (e.g. behavior)
  - Map system defined items to content defined items
    - GPS 100, 100 → Korea University
  - Data filtering, conversion, scaling, etc.

ObjectNode

- Specifies a particular object, virtual or real
  - Real objects provide description of things like real objects often used in MAR such as markers, image patches, GPS location, etc. → See later detailed specification
    - Eventually any physical/real object description should be supported
  - Virtual objects provide descriptions like any graphical, computational and synthetic objects like text, image, animation, 2D shapes, 3D shapes, bounding box, light, viewpoint, etc.
    - Ordinary computer graphics scene graph (like X3D, Java3D) will have similar support for these
      - This document need not describe detailed virtualobject subclasses
    - In addition some special VirtualObjects will be assumed to exist:
      - Live background node (see Gun Lee's work)
        - Used in video see through AR
      - Live moving texture node (see LAE work from Prof. Yoo)
        - Used in augmented virtuality for live captured object in 2D (e.g. chroma-keyed live actor)

Behavior

- Specifies dynamics of virtual objects in time
- Often amounts to a script with arguments from associated other nodes
- Often used behaviors are abstracted for ease of use
  - Simple visibility (i.e. show objects): Appear/disappear
  - Animated objects: Fixed translation/rotation/scaling, Animation files
  - Highlighted effects: Blinking, transparency, color, …
  - …
- Associated with MAR system (sensor, capturer, tracker, and recognizer) to receive events and data that will drive the behavior simulation
- Associated with other objects/nodes on which the behavior operates on

MetaInfo

The MetaInfo component optionally adds to the basic content of contextual and additional information about various content constructs – such information may include user(s)/author characterization and intent of the associated content component.

MetaInfo::User			
Access type	Data type	Attribute/Method name	Explanation
private	string	info	Meta information about the associated object
private	MARSGNode	about	cmponent object this meta information i s associated with in text

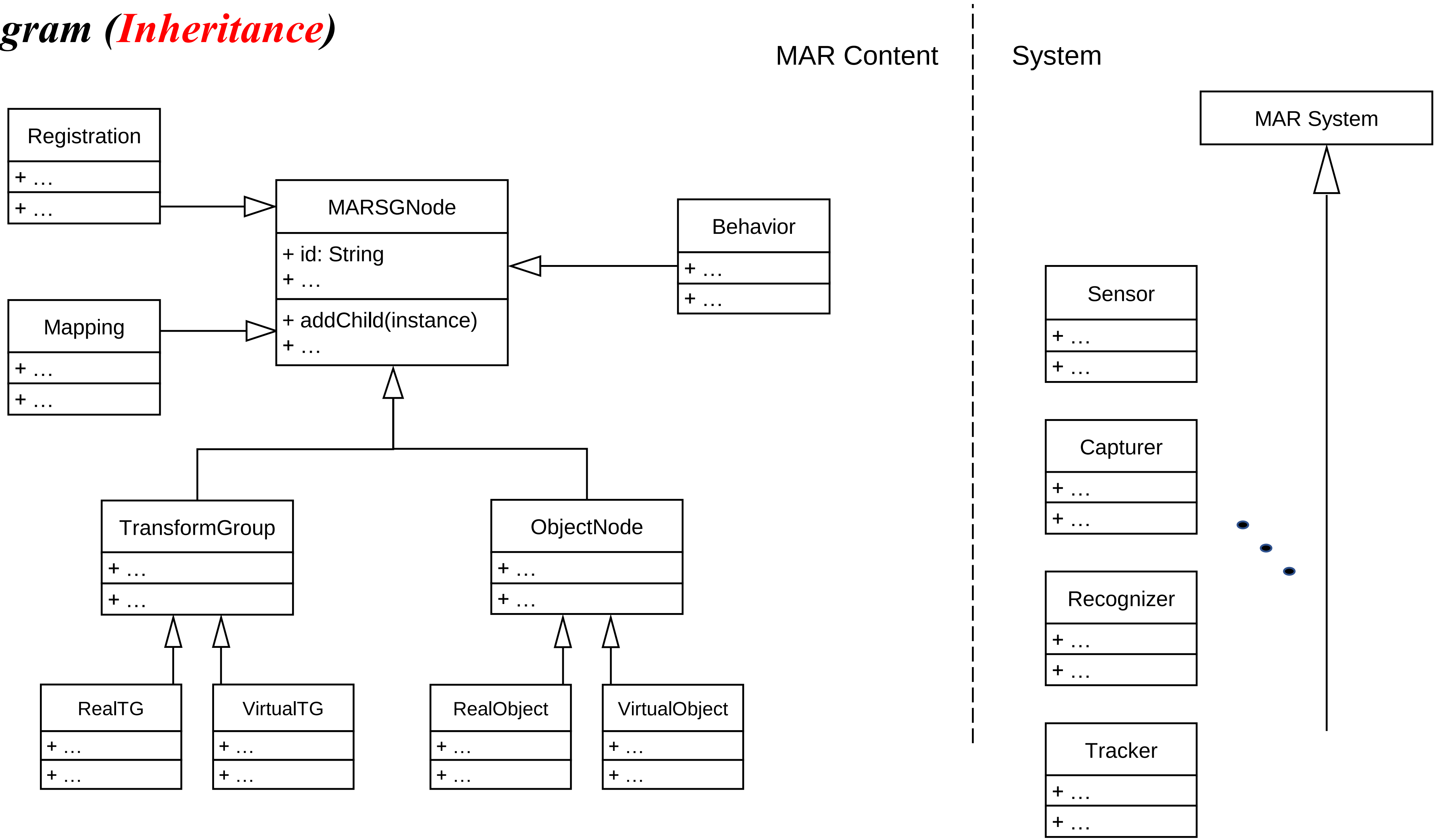
MAR System: Sensor - Capturer, Tracker, and Recognizer

- These objects get data from the sensor for additional processing and also produces stream of events/data like the sensors
  - They could be modeled as a subclass of Sensor (but not for now)
- Capturer captures real objects as a whole in some way (e.g. environment background, live human actor, 3D object reconstruction, etc.)
- Tracker returns dynamic and continuous position/rotation/pose date of a physical object
- Recognizer returns discrete events

Event/Data

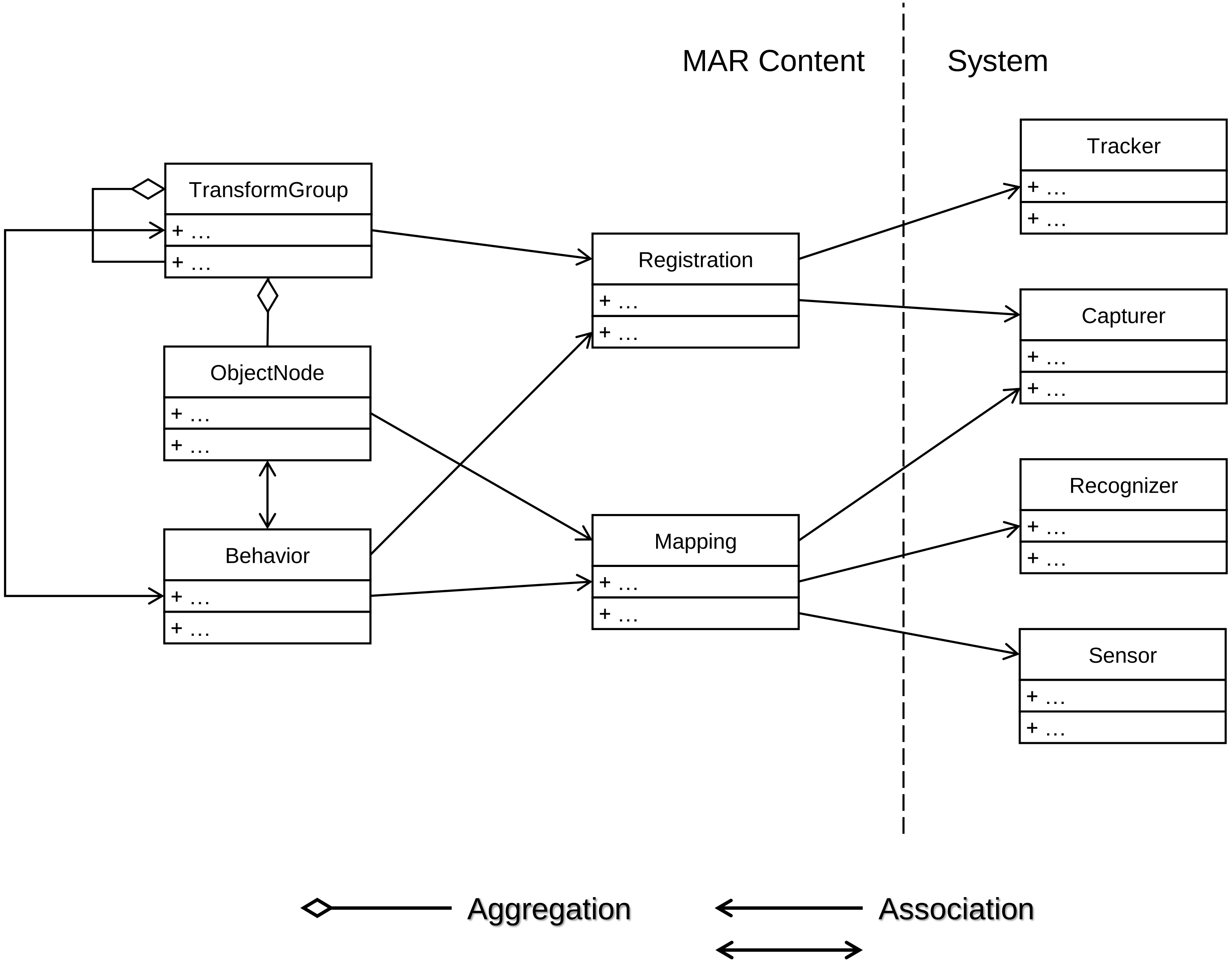
- Particular type of “data” that is used to drive MAR simulation/behaviors
  - Data: Any piece of information with a value and occupy memory location
  - Flows between MAR system and contents
- In X3D, for comparison, events are just any data values of attributes that can propagate through routes (or through association)
- Here we assume that different events and data types exist
  - Event (Discrete)
    - Object existence
    - Object pose/location
    - User interaction: Touch, gesture, click, …
    - Others: Context (time, identity, location, pose …), user defined, …
  - Data (Continuous)
    - Tracking information
    - Sensor data

UML Class Diagram (Inheritance)



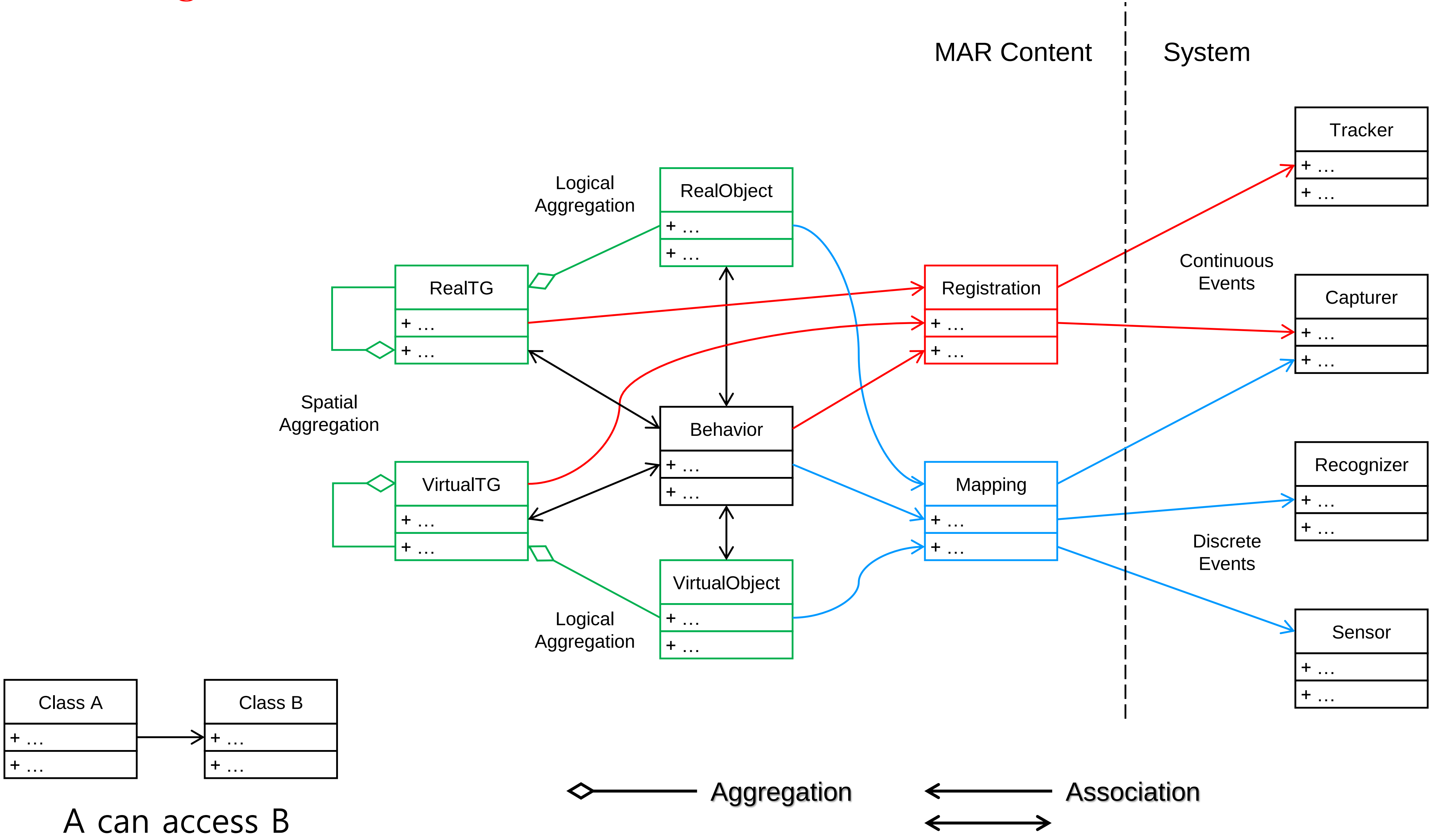
참고자료: https://www.lucidchart.com/pages/uml-class-diagram

UML Class Diagram (Aggregation and Association)



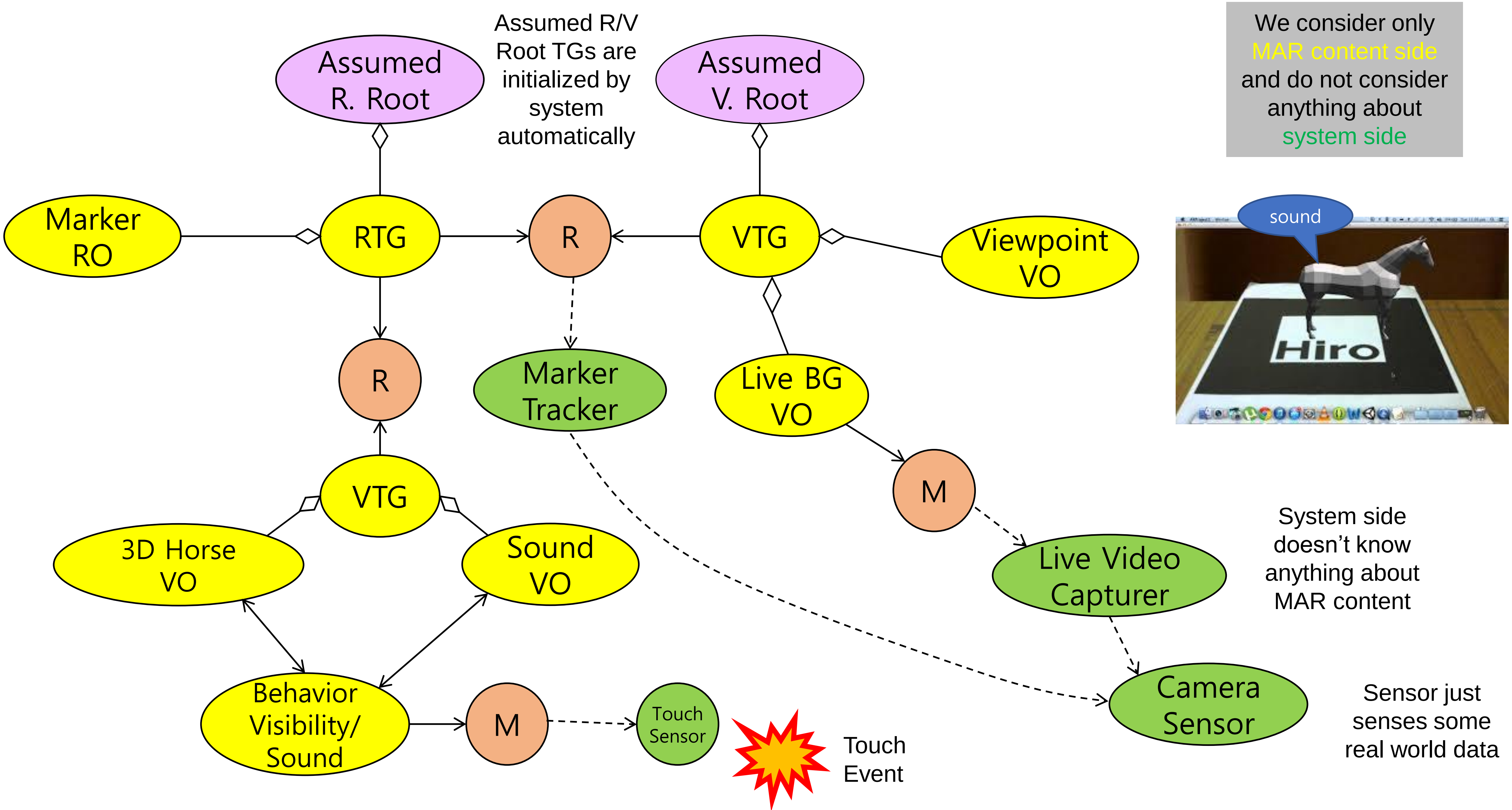
https://www.lucidchart.com/pages/uml-class-diagram

UML Class Diagram

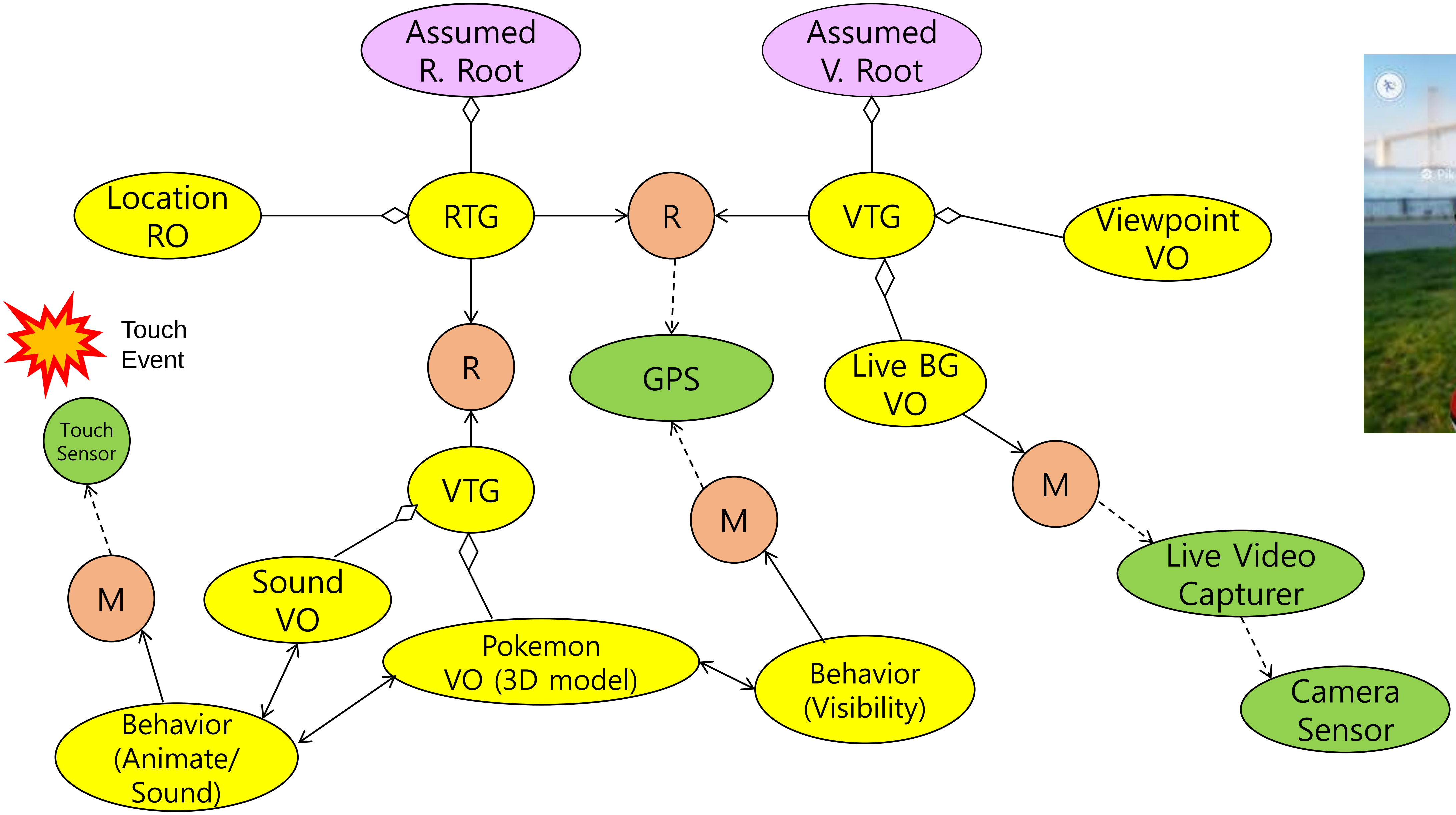


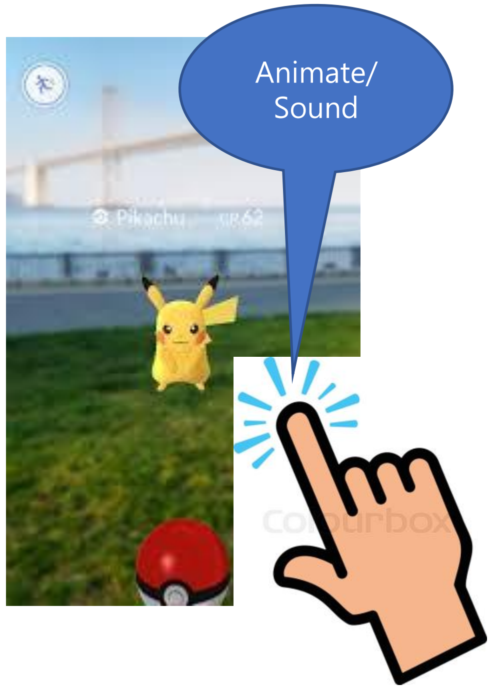
참고자료: https://www.lucidchart.com/pages/uml-class-diagram

Scenario (Marker or Image patch based, Video see-through)

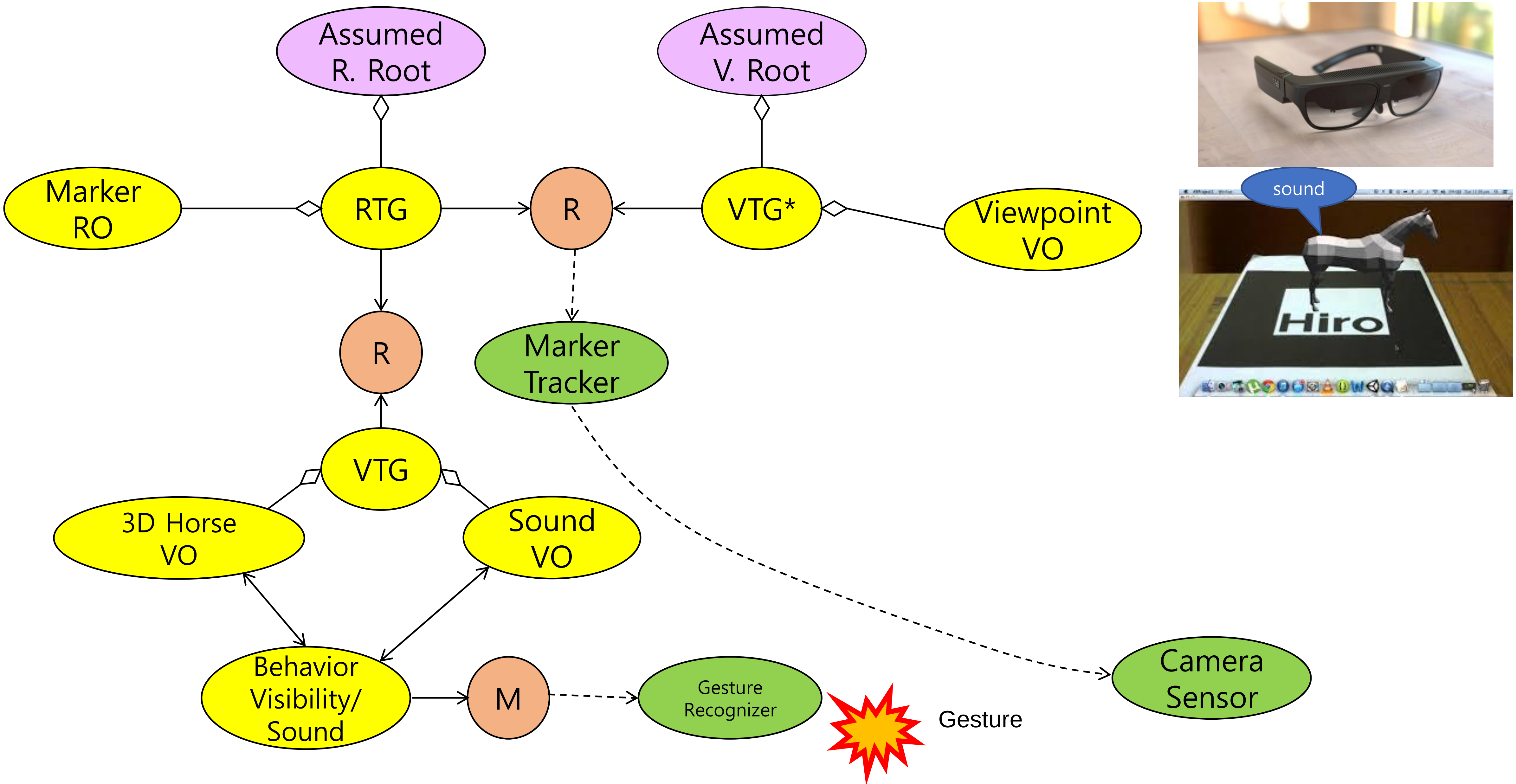


Scenario (GPS based / Video see-through / Pokemon)

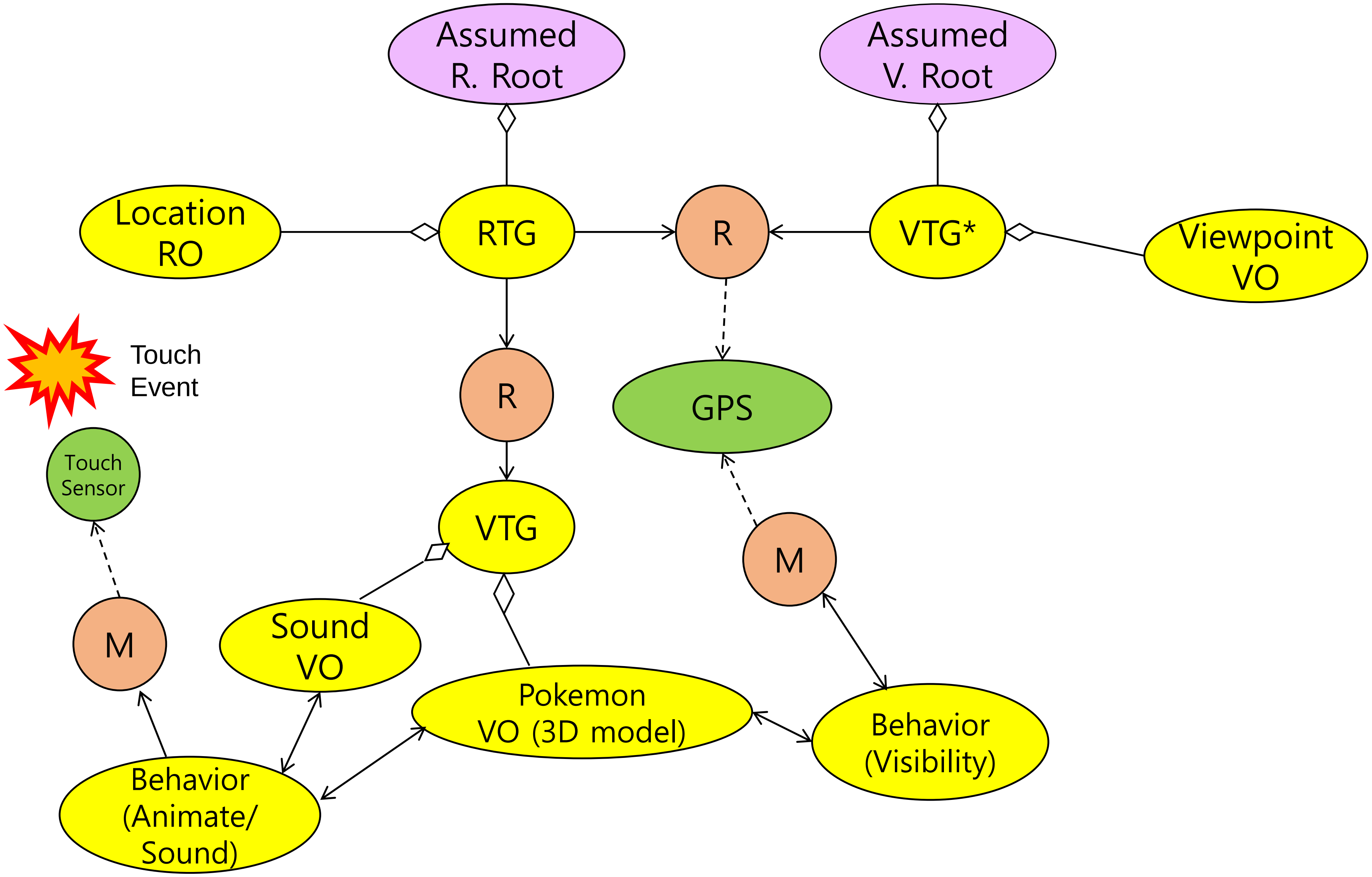




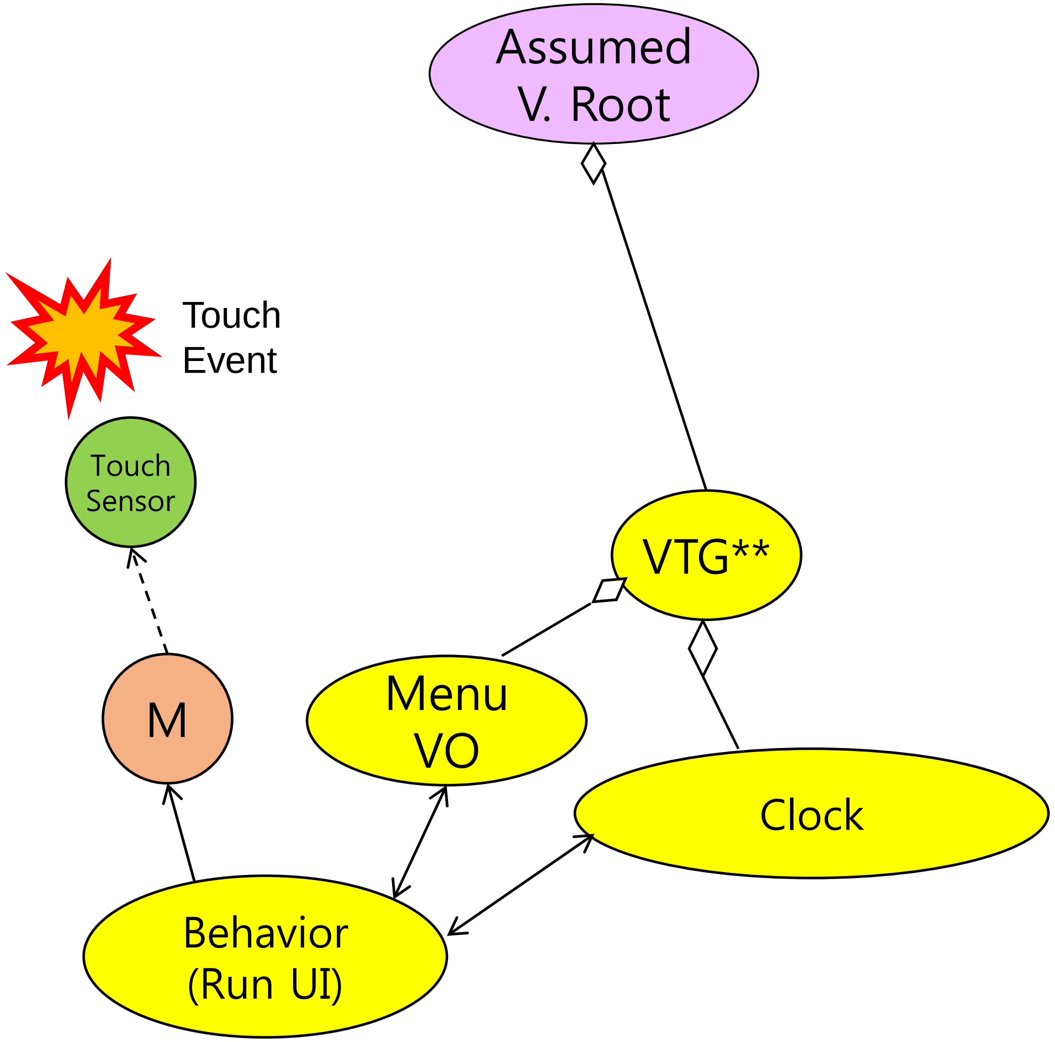
Scenario (Glass / Marker or Image patch based)



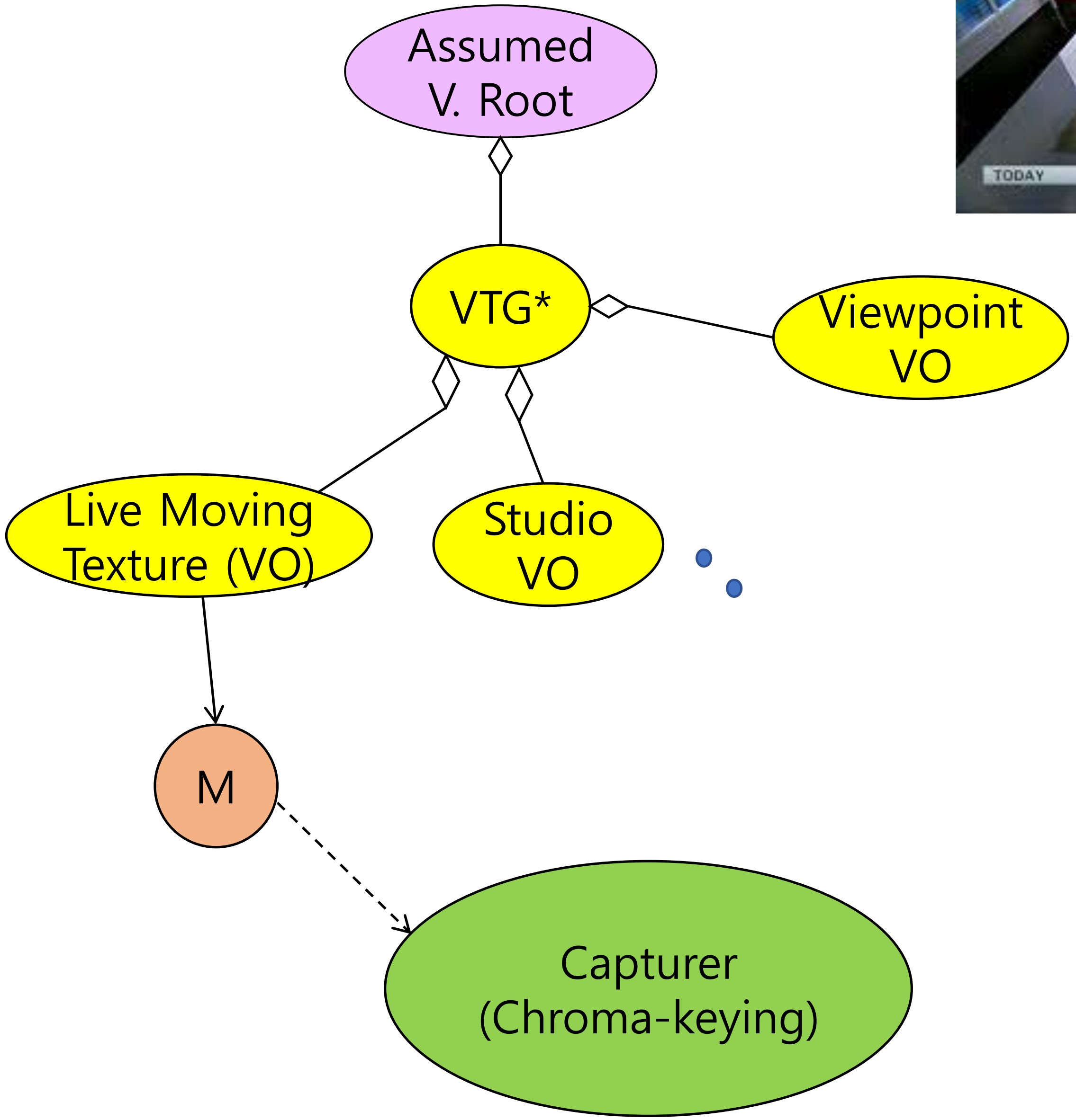
Scenario (Glass / Location based / Pokemon)



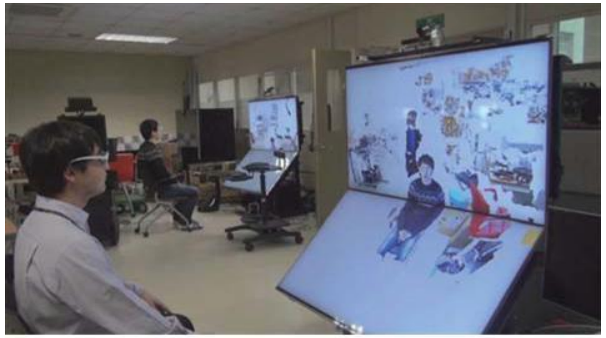
Scenario (Google Glass)

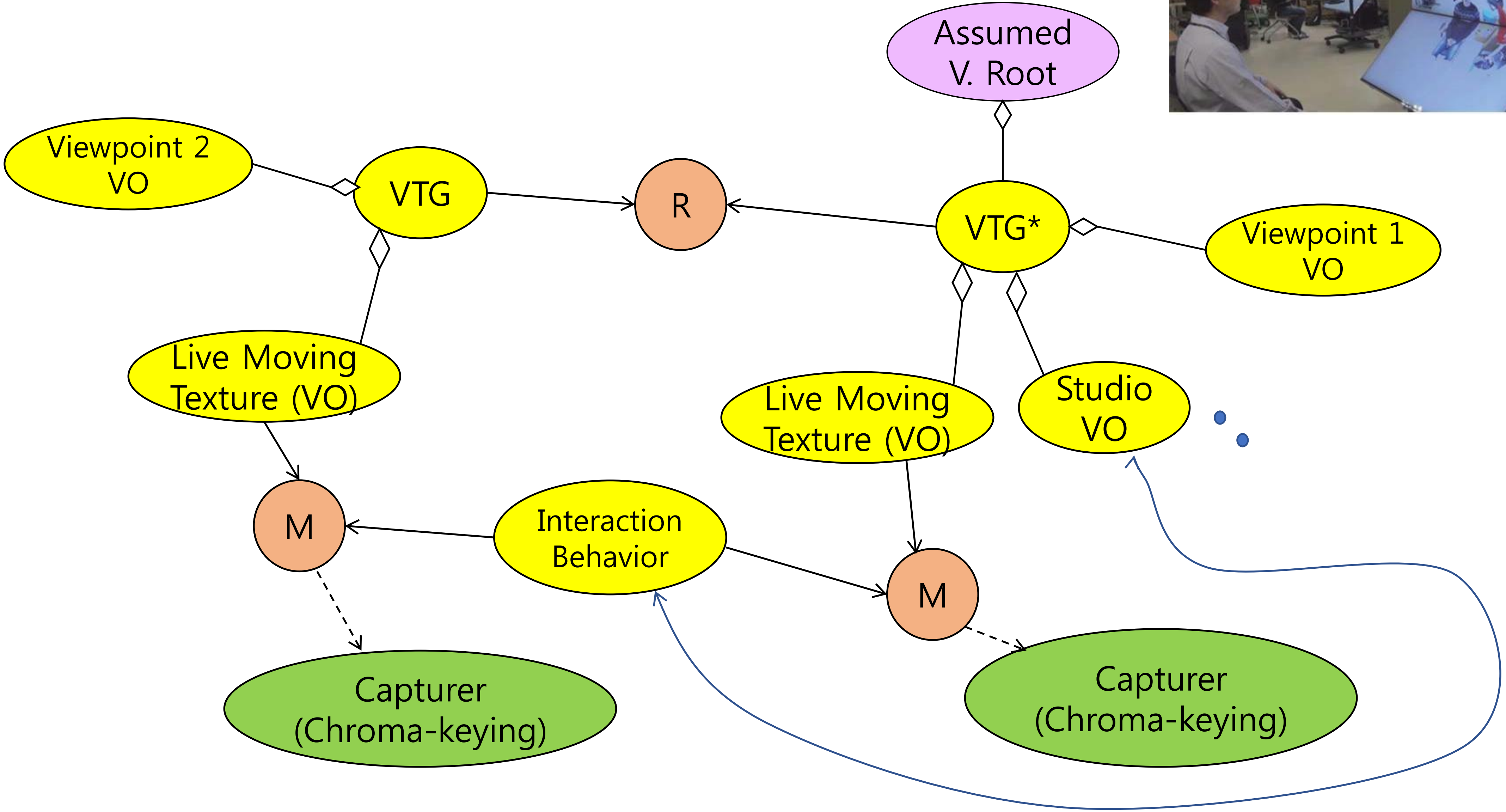


Scenario (Live actor in Augmented VR)



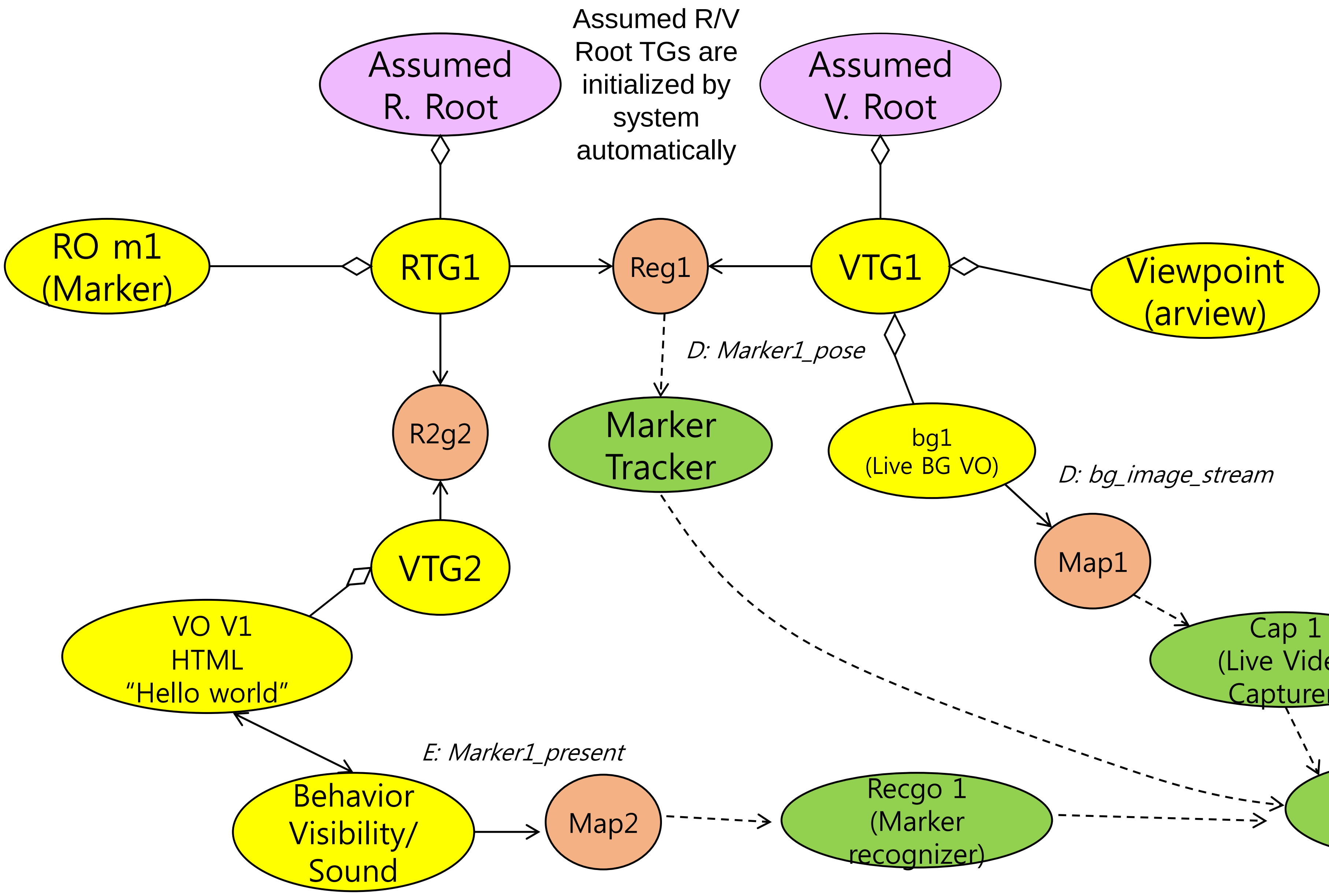
Scenario (Multiple interacting live actors)



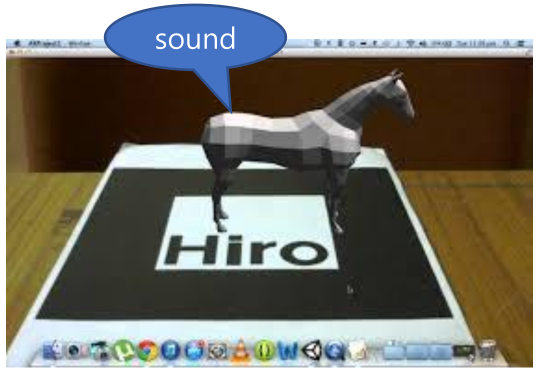


```
<capturer id = ‘cap1’ … >
<tracker id = ‘tracker1’ target = m1 … >
<recognizer id = ‘recog1’ target = m1 …>

<data id = ‘bg_image_stream’ type = video source= = ‘cap1’ …>
<event id = ‘marker1_present’ source = ‘recog1’ … >
<data id = ‘marker1_pose’ source = ‘tracker1’ … >

<scene id = ‘scene_1’ />
 <vtg id = ‘vtg1’ parent = ‘root’ … >
 <viewpoint id=‘arview’ parent = ‘vtg1’>
 <rtg1 id = ‘rtg1’ registration = ‘reg1’ …>
 <registration id = ‘reg1’ source = ‘tracker1’ child = ‘rtg1’ parent = ‘vtg1’ transform = ‘marker1_pose’ …>
 <vtg id = ‘vtg2’ registration = ‘reg2’ … >
 <registration id = ‘reg2’ child = ‘vtg2’ parent = ‘rtg1’ …>
</scene>

<background id = ‘bg1’ data_source = ‘map1’ data = ‘bg_image_stream’ parent = ‘vtg1’ …>
<robject id = ‘m1’ type = marker file = ‘hiro.dat’ parent = ‘rtg1’ …>
<vobject id = ‘v1’ type = HTML parent = ‘vtg2’
 content = ‘<h1 id = ‘aug1’ “Hello World” </h1>’
…>

<mapper id = ‘map1’ source = ‘cap1’ dest = [‘bg1’] …>
<mapper id = ‘map2’ source = ‘recog1’ out_event = ‘marker1_present’ dest = [‘beh1’ …] …>

<MARbehavior id = ‘beh1’ event = ‘marker1_present’ AND ‘marker1+pose’ object = [‘v1’]
 type = ‘show’ … >
```

Scenario (Marker or Image patch based, Video see-through)

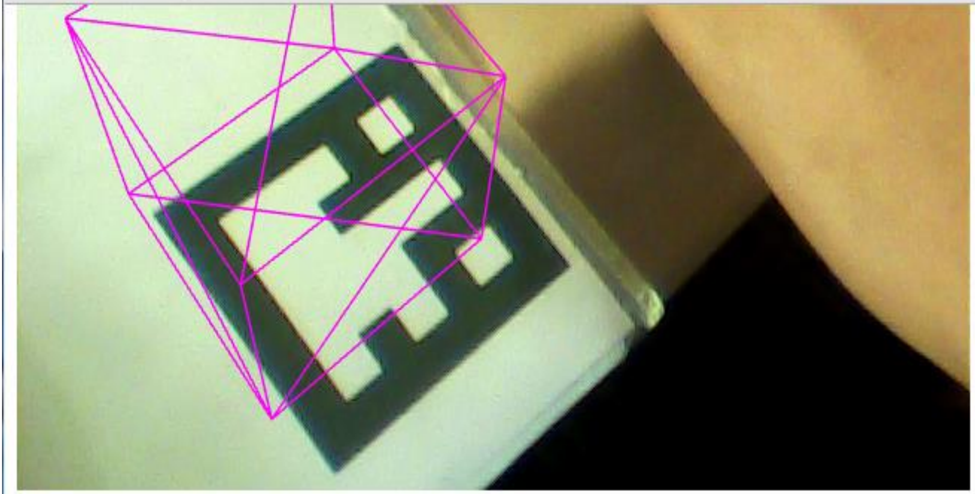


We consider only MAR content side and do not consider anything about system side



System side doesn't know anything about MAR content

Sensor just senses some real world data



```
<scene id="wld" type="ar"></scene>
<sensor id="cam_1" calibration="default"></sensor>
<background id="cam_based" sensor="cam_1" placeholer="default"></background>
<viewpoint id="arview" parent="default"></viewpoint>
<calibration id="cal_1"></calibration>

<Matrix id="forAobj1" x="0" y="0" z="-80"></Matrix>
<Matrix id="forAobj2" x="50" y="0" z="0"></Matrix>

<robject id="robj1" type="marker" marker-id="64" object="marker64" placeholder="#aobj1"></robject>
<robject id="robj2" type="marker" marker-id="88" object="marker88" placeholder="#aobj2"></robject>
<aobject id="aobj1" object="sphere" placeholder-for="#robj1" transform="#forAobj1" onclick ="default"></
<aobject id="aobj2" object="cube" placeholder-for="#robj2" transform="#forAobj2" onclick ="default"></ac
```

Conclusion: Component based extensions for MAR

- Individual constructs for different “modules” of info
  - Mix and match: realize a comprehensive set of MAR contents
  - Follow the MAR reference model (e.g. sensor, real capture, recognizer, sensor, tracker, …)
- Unified MAR Scene (which is virtual regardless of existence of real objects in it or not)
  - More content elements and logic more explicit and manageable
  - Decoupling into separate components (e.g. sensor, event, and recognizer)
  - Derive template for given system class
  - Minimize programming and explicit “routing”
  - Reuse existing constructs
    - Applicable to different formats as extensions: X3D, HTML 5, ARML, …
  - Initial UML-like based modeling

Conclusion: Component based extensions for MAR

- Future work
  - Complete specification and proofreading
  - More functions
    - Image based models
    - Haptics and other multimodality
    - Live actors and behaviors (c.f. K. Yoo)
    - Meta information (c.f. W. Woo)
    - Perceptual elements (e.g. brightness against dynamic environment conditions)
  - More use cases and application file formats
    - SLAM based
    - Spatial/Projection AR
    - Multi-user: Tele-presence, SNS, …
  - Continued validation by implementation
  - CD by August, 2018 / DIS by December 2018