Literature Review 1

Computer Graphics I

Zubin Bhuyan (Student id: 01744486)

Paper 1: Martin Fischbach, Dennis Wiebusch, and Marc Erich Latoschik. 2017. **Semantic Entity-Component State Management Techniques to Enhance Software Quality for Multimodal VR-Systems**. *IEEE Transactions on Visualization and Computer Graphics* 23 (4), (April 2017), 1342-1351. DOI: https://doi.org/10.1109/TVCG.2017.2657098

Paper 2: Eckstein, B., Lugrin, J.-L., Wiebusch, D. & Latoschik, M. (2016). **PEARS - Physics Extension And Representation through Semantics**. *IEEE Transactions on Computational Intelligence and AI in Games* 8 (2), 178-189. doi: 10.1109/TCIAIG.2015.2505404

Virtual Reality (VR) and Augmented Reality (AR) applications with multi-modal interfaces (MMI) are becoming common in areas such as gaming and robotics. Most of these systems (or "apps") fall in the category of Real-Time Interactive Systems (RIS). From a software architecture perspective, they are usually comprised of several subsystems, dedicated for tasks pertaining to input analysis, graphics/rendering, AI, etc. For consistency and low-latency in processing, these subsystems have specific data representations and interdependencies which results in close coupling. Here arises a contradiction, known as the coupling dilemma, because having de-couped system enables modularity, reusebility, modifierbility and other such desirable non-functional software qualities.

The techniques presented in [Paper 1] are use to further extend the usability of the Entity-Component-System (ECS) pattern. The ECS architectural pattern offers flexibility in describing scenes where all objects (in the scene) are entities, such as *chair, table*, etc. Every e*ntity* consists of one or more *components*. Components are behavior or functionality; adding (or removing) components modify of the behavior of an entity at runtime.

Even though the ECS approach greatly facilitates de-coupling, it runs into problem with inter-system communication and "mutual access to components outside of their primary data associations". Decoupling requirements and system state management of MMI-RIS combination provides motivation for this work. Fischbach, et.al., in [Paper 1], puts forth five semantics-based software techniques which attempts to remedy this dilemma. The authors claim that these techniques allow state representation and access in the ECS pattern, improves modularity in design and allows provision for improved maintainability. An example of MMI interior design application is taken to explain the concepts proposed in the paper.

The five techniques are:

a. Semantic grounding: Predefined tokens, called *grounded symbols*, are used instead of direct variable access or calling mutant methods. Defining these tokens externally allows their use in more than one application. Application states can be described using grounded symbols, semantic types, semantic values, and entities.

b. Code from Semantics: The second technique make use of an external ontology and a code generation method. This enables lookup and utilisation of external concept and associated roles. The use of native programming language result in faster knowledge representation resolutions (faster ontology access at runtime). The overhead incurred is that of time needed for code generation during compilation.

c. Grounded actions: After successful analysis of user's commands (as gestures and speech), the application reacts with operations which are perceivable by the user. The third technique involves semantic description of reusable system operations. These are called *grounded actions* and are implemented as implemented as functions in the code. Semantic actions comprise of a set of preconditions, a set of parameters and a set of effects. This enables the ECS pattern to allow for reusable operations of arbitrary complexity.

d. Semantic query: This allows accessing entities and actions by semantic descriptions. Semantic types and values are used to define entity selection filters. This also allows grounded action lookup. Further, subsystems can specify data depositories without having to explicitly reference individual entities. Semantic queries enable lookup based on predicates along with symbolic data.

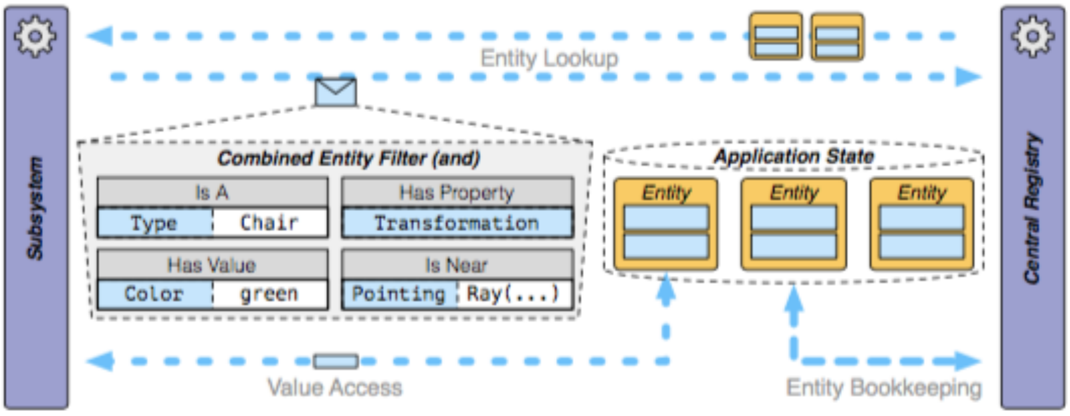
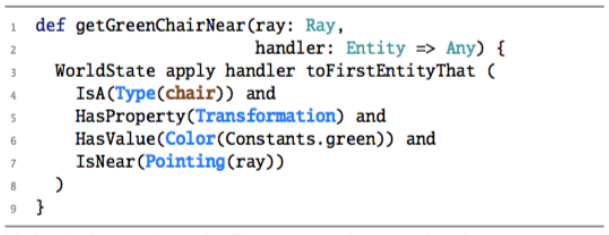


Fig1- Conceptual overview of an entity request by a subsystem using a logical combined entity filter.

The paper makes use of an example based on MMI app for interior modelling. The user provides as input the words an gestures: “Put [gesture] that green chair near [gesture] this table.”, followed by, “Turn it [kinematic gesture] this way.” Fig. 1 illustrates the technique for selecting the green chair from our example.



```
def getGreenChairNear(ray: Ray,
                      handler: Entity => Any) {
  WorldState apply handler toFirstEntityThat (
    IsA(Type(chair)) and
    HasProperty(Transformation) and
    HasValue(Color(Constants.green)) and
    IsNear(Pointing(ray))
  )
}
```

Fig 2- A filter to query the central registry, constructed by passing semantic values and types.

e. Decoupling by semantics: This applies to all the above mentioned techniques. All data sources/sinks and operations are described using grounded actions and semantic queries.

Multimodal Input Processing and integration with other subsystems:

MMI has to be processed on multiple aspects and several levels in order to achieve the desired output from the user utterance/inputs. Additional information, like the direction being pointed, must be obtained by analysing users posture and joint locations. This might further eases the design process of integrating multiple subsystems.

As already mentioned, subsystems can be integrated at low-level, requiring access to specific application data and states; or it can be loosely coupled. An example of the latter, is discussed in **[Paper 2]**. [Paper 2] couples commercials game engine and an AI framework. AI-based physics engines provide a finer grained user interaction as each process is defined as subprocesses which are able to make continuous transitions for discrete state changes. And since the AI and physics modules are decoupled, they allow greater applicability and reusability. This however, comes with an overhead of potentially expensive computation which hamper the real-time performance, especially for scenes which have a large number of objects.

PEARS: Physics and AI module integration

The framework uses a semantic representation of physical properties and a reasoning engine to simulate cause and effect between objects. Semantic rules are used along with classical collision detection to model physical processes. Additionally, an ontology is used to describe the domain being modelled. This ontology describes objects, their types and properties and relations between other objects and the environment.

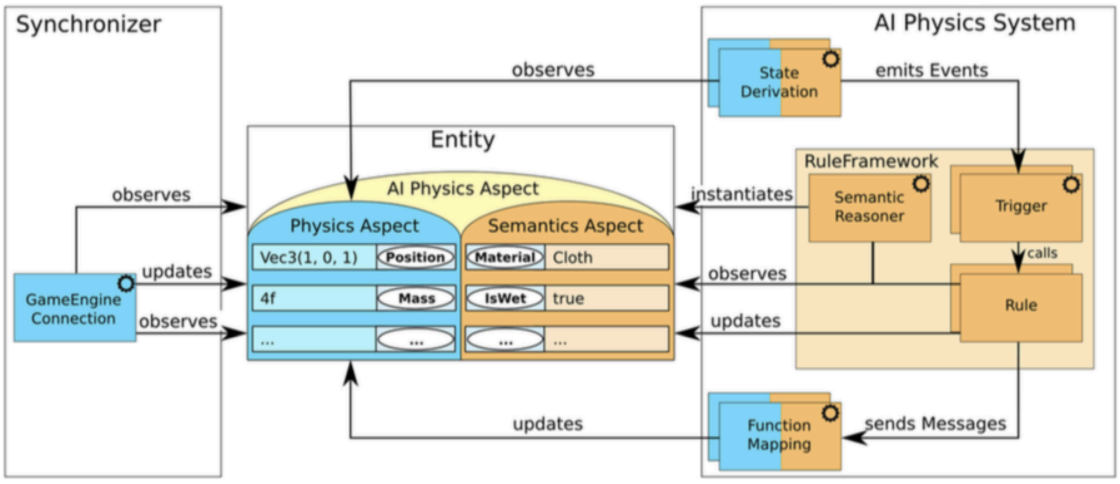


Fig 3- AI-Physics system

They try to achieve decoupling of the modules by having an entity model. This model has a centralized representation of the *simulation state* and the individual entities have *state variables*. Their primary example is the simulation of the Rube Goldberg machine.

Semantics ECS for PEARS with AR/VR

The AI-Physics system of [Paper 2] allows the simulation of a number of physical processes. It is somewhat scaleable with regards to realtime performance with concurrent processes. It also achieves several of the non-functional SE goals which are mentioned in [Paper 1].

However, when we want simulation of a larger number of concurrent processes, or an additional module(s) for *Augmented/Virtual Reality experience*, we must consider design and implementation using the extended-ECS pattern. Additional modules, along with an Upper Ontology, might bring into question the reusability, modularity, maintainability and even portability of the implementation. Since OWL or one of its variations is already used for the ontology in the design and development phases, it will be relatively easy to incorporate SPARQL query support.