The Journal of Supercomputing
https://doi.org/10.1007/s11227-021-04212-6

Knowledge-based virtual outdoor weather event simulator using unity 3D

Hamza Noueihed[1] · Heba Harb[1] · Joe Tekli[1]

Accepted: 15 November 2021

Abstract

During the past two decades, 3D simulation models have gained importance in the development of software solutions that aim to mimic real-world events and phenomena with increasing levels of accuracy and detail. In this context, knowledge representation and processing have recently shown a significant contribution to the simulation modeling domain, where knowledge graphs have been used in different fields to build knowledge representations for multiple purposes. In this paper, we introduce VOWES, a Virtual Outdoor Weather Event Simulator to replicate and measure outdoor weather events in vivid 3D visualizations. We design and implement an integrated knowledge graph (KG) representation for VOWES, by creating two constituent KGs: (i) Weather KG describing weather data and events, and (ii) Simulator KG describing 3D simulation components and properties, and connecting them with the (iii) Semantic Sensor Network (SSN) KG to form an integrated structure serving as the knowledge backbone of the VOWES simulation environment. We make use of the Unity 3D engine to build and design the simulator environment and its virtual sensors, and integrate the Mapbox SDK and the WeatherStack API for realistic real-world weather mapping. We have conducted qualitative evaluations involving 13 expert and 30 non-expert testers, to assess the quality of VOWES' KGs and its simulation environment. Results show that more than 80% of the testers gave a combined quality score ≥ 3 out of 4 on most evaluation criteria. We have also conducted performance evaluations to test VOWES loading, execution, and data search time, among other features. Results show that most operations require almost instantaneous or linear time, where search, refresh, and export operations share almost identical performance levels, with execution time increasing by approximately 179 μs for every added game object. This highlights the simulation tool's time performance in running large simulation projects, and its ability to simulate complex weather environments with large numbers of sensors and weather phenomena.

The author is also an adjunct researcher and member of the SPIDER research team, LIUPPA Laboratory, University of Pay and Pays Adour (UPPA), 64,600, Anglet, France.

Extended author information available on the last page of the article

Published online: 24 January 2022

Keywords Simulator modeling · Knowledge graphs · Unity 3D · Virtual sensors · Semantic sensor network · Weather events

1 Introduction

With the rising interest in creating realistic and vivid simulations, 3D models have been gaining increasing importance in the development of software solutions that aim to mimic real-world events and phenomena. Simulation modeling allows creating and analyzing the behavior of a digital prototype system representing a physical real-world entity, aiming to study and predict the latter's behavior and performance in the real-world [17]. Hence, simulation software has become one of the most commonly used techniques for virtual demonstrations in different fields, especially 3D models used to simulate real-world structures, objects, and events, with increasing levels of accuracy and detail, e.g., [17, 25, 61]. In this context, knowledge representation and processing have recently shown a significant contribution to the simulation modeling domain, where knowledge graphs have been used in different fields to build knowledge representations for multiple purposes, e.g., [3, 14, 46]. A knowledge graph (KG) is a formal description of knowledge presented as a set of concepts within a domain, and the relationships that connect the concepts [50]. KG-based knowledge representation and reasoning techniques help in providing semantic information about the environment and enable knowledge sharing, processing, reuse, capture, and communication, e.g., [9, 20, 49].

In this paper, we introduce VOWES, a Virtual Outdoor Weather Event Simulator to represent outdoor weather events and data in vivid 3D visualizations. It is designed as a digital twin solution to describe and replicate weather measurements, events, sensors, and their properties from the real-world, into a software simulation environment. It provides the knowledge graphs and visualization components needed for that purpose. First, in terms of knowledge representation, we design and implement an integrated knowledge graph (KG) by creating two constituent KGs: (i) Weather KG describing weather measurements (e.g., wind, temperature, humidity) and weather events (e.g. storm, fire, and tornado), by extending an existing representation from [10]; and (ii) Simulator KG describing the simulator's components and properties. We also utilize an adaptation of the Semantic Sensor Network (SSN) KG [53] to represent virtual and physical weather sensors and connect all three KGs to form an integrated structure serving as the knowledge backbone of the simulator. Second, in terms of 3D visualization, we make use of the Unity 3D engine to build and design the simulator environment and its virtual sensors. We develop special visualizations and behaviors to present weather measurements, events, and sensors as visible 3D structures with specifications controllable by the user. In addition, we utilize the Mapbox SDK [31] to import high-resolution world maps showing countries, cities, and buildings. We also utilize the WeatherStack API [58] to capture real-time weather measurements and conditions from the geographic area that is being simulated and integrate them in the simulation environment to allow for more realistic and accurate simulations. VOWES provides a foundation layer, including the needed knowledge representation components to represent and describe weather

measurements and events. This is a requirement and serves as a preparation step to perform event prediction and weather forecasting functionalities in the future.

We have implemented VOWES as a user-friendly desktop application, allowing users to run multiple simulations at one or multiple geographic locations simultaneously. Users can provide their input measurements, events, and virtual sensors to initialize the simulation exercise, and can interact with the system without requiring any previous knowledge about Unity 3D or other software simulators. We have conducted qualitative evaluations involving 13 expert and 30 non-expert testers, to assess the quality of VOWES' KGs and its simulation environment. Results show that more than 80% of the testers gave a combined quality score ≥ 3 out of 4 on most evaluation criteria. We have also conducted performance evaluations to test VOWES loading, execution, and data search time, among other features. Results show that most operations require almost instantaneous or linear time, where search, refresh, and export operations share almost identical performance levels, with execution time increasing by approximately 179 µs for every added game object. This highlights the simulation tool's time performance in running large simulation projects, and its ability to simulate complex weather environments with large numbers of sensors and weather phenomena.

2 Background and related works

In this section, we briefly review the background and existing works covering: (i) KGs related to our study, and (ii) simulation environments based on Unity 3D.

2.1 Knowledge graphs

A Knowledge Graph (KG) is a structured knowledge base that uses a graph-based data model to describe entities (e.g., objects, events, situations) in a given domain [40]. A KG consists of nodes and arcs, where nodes represent semantic concepts underlining one or multiple entities designating the same meaning, and arcs underline the semantic links connecting the concepts, representing semantic relationships (e.g., *synonymy*, *hyponymy* (*IsA*), *meronymy* (*PartOf*), etc. [33, 40]). KGs have been used in different domains to describe different kinds of information, ranging over lexical concepts (e.g., Yago [22]), webpages (e.g., ODP [26]), images (IKG [59]), social Web events (e.g., SEDDaL [3]), and sentiment analysis (e.g., LISA [15]). In the following, we briefly describe two KGs used in our study: (i) the SSN KG [53] describing sensor networks, and (ii) the Blizzard KG [10] describing weather properties.

2.1.1 Sensor network KGs

Several sensor network KGs have been developed in the literature. One of the earliest approaches in [18] provides a set of KGs describing missions, tasks, sensors, and deployment platforms, and utilizes semantic reasoning to recommend collections

of sensors and platforms that are adapted for particular tasks. In [11], the authors introduce a KG to represent images for object recognition in remote sensing image interpretation. The scope of the work is limited to images and does not extend to other types of multimedia data (e.g., video, audio). A similar KG is described in [32] where the work is also limited to image representation for object-based image retrieval. In [38], the authors introduce a KG to describe and monitor noise pollution in urban zones by sensing audio noise levels using the occupants' mobile phones. The proposed KG only describes noise data and geo-locations to generate noise-level maps. While expressive in their application areas and domains, most aforementioned KGs do not fully consider the sensor, platform, or data diversity required in designing and describing generic sensor networks [28].

As a result, most recent studies rely on the Semantic Sensor Network (SSN) KG [53] produced by the W3C SSN Incubator group, as a generic and expressive KG often utilized as the starting point for the generation of new KGs describing different kinds of sensor networks, e.g., [24, 41, 45]. It is defined based on the Ontology Design Pattern model [53] which describes the relationships between sensors, observations, features, and stimuli using four main perspectives: (i) *sensor perspective*, focusing on what and how the sensor senses, (ii) *observation perspective*, focusing on observation data and their related metadata, (iii) *feature and property perspective*, focusing on observations made w.r.t.[1] a particular feature or property, and (iv) s*ystem perspective*, focusing on the sensor network and its deployment. Moreover, SSN follows a horizontal and vertical modularization architecture by including a lightweight and self-contained core KG for its elementary concepts and properties called SOSA (Sensor, Observation, Sample, and Actuator) [54]. SSN and SOSA have been used in a wide range of applications, including scientific data monitoring, observation-driven ontology engineering, and the Web of Things, e.g., [37, 41, 42]. In this study, we utilize an extract from the integrated SSN/SOSA KG [54] shown in Fig. 1 to handle the sensors used in our simulator.

For instance, the authors in [45] propose a standard KG for IEEE[2]'s Ontologies for Robotics Automation (ORA) working group, by extending SSN with a refined set of requirements including new concepts like *domain*, *state*, *intention*, and *physical location* to cover the knowledge representation and reasoning needed in robotics and automation. In [24], the authors design and implement an e-Health system based on SSN, including a formal specification to express e-Health domain knowledge using e-Health terminologies and their relationships. The proposed KG supports different styles of data formats received from patients' devices to the e-Health system and specifically addresses device interoperability issues. In [6], the authors address Wireless Sensor Networks (WSN) and focus on features that describe sensor nodes, their functionality, and their current processing, memory, and power supply states, to determine the future state of the WSN. The authors in [5] propose an extension of SSN, denoted MSSN (Multimedia SSN), describing the properties of *multimedia data* (e.g., *video*, *audio*, and *frequencies*). They improve sensor diversity by adding

[1] With respect to.

[2] Institute of Electrical and Electronics Engineers.

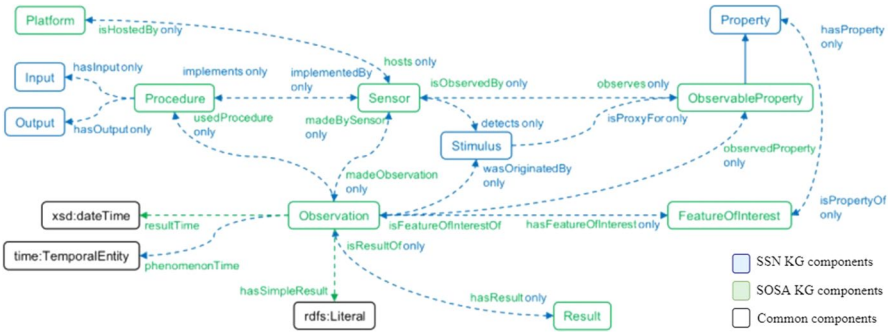


Fig. 1 Extract of the integrated SSN/SOSA KG [54]. SSN/SOSA's main concepts include: *feature of interest*—the thing whose property is being estimated, calculated, or sampled; *observable property*—an observable quality (property, characteristic) of a *feature of interest*; *procedure*—a workflow, protocol, plan, algorithm, or computational method specifying how to make an observation or create a sample, or make a change to the state of the world (via an actuator); *sensor* – a device, agent, or software (simulation) involved in, or implementing, a *procedure*; and *stimulus*—an event in the real world that triggers the *sensor*. The main concept relationships include: *detects*—from a *sensor* to the *stimulus* that is detected; *is observed by*—between an *observable property* and the *sensor* able to observe it; and *has property*—between an entity and a property of that entity, among others. Note that the properties associated with the *stimulus* may be different from the *observable property*, since it is the event, and not the object, that triggers the *sensor*

a so-called *media sensor* that observes multimedia properties. In [28], the authors propose an extension of SSN to describe hybrid sensors, denoted Hybrid SSN (or HSSN), by adding concepts describing *mobile* and *multimedia sensors*, *multimedia data*, and hybrid *platforms*. They focus on describing different sensor types (e.g. *mobile* and *static* sensors), different platforms (e.g., *environments*, *systems*, and *devices*), and different data (e.g., *scalar*, *textual*, and *multimedia*), and propose a generic model to facilitate KG re-usability in various application domains. In [41], the authors present an RDF[3] dataset on meteorological measurements described using an extension of SSN/SOSA, by adding concepts describing *units of measurement* as well as *geographical places* and their *locations*.

To our knowledge, our present study provides the first attempt at extending SSN/SOSA to describe weather properties and events in the context of a virtual outdoor weather simulation environment.

2.1.2 Weather KGs

Various KGs have been developed in the domain of earth sciences to describe various aspects of the earth's environment and weather, e.g., [8, 39, 60]. Most of them describe physical properties and geographical features (e.g., waterbody, forest, atmosphere), and only include basic relationships to represent geographic or weather events. In most cases, the weather events are not explicitly related to their sensing

[3] Resource Description Framework [27].

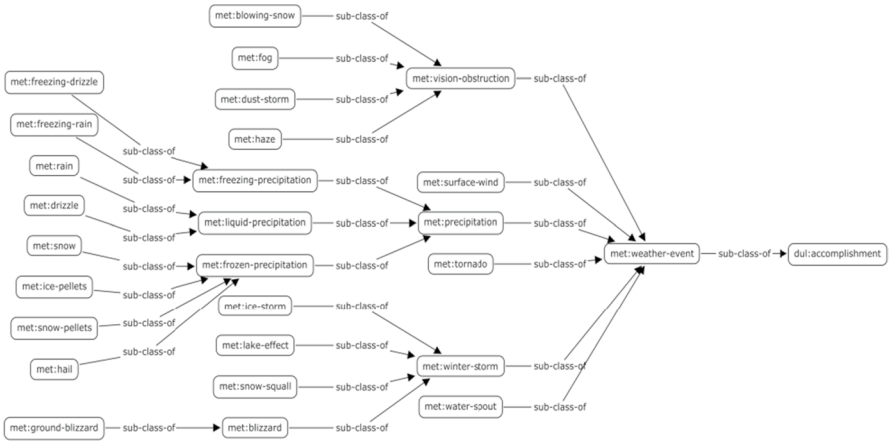


Fig. 2 Extract of the Blizzard KG [10]. The Blizzard KG classifies winter weather events into three main categories: i) precipitation, ii) obscurations, and iii) other surface weather events. *Precipitation* refers to the deposition of any form of water particles (e.g., liquid or solid) from the atmosphere to the ground; *vision-obstruction* specifies any weather occurrence in the atmosphere; and other weather events cover *surface wind*, *tornado*, *waterspout*, *winter storm* among others. The Blizzard KG also allows relationships between basic and more complex events. For instance, a *winter storm* event is marked by a combination of *freezing precipitation* and *strong wind*. Other types of *winter storms* included *ice-storm*, *lake-effect*, *snow squall* and *hail storm*. The *sub class of* relationship (also known as the *inheritance* relationship) is used to connect basic (more specific) and complex (more generic) events together

concepts (e.g., observed property, observation, event, and results), which is essential to infer information about weather events from sensor observations. Moreover, many interesting queries over sensor observations concern weather events rather than weather properties [47]. In other words, queries expressed solely in terms of weather properties are not sufficient to reason about weather events [4]. In this study, we adopt the Blizzard KG from [10] and extend it to cover the concepts and relationships needed in our simulation environment. An extract of the KG is shown in Fig. 2. It includes basic winter weather events and describes their relationships with the weather properties observed by the measuring sensors.

In this paper, we extend the Blizzard KG to consider additional weather concepts describing weather events and measurements computed by real and virtual sensors, following the design requirements of our simulation environment. The extended Weather KG is then connected with SSN/SOSA (cf. Sect. 2.1.1) and our Unity-based Simulator KG (cf. Sect. 3.2), to form the integrated VOWES KG (cf. Sect. 3.3).

2.2 Unity 3D engine

With the rising interest in creating realistic and vivid models, Unity 3D has been gaining increasing importance as a powerful tool for the creation of 3D visualizations, functions, and attributes, and their integration with dedicated processing features and metric measurements to achieve accurate outputs and analyses. Unity is a cross-platform game engine developed by Unity Technologies, which was announced and released in June 2005 at Apple Inc.'s Worldwide Developers

Conference as a Mac OS X-exclusive game engine. Starting in 2018, the engine has been extended to support more than 25 platforms for creating two-dimensional (2D), three-dimensional (3D), augmented reality, and virtual reality games. In addition, the Unity engine has been used for simulations in various fields including architecture, engineering, automotive, and construction, e.g., [44, 48, 55]. One of its most distinctive features is the power of its real-time 3D rendering, making it one of the world's leading real-time development platforms [51]. A Unity 3D game or simulation project starts by building the scene where the visual assets are placed. It follows the Cartesian (x, y, and z) coordinate space, allowing to easily move around a scene interactively, either in a first-person or third-person perspective. The user can also switch between several views, including isometric (2D) and perspective (3D), allowing the scenes to update in real-time when the user is previewing the project. Project assets include graphical objects, sounds, scripts, and prefabs (pre-assembled game objects), and can be organized in a hierarchy where children objects can be associated as subordinates to their parent objects, following their parents' behaviors (they react to their parents' actions, movements, and sounds, etc.). Fully configured game objects can be easily saved for reuse and sharing between scenes or projects without having to be redesigned or reconfigured.

In this context, several Unity 3D-based simulation solutions have been developed in the literature. In [57], the authors design a Unity-3D simulator to help navigate unmanned aerial vehicles (UAVs). The latter are coupled with sensors and physical hardware allowing to collect data from the UAV's surrounding environment and feed it into the virtual simulation for processing and analysis. The authors emphasize the importance of Unity 3D in presenting a realistic and precise model while tracing the performance of UAVs in the real world. In [1], the author designs Virtual Intelligent Sensors (VIS) to mirror the operations of Physical Intelligent Sensors (PIS) which are deployed to measure rocket engine properties and trace the accuracy and performance of the VIS in comparison with the PIS. The study highlights the significance of simulating rocket engines to imitate and study real life events, variables, visualizations, and features in an easily controlled and completely monitored 3D environment. The author also emphasizes the practicality of Unity 3D in integrating virtual sensors and related objects and behaviors in different types of simulation exercises to measure different discrete parameters and variables. In [7], the authors develop a Unity 3D virtual environment to study the properties and potential prospects of using solar energy on buildings in a highly populated urban area. They mimic building structured using dedicated 3D visualizations, and mimic solar energy measurements based on values and calculations accumulated from a real-world urban area in the city of Istanbul. The authors specifically address the challenge of attaining high accuracy in predicting solar energy outcomes with the influence of the buildings' shadow casting. The authors extend their simulated environment to represent and study the underground utility systems in the city, where the whole city map is translated into a dedicated underground 3D model. In [23], the authors design and integrate virtual sensors to measure light conditions in both indoor and outdoor environments. They focus on monitoring daylight conditions and design artificial light sources to map different sunlight conditions in the real world. They accumulate measurements from real sensors and map the data to the virtual

sensors to create realistic conditions in the virtual environment. The authors utilize a dedicated CAD software to create both indoor and outdoor environments with high degrees of precision and accuracy. Unity 3D is used to animate the CAD environment and handle light condition variations and sensor simulations. In [57], the authors show how environmental events, such as fire, can be demonstrated in a 3D manner. They attempt to imitate real-life scenarios, and study the level of stress that different people might face in simulating different kinds of fire events. The authors highlight the capabilities of Unity 3D in visualizing and animating complex objects and events such as *fire*, *flame*, *smoke*, and their propagation.

Different from most existing solutions which are application specific, our present study combines Unity 3D with a KG-based data model, producing a generic and extensible 3D environment for simulating and studying weather conditions and event. The use of KGs to represent and describe the environment, the measurements, and the sensors, ensures flexibility and extensibility, and allows to easily add new environmental properties, new measurements, and new sensors, thus seamlessly adapting to new situations and simulation scenarios.

3 VOWES knowledge graph

We design an integrated KG representation by creating two constituent KGs: (i) Weather KG describing weather measurements and events, and (ii) Simulator KG describing the simulator tool's components and properties. We utilize an adaptation of the SSN/SOSA KG (cf. Sect. 2.1.1) to represent virtual and physical weather sensors and connect all three KGs together to form our integrated VOWES KG structure.

3.1 Weather knowledge graph

We first introduce Weather KG to represent the main weather concepts and relationships required in our solution. We utilize Blizzard KG [10] as a starting point (cf. Sect. 2.1.2) to describe *weather events*, and extend it to include the needed concepts describing *weather measurements* and *weather states*, and their observed properties.

3.1.1 Weather events sub-graph

Weather events represent simple event concepts (e.g., *rain*, *drizzle*, *hail*) or complex event concepts (e.g. *winter storm*, *precipitation*) made of simple or complex sub-events. Our *weather events* sub-graph is shown in Fig. 3. It mainly consists of the Blizzard KG from [10] (cf. Sect.2.1.2), while highlighting the possibility of adding new concepts to extend the descriptiveness of the KG. For instance, we add *heat-wave*, *fire*, and *smoke* concepts as sub-classes of *weather events* in order to represent them in our simulator. These are essential types of weather incidents that have become specifically relevant to the current threat of global warming. This adds to the pre-defined list of weather event concepts marking a total of 22 events that are

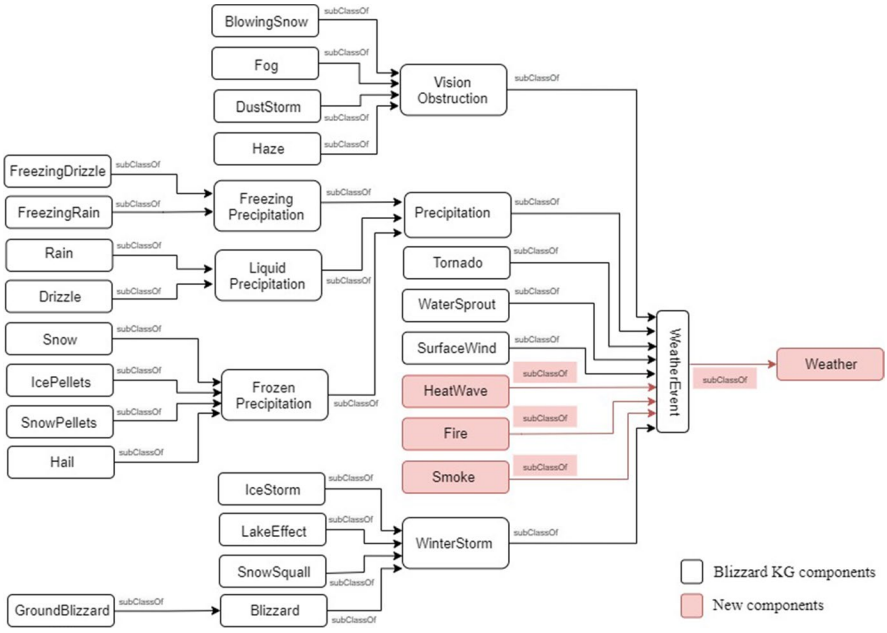


Fig. 3 *Weather events* sub-graph

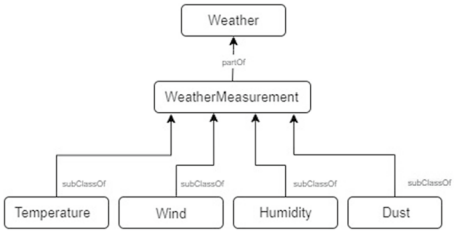


Fig. 4 *Weather measurements* sub-graph

currently represented in our KG. Other similar event concepts can be easily added with their relationships, based on the user's needs in simulating different weather scenarios.

3.1.2 Weather measurements sub-graph

Weather measurements represent measurable properties whose values are defined based on the occurring weather events. In other words, they represent the values produced by a weather sensor to measure the properties of weather events. Our *weather measurements* sub-graph is shown in Fig. 4. It consists of four main weather measurement concepts considered in our present simulator: *temperature*, *wind speed*, *humidity*, and *dust*. Other weather measurements can be easily added following the user's needs.

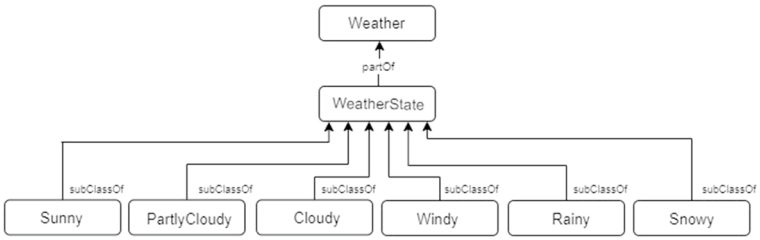


Fig. 5 *Weather states* sub-graph

Note that while weather events occur over large geographic areas (e.g., a whole city), yet weather measurements represent punctual data values collected from multiple sensing points in multiple different locations within the geographic area, where the locations are designated by the corresponding weather sensors. For example, if a *snow* event is taking place in a certain geographic area, *temperature*, *humidity*, *wind*, and *dust* levels will be sensed by one or multiple sensors placed in one or multiple geographic locations, allowing to trace of the snow's fluctuating parameters over the concerned area. These measurements are individually collected at the sensors' specific locations and are then aggregated in the data repository to allow weather event analysis. This provides a more realistic representation of weather measurements, allowing to describe both real and virtual weather environments using the same KG structures.

3.1.3 Weather states sub-graph

Weather states represent the atmospheric conditions describing the weather within a geographic area, revealing general expectations regarding weather events and measurements. Our *weather states* sub-graph is shown in Fig. 5. It consists of 6 basic states, and is extensible to others following the user's needs: (i) *sunny*: clear sky marked by brilliant sunlight, (ii) *partly cloudy*: cloud transitions covering part of the sky, (iii) *cloudy*: sky characterized by clouds, (iv) *windy*: strong winds exposition, (v) *rainy*: characterized by considerable rainfall, and (vi) *snowy*: characterized by snowfall.

3.1.4 Integrated weather knowledge graph

Our integrated Weather KG is shown in Fig. 6. It connects the *weather events*, *weather measurements*, and *weather states* sub-graphs mentioned above, through the generic concept of *weather*, which also serves as the root of the KG. We represent *weather events* as a special kind of *weather* concept and connect them using the *sub-class of* inheritance relationship. We represent *weather measurements* and *weather states* as components of a *weather* instance, and connect them using the *part of* composition relationship. The latter is because weather measurements and weather states do not represent different types of weather, but are rather parts of a weather's description.

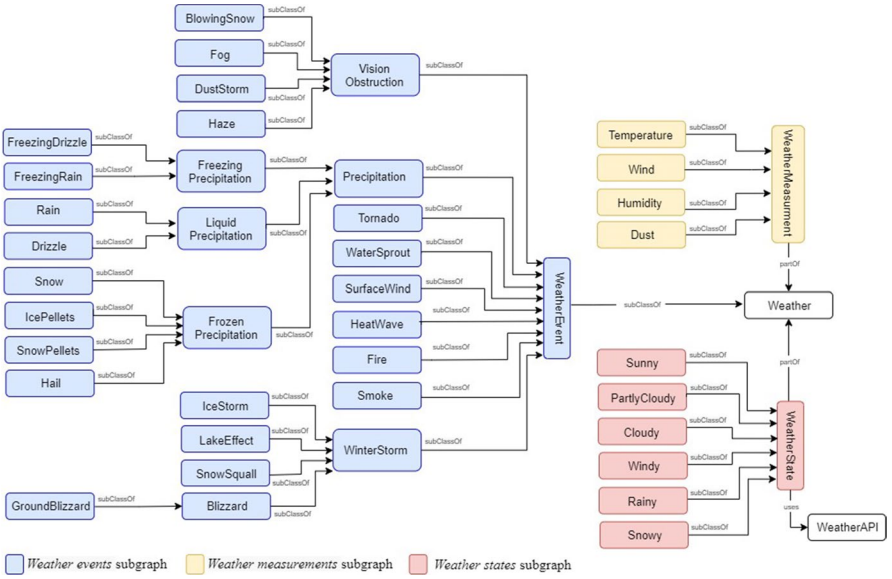


Fig. 6 Weather KG

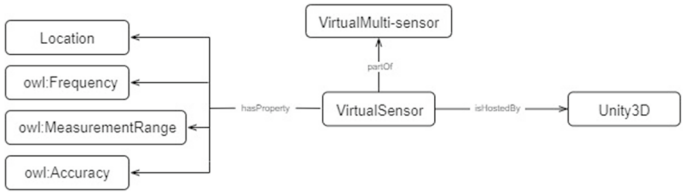


Fig. 7 *Virtual sensor* sub-graph

Note that our simulator allows collecting real weather states and mapping them into the simulation environment, to re-create and monitor real-time weather conditions (e.g., the simulation of New York city would show and trace the same weather states present in the real-world city of New York). This is represented using the *weather API* concept, connected with the *weather state* concept through the *uses* cross relationship. Section 4 provides a technical description of the weather API and its measured properties.

3.2 Simulator knowledge graph

We also develop a comprehensive KG to describe the simulator's components, including: (i) virtual sensors, (ii) virtual weather measurements, and (iii) virtual sensing procedure.

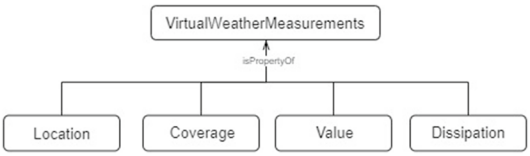


Fig. 8 *Virtual weather measurements* sub-graph

3.2.1 Virtual sensor sub-graph

Our *virtual sensor* sub-graph is shown in Fig. 7. It highlights two types of virtual sensors: (i) an individual sensor, labeled *virtual sensor*, designed to capture one single weather measurement at a time (e.g., temperature only, or humidity only), and (ii) a *virtual multi-sensor* consisting of an aggregation of individual virtual sensors allowing to capture multiple weather measurements simultaneously. This allows flexibility and modularity in designing different kinds of virtual sensors based on the user's needs.

We define a virtual sensor as a spherical Unity 3D game object having four main properties: (i) *location* (defining the sensor's location or point of origin within the studied area), (ii) *sampling frequency* (describing how often the sensing process occurs), (iii) *coverage* (describing the sensing range, as a certain radius from the point of origin), and (iv) *measurement accuracy* (describing the sensor's precision in data capture). Here, we adopt three pre-defined SSN/SOSA classes (cf. Sect. 2.1.1) that can describe the last three above-mentioned properties: (i) *owl:Frequency*: the smallest possible time between one observation, actuation, or sampling and the next, (ii) *owl:MeasurementRange*: the defined range from which the sensor can return the result of an observation, and (iii) *owl:Accuracy*: the closeness of agreement between the result of an observation and the true value of the observed property under the defined conditions. Note that the SSN/SOSA KG does not offer a class description for *location*, since it identifies the location of a sensor based on the identifier of the platform it is hosted in. Hence, we add the *location* property concept to our sub-graph with no prefix attached, describing the $<x, y, z>$ coordinates in the Unity 3D virtual space referential.[4]

3.2.2 Virtual weather measurements sub-graph

The *virtual weather measurements* sub-graph is shown in Fig. 8. It designates the weather measurements and their properties within the simulation environment. We define a virtual weather measurement as a spherical Unity 3D game object having four properties: (i) *location* (designating the measurement's center point, or point of origin), (ii) *coverage* (designating the measurement's coverage area, described using the sphere's radius from the point of origin), (iii) *value* (designating the weather measurement's value at the point of origin), and (iv) *dissipation* (designating the measurement value dissipation as one moves away from the point of origin, toward the extremities of the coverage area). For instance, a temperature pocket would have

[4] Note that the *location* property can be substituted by any other location definition concept depending on the referential space being used (e.g., from GML [35] or KML [19] in a geo-referential space).

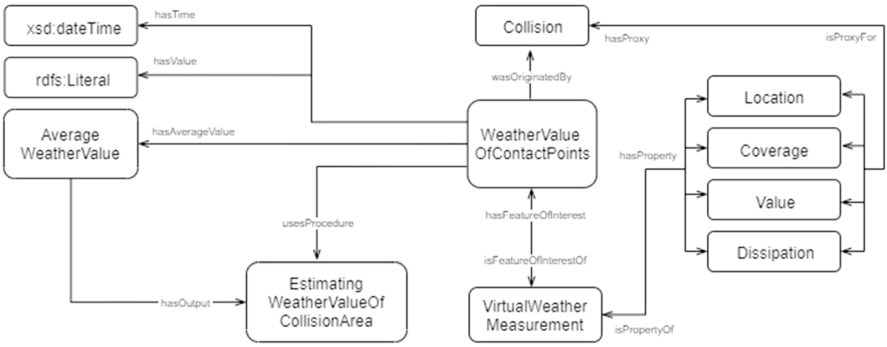


Fig. 9 *Weather sensing procedure* sub-graph

a temperature value at its point of origin, and then the temperature would dissipate as one moves away from the point of origin. The dissipation process is described in the following section.

3.2.3 Virtual sensing procedure sub-graph

Our *virtual sensing procedure* sub-graph is shown in Fig. 9. It defines a virtual sensing procedure as a collision event between two Unity 3D spherical game objects: the *virtual sensor* on the one hand, and the *virtual weather measurement* on the other hand.

We make use of Unity 3D's collision events in defining the sensing procedure: (i) the *on collision enter* event is called when a sensor object and a measurement object intersect, (ii) the *on collision stay* event continuously executes as long as the sensor and measurement objects collide, and (ii) the *on collision exit* event is executed, producing a *collision break* message when the sensor and measurement objects are no longer in collision. A virtual sensor is stimulated by its collision with a virtual weather measurement. The collision initializes a weather measurement sensing algorithm executed within the *on collision stay* event (cf. Fig. 10). The algorithm considers the contact points intersecting the *virtual sensor* and *weather measurement* range and coverage areas, respectively, which make up the *collision area* (lines 1–4), computes the intersecting contact point distances with the weather measurement's center point (lines 5–6), computes the weather value at each contact point considering its timestamp, distance, and dissipation properties w.r.t. the weather measurement (lines 7–8), and adjusts the weather values considering the virtual sensor's detection accuracy (lines 9–10). Note that the timestamp is described using SSN/SOSA's *date-time* property as shown in Fig. 9. The algorithm finally computes and returns the average weather value for all intersecting points in the collision area (lines 12–15). The latter represents the weather value that is sensed by the virtual sensor during the collision event.

Note that the above-described computation algorithm is made efficient by Unity 3D, which allows seamless and instantaneous access to the contact points in a collision area. We believe the latter process allows a realistic simulation of sensors in

Algorithm: *EstimatingWeatherValueOfCollisionArea*

Input: *Virtual Sensor* and *Weather measurement* properties
Output: *Weather value* of collision area between *virtual sensor* and *weather measurement*

Begin

While *onCollisionStay*()
{
For each *contact point p* intersecting *virtual sensor coverage* and *weather measurement* objects
{ // $\forall\ p \in N$, where *N* denotes the number of intersecting contact points

Compute *distance* between *contact point p* and *weather measurement* center point *c*:

$$Dist(p,c) = \sqrt{(p.x - c.x)^2 + (p.y - c.y)^2 + (p.z - c.z)^2}$$ // We use Euclidian distance here,
// yet another distance measure can be used

Compute *weather value* at contact point *p*:

$$weatherValue(p, t_i) = weatherValue(c, t_i) \times (1 - dissipation\%)^{\,Dist(p,\,c)-1}$$

// computed w.r.t. the time stamp t_i and dissipation percentage of the *weather measurement*,
// starting with the absolute *weather value* at the *weather measurement* center point *c*,
// following the definition of a geometric sequence: $a_n = a_1 \times r^{n-1}$

Adjust *weather value* following *virtual sensor* accuracy:

$$weatherValue(p) = weatherValue(p) \times accuracy\%$$

}
Compute *average weather value*:

$$averageWeatherValue = \frac{\sum_{p \in N} weatherValue(p)}{N}$$ // *N* is the number of intersecting contact points

}
Return *averageWeatherValue* // weather value of collision area

End

Fig. 10 Pseudo code for *estimating weather value of collision area* between virtual sensor and weather measurement objects

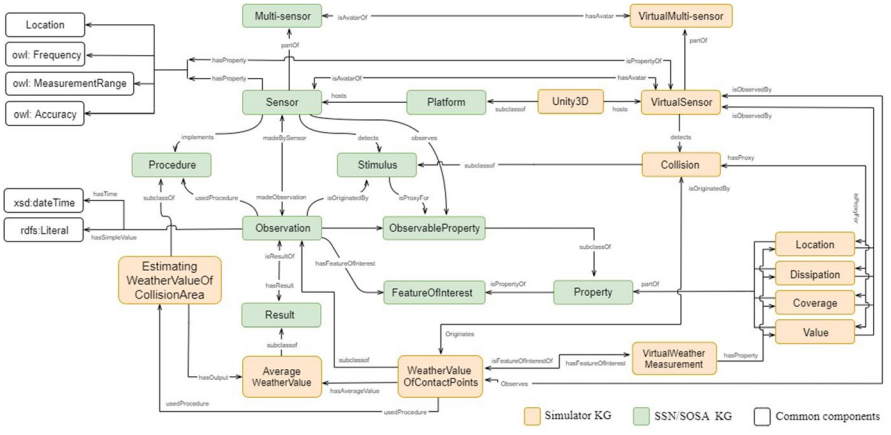


Fig. 11 Simulator KG

the real world, where a sensor would make multiple measurements as it approaches or touches a certain weather pocket, producing values that can dissipate depending on the sensor's distance from the weather pocket's point of origin during the sensor measurement time.

3.2.4 Integrated simulator knowledge graph

The integrated Simulator KG is shown in Fig. 11. It combines the virtual sensor, virtual weather measurements, and virtual sensing procedure components mentioned above, and connects them through the SSN/SOSA KG, bridging the gap between the physical and virtual components. Similar to virtual multi-sensors, we introduce the concept of physical *multi-sensor* as an aggregation of individual physical sensors, capable of capturing multiple weather measurements simultaneously. This allows flexibility in designing different kinds of physical sensors, following the same design logic and modularity adopted with virtual sensors (cf. Sect. 3.2.1). To bridge the gap between physical and virtual components, we introduce the *avatar of* relationship, allowing to represent virtual sensors, virtual multi-sensors, and virtual weather measurements as avatars of their physical counterparts. The *avatar of* relationship is designed to seamlessly replicate physical sensors and physical weather measurements, by propagating their knowledge properties and values from the real world into the simulated environment.

3.3 Integrated VOWES knowledge graph

Our integrated VOWES KG is shown in Fig. 12. It connects the Weather KG and the Simulator KG through the SSN/SOSA KG to form the knowledge backbone of our simulation environment. We utilize *weather measurements* and *virtual weather measurements* as the physical and virtual sensors' features of interest, respectively, whereby any added sensor can measure any weather property (e.g., *temperature*, *humidity*, *wind*, *dust*). In other words, the *virtual weather measurements* sub-graph (Fig. 8) serves as the central connector between the Weather KG (Fig. 6) and the Simulator KG (Fig. 11). The resulting KG then connects with the SSN/SOSA KG (through five main relationship instances: (i) *Unity3D—subclassOf—Platform*, (i) *WeatherWeatherValue—subclassOf—Result, EstimatingWeatherValueOfCollisionArea—subclassOf—Procedure*, (iv) *VirtualSensor—isAvatarOf—Sensor*, and (v) *VirtualMulti-sensor—isAvatarOf—Multi-sensor*) in order to produce the integrated VOWES KG.

4 VOWES simulation tool

We design and develop our VOWES simulation tool using the Unity 3D game engine to build the environment and its virtual sensors, and integrate them with real-world 3D maps and a weather API for realistic weather mapping. Guided by the integrated VOWES KG described in the previous Sect. 3.3, we develop special visualizations and behaviors to present weather measurements, events, and sensors as visible 3D structures with specifications controllable by the user. The following sub-sections describe the main components of our simulator tool.

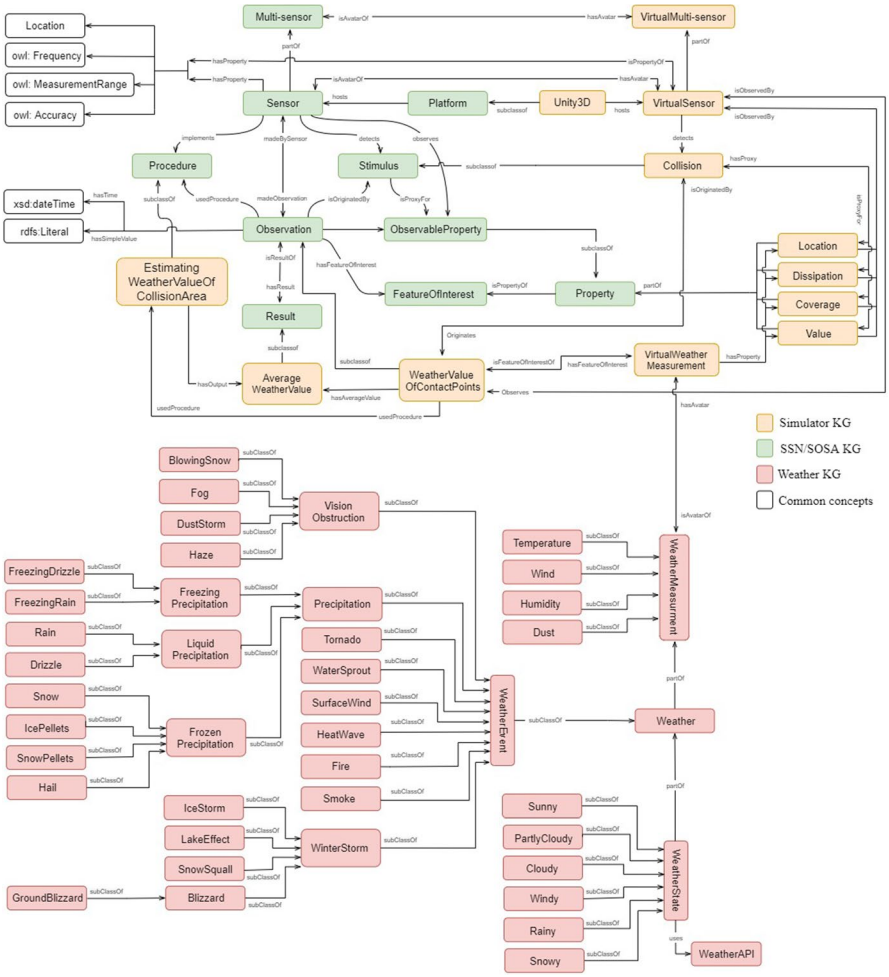
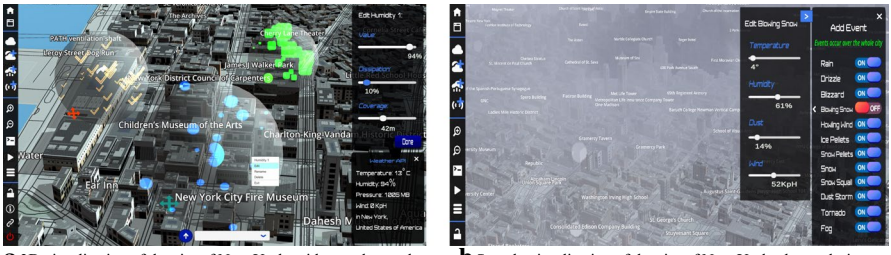


Fig. 12 Integrated VOWES KG

4.1 Virtual 3D world

To achieve a realistic 3D simulation of outdoor environmental events and measurements, we make use of Unity's flexibility in integrating third-party APIs to acquire dynamic 3D maps and real-world weather measurements. More specifically we utilize the Mapbox SDK [31] to import high-resolution world maps showing countries, cities, and buildings, and we integrate the WeatherStack API [58] to capture real-time weather measurements and conditions from the geographic area that is being simulated.

Mapbox offers APIs, SDKs, and live-updating map data, allowing to build better mapping, navigation, and search experiences across different platforms [31]. We utilize the Mapbox SDK to import the 3D maps of real-life cities and allow the user



a 3D visualization of the city of New York, with sample weather measurements and some of their parameters

b Sample visualization of the city of New York, shown during a snow storm in late January 2021

Fig. 13 VOWES simulation tool snapshots of the city of New York

to explore and visualize those cities from within the Unity 3D environment, with high levels of detail, where particular locations or buildings can be easily leveraged for procedurally generating user-specific experiences or styling. Users are prompted to select their city of choice upon launching a new simulation project. Consequently, the data layers are imported and built into the Unity 3D environment, including buildings data, points of interest (POIs), roads, and real-time traffic data, where the data can be fully customized within Unity 3D's development environment (e.g., changing the layout of certain buildings, adding a building, removing or changing the properties of a road, etc., cf. Fig. 13a). In addition, we utilize the WeatherStack API [58] to acquire real-time weather data for the selected city being simulated by the user,[5] while storing a 14-day historical record of the weather information. The historical record is useful to allow weather forecasting through the simulator. Following the user's selection of the city of interest, and upon launching the simulation project, the tool automatically acquires and processes the real-data weather information and presents the corresponding visualizations and behaviors on-screen (cf. Fig. 13b).

4.2 Virtual weather measurements and events

We develop a dedicated weather simulation module using Unity 3D's Particle System graphics [52] to create dynamic weather objects, visualizing and simulating the behaviors of *weather measurements* (e.g., *wind*, *humidity*, *temperature*) and *weather events* (e.g., *storm*, *tornado*, *fire*). Guided by the Weather KG presented in Sect. 3.1, we utilize Unity's particle system to render small images, called particles and control their collective behavior to produce visual effects where every particle within the system presents an individual graphical element in the effect. Every particle system is modeled as a 3D sphere object with mutable boundaries, serving as a container for a blob of particles associated with the target weather measurement or

[5] WeatherStack API is utilized by more than 75 k companies worldwide, providing multi-year history data all the way to live information [53].

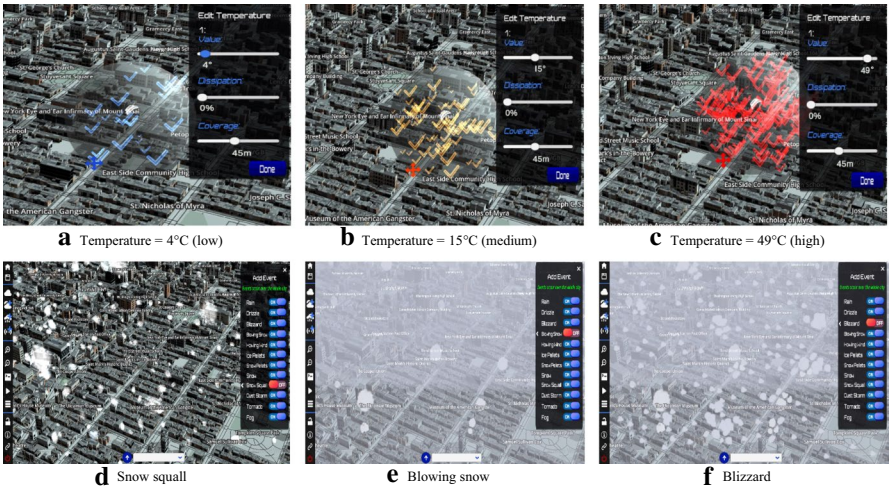


Fig. 14 Simulation snapshots with varying temperature measurements (**a**, **b**, and **c**) and snow events (**d**, **e**, and **f**)

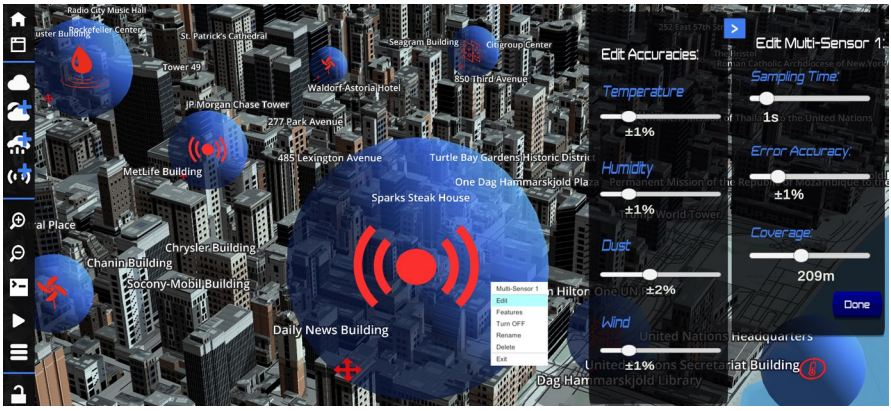


Fig. 15 Virtual sensor configuration panel

event. The object's properties can be defined and fine-tuned by the user through controllable parameters (e.g., coverage, value, dissipation) as seen in Figs. 13 and 14.

4.3 Virtual sensors and multi-sensors

Guided by the Simulator KG graph described in Sect. 3.2 (cf. Fig. 11), we define a *virtual sensor* as a spherical Unity 3D game object with mutable boundaries, having user-controllable properties including *location* (coordinates of the sphere's center point), *measurement range* (radius of the sphere), *sampling rate* (frequency of capture), and *sampling accuracy* (precision of capture, cf. Fig. 15). Every weather

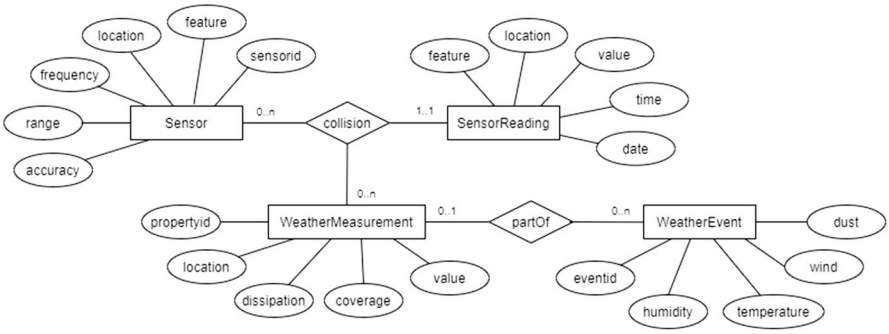
measurement is associated with an identifying tag, which is assigned to the corresponding virtual sensor objects once its measurable feature is chosen by the user. The user can easily toggle between the sensors' measurable features using their identifying tags. The tags help identify all virtual sensor game objects without the need for any additional manual code writing or Unity scripting. A *virtual multi-sensor* is modeled as a set of multiple overlapping 3D sphere objects where each sphere object represents an individual virtual sensor. This allows a multi-sensor to capture multiple weather measurements from its constituent virtual sensors and allows flexibility and modularity in designing different kinds of virtual sensors.

In case of a collision, the sensor is initiated to identify the type of the game object with which it has collided, based on the latter's identifying tag. As a result, the Unity 3D physics engine[6] calls the collision-driven functions associated with the weather objects involved in the collision event, to start the proper sensing process based on the corresponding weather measurements. On the first physics engine update where the collision is detected, the *on collision enter* function is called reflecting the initial state of collision. This signals the sensor's collision detection. Consequently, and while continuous contact is maintained, function *on collision stay* is called to update the sensor readings database where the data is stored (cf. Sect. 4.4), and to provide brief and informative messages for the user via the Unity console. The sensed values are based on the user-chosen properties for the corresponding weather measurement or the event object. Knowingly, the sensor starts first finding the contact points with the weather game object, estimating the corresponding weather value at each point, accumulating the average of all points, and showing the output values to the user through the database console. This process is done continuously until no weather item is detected within the sensing range. As soon as the collision ends, the function *on collision exist* indicates that contact has been broken between the sensor and the weather game objects, signaling the end of the weather measurements sensing process.

4.4 Environment data storage

The data generated through the VOWES simulation environment, including virtual weather measurements and events, as well as virtual sensor properties and readings, are organized and stored in a relational database structured following the integrated VOWES KG described in Sect. 3.3 (cf. Fig. 12). We use a normalized relational database model, compared with a non-normalized No-SQL model, given our need to comply with the well-defined integrated VOWES KG, and given the structured nature of the data following the latter KG. Figure 16 shows the database conceptual schema and corresponding sample data snapshots from the simulator tool. The data from every simulation project are saved in the database, with its timestamp under the user's account, and can be utilized by the user to save, exit, reload, refresh and query the simulation project. The data are also essential to allow the development

[6] Physics engines include packages allowing to simulate real-world physical properties in the virtual environment.



a Conceptual ER model describing an extract of database design

b Sample data produced by the VOWES simulator tool

Fig. 16 Extract of the VOWES database schema and sample data

of data monitoring, mining, and extrapolation functionalities, including project versioning, temporal querying, measurement forecasting, and event prediction. For instance, while VOWES does not currently perform forecasting and prediction, yet it will allow visualizing predicted events once their data becomes available. In other words, VOWES will allow the user to easily fast-forward (or fast-backward) in time to visualize the weather environment and its events in the future (or in the past), according to the available temporal data in its database. The predicted events and

their measurements will simply plug into VOWES and benefit from all its knowledge representation and visualization functionalities. The latter are outside the scope of this work and will be addressed in a dedicated future study.

5 Experimental evaluation

We have conducted various qualitative and performance evaluations to assess the quality and behavior of VOWES' KGs and the software simulation tool. The experimental setup and results are described in the following sub-sections.

The prototype system including the KGs and the simulation software is available online.[7]

5.1 Knowledge graph evaluation

We have conducted a qualitative evaluation involving 13 expert testers to assess the quality of our Weather KG, Simulator KG, and integrated VOWES KG. We have also performed query evaluation against a set of SPARQL queries targeting every KG. We describe both experiments and their results in the following sub-sections.

5.1.1 Qualitative evaluation

We created an online survey[8] to evaluate the design quality of our KGs considering six evaluation criteria including: (i) *accuracy*, (ii) *completeness*, (iii) *conciseness*, (iv) *adaptability*, (v) *clarify*, and (vi) *consistency* (cf. Table 1). A total of 13 expert testers (3 full professors, 3 associate professors, 3 assistant professors, 2 research engineers, and 2 post-docs, cf. Fig. 17[9]) were invited to contribute to the experiment, where they independently rated every KG and evaluation criterion on an integer scale from 0 to 4 (i.e., from *highly dissatisfied* to *highly satisfied*). A total of 234 responses were collected, with every KG receiving 78 rating scores. Results in Fig. 18 show the average rating scores and their standard deviations aggregated for every KG and every criterion, to evaluate KG design quality.

[7] http://sigappfr.acm.org/Projects/VOWES/.

[8] Available at: https://forms.gle/KmXeFYru9boqc23F7.

[9] The names of the participating experts are also mentioned in the acknowledgements: Caetano Traina Jr., Ph.D., Full Professor, University of Sao (USP) Paulo, Brazil, William Grosky, Ph.D., Full Professor, University of Michigan (UM), USA, Agma Traina, Ph.D., Full Professor, University of Sao (USP) Paulo, Brazil, Richard Chbeir, Full Professor, University of Pau and Pays Adour (UPPA), France, George Khazen, Ph.D., Associate Professor, Lebanese American University (LAU), Lebanon, Regina Ticona, Ph.D., Associate Professor, Catholic University of San Pablo (UCSP), Peru, Fekade Getahun, Ph.D., Associate Professor, Addis Ababa University (AAU), Ethiopia, Gilbert Tekli, Ph.D., Assistant Professor, University of Balamand (UoB), Lebanon, Khouloud Salameh, Ph.D., American University of Ras AL Khaimah (AURAK), UAE, Nathalie Charbel, Ph.D., Research Engineer, Nobatek, France, Lara Kallab, Ph.D., Research Engineer, OPEN Group, France, Sabri Allani, Ph.D., Post doc, UPPA, France, and Elio Mansour, Ph.D., Post doc, UPPA, France.

Table 1 KG quality evaluation criteria

Criterion	Description	Evaluation question
Accuracy	Indicates that the concepts (axioms) of a KG accurately describe and comply with the domain knowledge	Given the criterion's description, how accurate are the concepts of the KG w.r.t. your domain knowledge?
Completeness	Indicates that the domain of interest and the needed semantic descriptions are appropriately covered in the KG	Given the criterion's description, how complete is the KG?
Conciseness	Indicates that the ontology includes the needed concepts regarding the application domain, without redundant representations or useless additions to the semantics	Given the criterion's description, how concise is the KG?
Adaptability	Indicates the KG's ability and flexibility in updating and adapting its concepts and relationships to match occurring needs. This usually depends on and reflects how far the KG anticipates its usage	Given the criterion's description, how adaptable is the KG?
Clarity	Indicates the KG's effectiveness in communicating the intended meaning of the defined concepts	Given the criterion's description, how clear is the KG
Consistency	Indicates that the KG does not include any contradictions	Given the criterion's description, how consistent is the KG

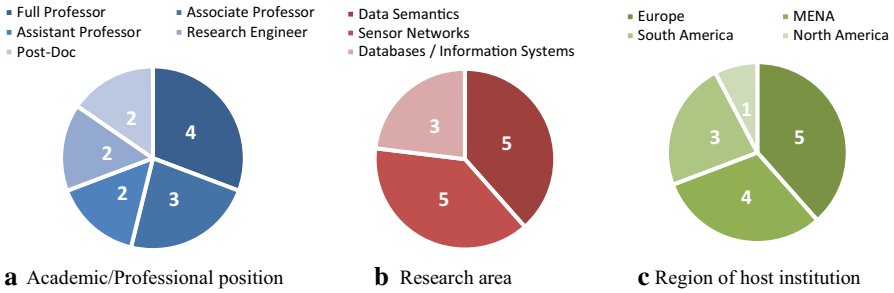


Fig. 17 Expert testers' academic/professional positions, research areas, and institutions

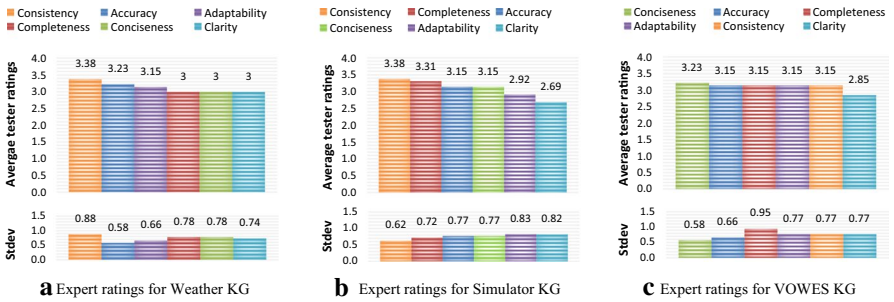


Fig. 18 Average expert tester ratings for Weather KG, Simulator KG, and the integrated VOWES KG

Results for the Weather KG in Fig. 18a indicate that the average rating scores for all six evaluation criteria are ≥ 3 out of 4 (coined with relatively low standard deviations ≤ 0.88), which means that most expert testers are satisfied with the design quality of the Weather KG. Feedback provided by the testers confirms that the Weather KG's concepts and relationships seem coherent, comprehensive, and meaningful, with no contradictions or redundancies. Results for the Simulator KG in Fig. 18b show that average rating scores for most evaluation criteria are ≥ 3 out of 4, except for *adaptability* and *clarity* which produce average scores of 2.92 and 2.69, respectively. Feedback provided by the testers shows that the labels of some of the concepts used to build the Simulator KG seem ambiguous (namely *WeatherValueOfContactPoints*, *AverageWeatherValue*, and *EstimatingWeatherValueOfCollisionArea*) due to their unfamiliarity with the Unity 3D engine capabilities and features. This is also because the Simulator KG is the first of its kind, where the concept labels might seem original and initiative to the testers. The latter is also reflected in the results of the integrated VOWES KG in Fig. 18c, where the average scores for most criteria are ≥ 3 out of 4, except for *clarity* which is equal to 2.85.

To further elaborate on the obtained results, Fig. 19 depicts the percentage of testers with regards to the rating scores obtained for each evaluation criterion. Results in Fig. 19a show that the majority of the testers are satisfied with the accuracy of the KGs, where more than 91, 91, and 83% of the testers gave an *accuracy* score ≥ 3 out of 4 for the Weather KG, the Simulator KG, and the integrated KG, respectively.

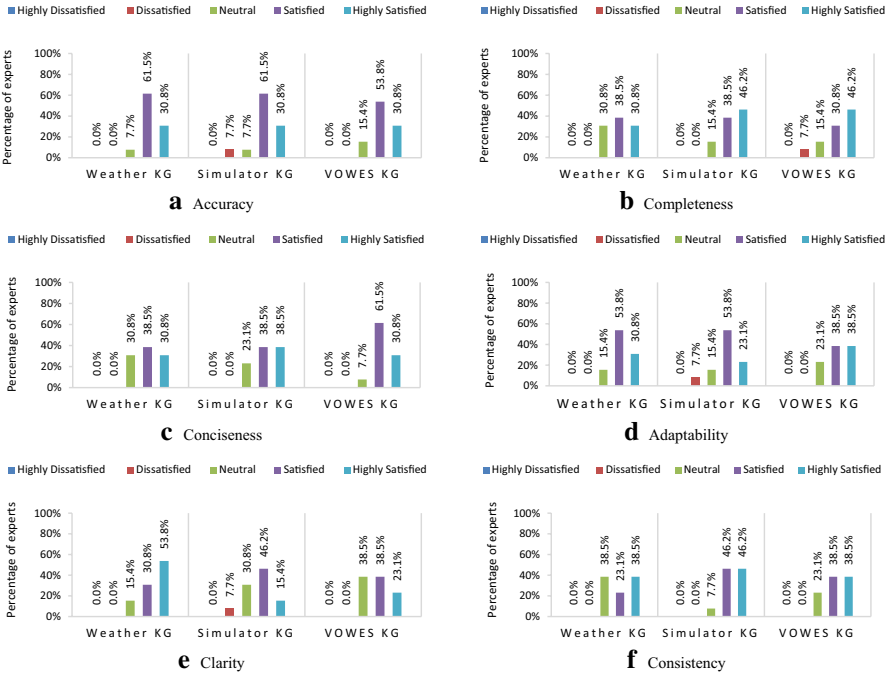


Fig. 19 Percentage of expert testers producing different KG rating score levels for each evaluation criterion

Similar results are obtained for most of the other criteria. Nonetheless, *adaptability* results Fig. 19d shows that a significant portion of the testers, i.e., 23%, produced average scores $\in \{0, 1\}$ out of 4 when rating the Simulator KG. Similarly, *clarity* results in Fig. 19e show that 38% of the testers produced average scores $\in \{0, 1\}$ out of 4 when rating the Simulator KG. These numbers reflect the scores of the testers who seemed to be confused with some of the Simulator KG concept labels mentioned in the previous paragraph, due to their unfamiliarity with the Unity 3D engine and its features.

To sum up, results clearly show that most expert testers are satisfied with all three KGs considering most evaluation criteria, highlighting the design quality of the produced KGs. Note that a few concept labels in the Simulator KG were deemed ambiguous by certain testers, due to their unfamiliarity with Unity 3D. To handle this issue, we have added brief textual descriptions (in the form of textual glosses) to the mentioned concepts definitions to clarify their meanings (e.g., the gloss of *ContactPointCoordinates*: the $<x, y, z>$ coordinates of a set of representative intersection points between two colliding Unity 3D objects).

5.1.2 Query evaluation

We have generated a dataset of 1500 data instances based on the integrated VOWES KG, to evaluate its accuracy and consistency in answering user queries. The dataset

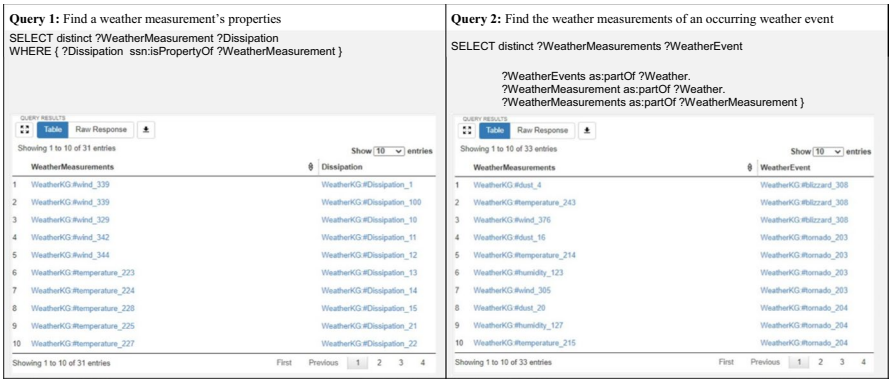


Fig. 20 Sample top-10 outputs of queries 1 and 2

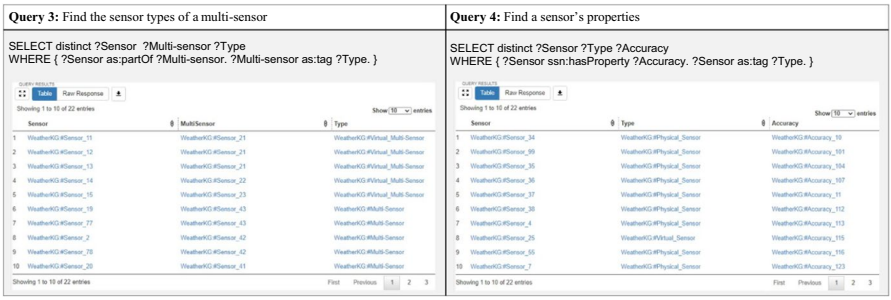


Fig. 21 Sample top 10 outputs of queries 3 and 4, respectively

is composed of 100 randomly generated instances from each of the following KG concepts: physical *multi-sensor*, *virtual multi-sensor*, physical *sensor*, *virtual sensor*, *dust*, *humidity*, *temperature*, *wind*, *rain*, *fire*, *tornado*, *dissipation*, *accuracy*, *collision*, and *weather value of contact points*. We describe below the results obtained with 8 typical SPARQL queries, considering weather data diversity, sensor data diversity, and weather-sensor connectivity. The integrated VOWES KG and the SPARQL queries were implemented and tested using the Protégé ontology editor tool.[10]

Weather diversity: queries 1 and 2 aim at finding all weather measurements, occurring weather events, and their weather properties, to verify that all weather phenomena information can be accurately and distinctly extracted following the users' requests. Results in Fig. 20 reflect successful and accurate data extraction describing the aforementioned inquires.

Sensor diversity: queries 3 and 4 aim at finding all multi-sensors, their types (virtual or physical), the single sensors that they contain, and their sensor features

[10] https://protege.stanford.edu.

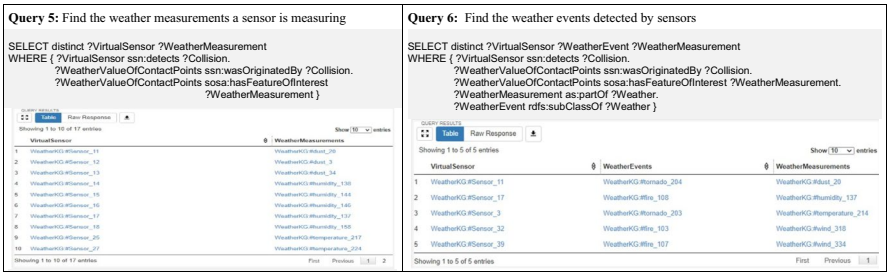


Fig. 22 Sample outputs of queries 5 and 6, respectively

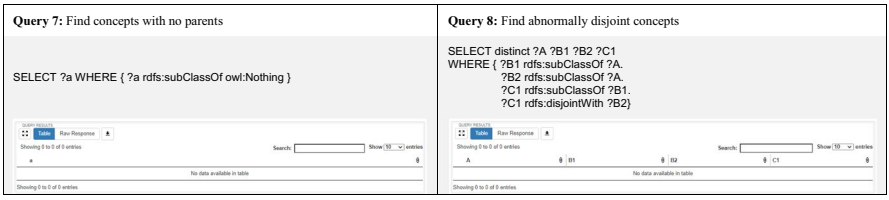


Fig. 23 Outputs of queries 7 and 8, respectively

and properties, in order to verify that all sensor information can be accurately and distinctly extracted following the users' requests. Results in Fig. 21 reflect successful and accurate data extraction describing the aforementioned inquires.

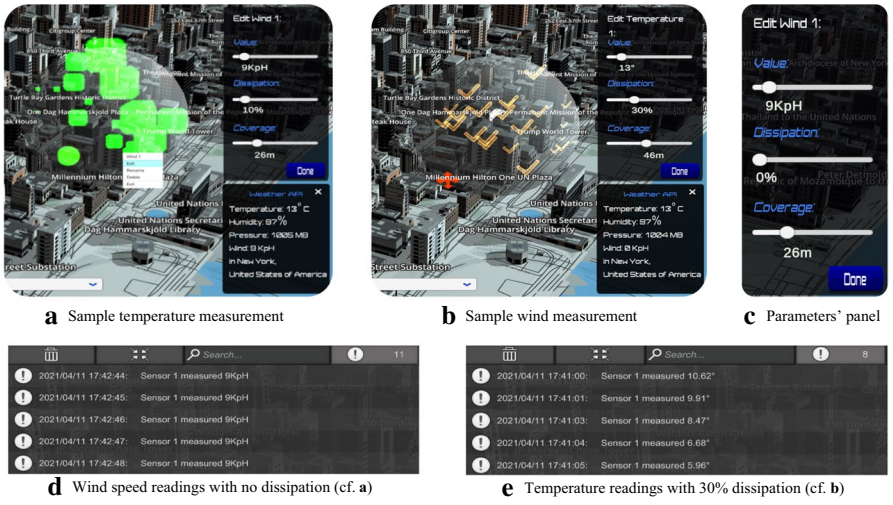
Weather-Sensor connectivity: queries 5 and 6 aim at finding all weather phenomena and properties that each sensor is detecting within its sensitivity coverage area, to verify that all relevant weather information can be accurately and distinctly extracted based on their generating sensors. Results in Fig. 22 reflect successful and accurate data extraction describing the aforementioned inquires.

KG consistency: queries 7 and 8 aim at finding incongruousness among KG entities, such as concepts with no parents or missing relationships, to verify whether certain KG concepts or relationships include inconsistencies or contradictions. Results in Fig. 23 reflect zero query results highlighting the KG's overall consistency and cohesion.

Query time performance results are described in the following section.

5.2 Simulation tool evaluation

In addition to evaluating the quality of VOWES' KGs, we also assess the quality and performance of the software simulation tool itself, considering three evaluation criteria: (i) *simulation accuracy*, (ii) *user friendliness*, and iii) *time performance*. We describe the evaluation criteria and discuss their results in the following sub-sections.



a Sample temperature measurement **b** Sample wind measurement **c** Parameters' panel

d Wind speed readings with no dissipation (cf. **a**) **e** Temperature readings with 30% dissipation (cf. **b**)

Fig. 24 Display of weather measurement properties (**a**, **b**, **c**) and sensor readings (**d**, **e**)

5.2.1 Simulation accuracy

An essential and critical feature in our simulator is the functionality of the *virtual sensor* (and *virtual multi-sensor*) component(s). As described in Sect. 4.3, a virtual sensor is designed to mimic the behavior of a real sensor in the virtual simulation environment, by capturing weather measurements (e.g., *temperature*, *humidity*, *wind*) based on the occurring weather event. To test the accuracy of the weather measurements made by virtual sensors, we refer to the real-time weather values given by the integrated weather API, which are set as the initial values for any weather measurement or event as seen in Fig. 24a and b. The weather values provided by the weather API are regularly updated in the simulation tool, to highlight the real weather conditions in the chosen geo-location being simulated. In addition to comparing virtual sensor values with those of the weather API, we test the performance of the virtual sensors by checking their readings in comparison with the selected weather measurements and their associated properties (e.g., *value*, *dissipation*, *location*, *coverage*). For example, if we select a *wind* measurement and set *dissipation* to 0% (cf. Fig. 24a), we expect the sensor to capture the same specified wind speed value returned by the API as long as it occurs within its coverage area, regardless of its collision location (cf. Fig. 24d). Yet if we set the temperature dissipation parameter to 50%, and we incrementally move the virtual sensor away from the weather measurement's location, we expect the sensor to capture temperature values at a decreasing rate of 50% considering the sensor's collision location w.r.t. the temperature measurement location (cf. Fig. 24e). We follow the above approach by modifying all the weather measurement properties and checking the virtual

sensors' measurements accordingly. For every property, we consider 10 variations of equal spans (e.g., *temperature* varies between −30, −20, …, 60° Celsius, *dissipation* varies between 0, 10, 20, …, 100%). The results produced for all property variations and tests consistently concur with the virtual sensors' expected measurement readings and behavior, highlighting their simulation accuracy.

5.2.2 User-friendliness

The VOWES tool is designed to allow non-expert users who have no previous knowledge about the simulation tool to be able to easily utilize it and benefit from its functionalities. Hence, we evaluate the tool's user-friendliness by performing two kinds of evaluations: (i) GUI[11] testing, and (ii) usability testing. The former aims at checking the GUI's input fields and components and their impact on simulation display, while the latter aims at checking the ease/difficulty of usage of the software tool by non-expert users.

GUI Testing: In this experiment, we check the display of input fields and buttons on the screen considering the aspects of size, alignment, and content. We also check the menu and parameter panels of the application by testing their buttons and mouse hovering functionality, and their impact on the main display. This is applied on all user-interfaces in the whole simulator, starting from testing the capability of generating more than one project simultaneously (through the main page), to the ability to select a country/city and viewing it in a 3D environment, as well as scrolling and zooming in and out of the map with high resolution and details. We also evaluate and test the ability to add weather measurements and events in the same simulation project, and we test the functionality of the designed buttons by pressing each button more than 50 times consecutively. In addition, we make sure that all the weather measurements are movable around the map, by relocating every one of them more than once. We apply the same testing on the virtual sensors, where we perform 10 consecutive addition, renaming, deletion, and movement operations on every sensor in the simulation exercise. Similarly, we test up to 10 separate projects by launching every project using a different city map, populating it with weather measurements, weather events, and virtual sensors, saving it, closing it, re-opening it, and verifying that the sensors, measurements, events, and their values and locations are correctly loaded and initialized, respectively. Furthermore, we test the parameter panels associated with every visual component by checking the functionality of its buttons and range sliders (describing *coverage*, *value*, and *dissipation*, cf. Fig. 24.c) and observing their impact on the visual component. Results of all GUI tests were successful and allowed fine-tuning and improving certain visual aspects and behaviors in the simulation tool.

Usability Testing: We also created an online survey[12] to evaluate the usability and user-friendliness of our simulation tool considering nine evaluation criteria:

[11] Graphical User Interface.

[12] Available at: https://forms.gle/F6odKynC9pcvmCzq6.

Table 2 Simulation tool usability evaluation criteria

Criterion	Description	Evaluation question
Stability	It is the ability of the software tool to function over a long period of time without crashing	Given the criterion's description, how satisfied are you with the stability of the simulation tool?
Look and Feel	It refers to the first impression a user has after using the software tool	Given the criterion's description, how satisfied are you with the look and feel of the simulation tool?
Ease of Use	It describes how easy and straightforward it is to use and manipulate the software tool	Given the criterion's description, how satisfied are you with the ease of use of simulation tool?
Functionality	It refers to the capacity of the software tool to provide useful functions and features serving its main objective	Given the criterion's description, how satisfied are you with the functionality of the simulation tool?
Responsiveness	It refers to the time it takes the software tool to execute a certain action or behavior	Given the criterion's description, how satisfied are you with the responsiveness and overall speed of this application?
Format	It refers to the materials and options provided (e.g., buttons, instructions) and their organization within the software tool	Given the criterion's description, how satisfied are you with the format of this simulation tool?
Navigation	It refers to the interactions that allow users to navigate across, into, and back-out of the software's format (e.g., back to the main page, opening/closing side menus, zoom in/out)	Given the criterion's description, how satisfied are you with the navigation of this simulation tool?
Icon Intuitiveness	It reflects how easy it is to guess a button's resulting action or behavior before a user presses it	Given the criterion's description, how satisfied are you with the intuitiveness of the icons of this simulation tool?
User Interface	It is the means through which a user controls a software application and interacts with it	Given the criterion's description, how satisfied are you with the interface of this simulation tool?

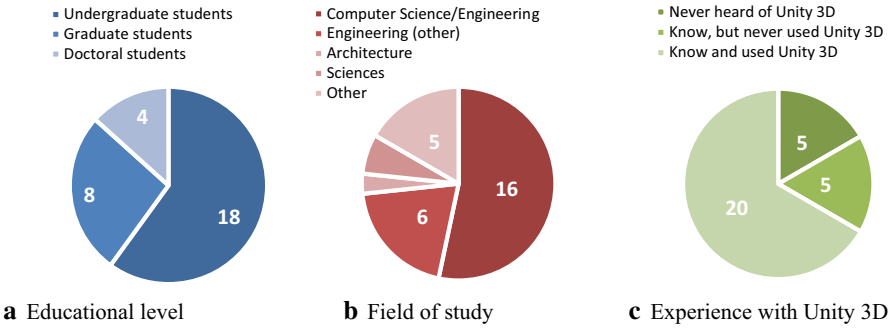


Fig. 25 Non-expert testers' education levels, majors, and experience with Unity 3D

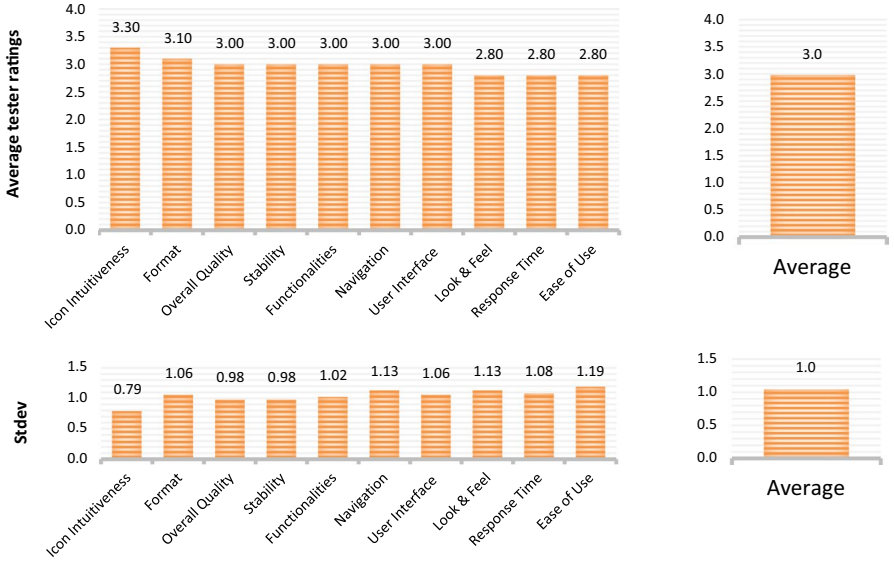


Fig. 26 Average non-expert rating scores produced for every usability criterion

(i) *stability*, (ii) *look and feel*, (iii) *ease of use*, (iv) *functionality*, (v) *responsiveness*, (vi) *format*, (vii) *navigation*, (viii) *icon intuitiveness*, and (ix) *user interface* (cf. Table 2). A total of 30 non-expert testers (undergraduate and graduate students, cf. Fig. 25) were invited to contribute to the experiment, where they independently rated every evaluation criterion on an integer scale from 0 to 4 (i.e., from *highly dissatisfied* to *highly satisfied*). Tests were conducted on a network version of the tool made available through the university's computer labs, where every computer lab consists of an HP ProLiant ML350 Generation 5 (G5) Dual-Core Intel XeonTM 5000 processor with 2.66 GHz processing speed and 16 GB of RAM. A total of 170

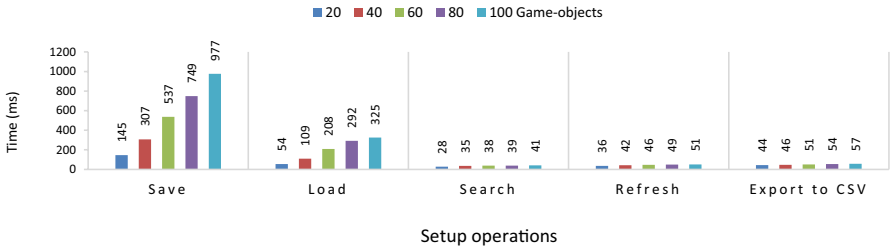


Fig. 27 Execution time of setup phase data processing operations

responses were collected, with every criterion receiving 30 rating scores. Results in Fig. 26 show the average rating scores and their standard deviations aggregated for every criterion. Results show that most testers are satisfied with the tool's usability, producing an overall average rating of 3 out of 4 considering all criteria combined. Note that three criteria received average scores below 3: *look and feel* (2.80), *ease of use* (2.80), and *responsiveness* (2.80). Discussions with the testers revealed that the latter are generally due to the perceived loading time delays of certain Unity 3D components, visual effects, or animations, which probably require increased processing power. This is a common issue with most 3D rendering environments due to their high processing and memory requirements and can be improved with the usage of GPUs and other hardware and software enhancements. Note that few testers recommended including additional features like: (i) considering the impact of outside weather conditions on indoor environments (e.g., indoor heating/cooling systems), and (ii) including pollution-related measurements (e.g., carbon dioxide concentration) and their impact (e.g., carbon footprint) on the overall environment behavior. We plan to consider the latter features in a future study.

5.2.3 Time performance

In addition to simulation accuracy and user-friendliness, we also test the tool's time performance under heavy data loads, complex database queries, and repetitive user input commands. The following paragraphs highlight and discuss the time results obtained during the tool's: (i) setup phase and (ii) simulation phase. Experiments were conducted on an HP ProLiant ML350 Generation 5 (G5) Dual-Core Intel XeonTM 5000 processor with 2.66 GHz processing speed and 16 GB of RAM.

Setup Phase: The simulation tool allows the user to visualize sensors, weather measurements (e.g., *wind*, *humidity*, *temperature*), and weather events (e.g., *storm*, *fire*, *tornado*) as objects with editable and controllable parameters. As such, we evaluate the tool's setup phase by measuring the time to create and load large numbers of game objects, ranging over: 20, 40, 60, 80, and 100 different objects where half of them represent sensors and the other half represent weather events and measurements. For instance, we start by adding 10 sensors and 10 weather phenomena with random values for their attributes. Consequently, we measure the time consumed to *save* and then *load* these game objects from the database, along with their respective

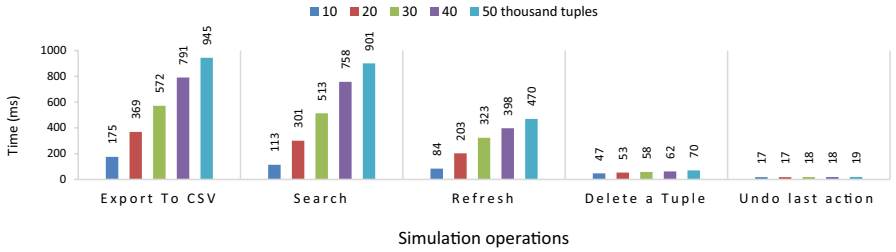


Fig. 28 Execution time of simulation phase data processing operations

features. In addition, we measure the time to *search*, *refresh*, and *export* the game objects' data from the database, which are necessary to keep track of all the sensors and weather phenomena placed or edited in a project environment. Results in Fig. 27 show that most setup operations run in almost instantaneous time, where *search*, *refresh* and *export* operations share almost identical performance levels with execution time increasing by approximately 179 μ s for every added game object. However, we note that the *save* and *load* operations clearly consume more time than their counterparts, which is expected since the tool loads all game objects simultaneously when the user imports a project, and it saves all the game objects simultaneously when the user moves into the simulation phase or exists the project.

Simulation phase: This phase demonstrates the sensors' behavior in action, where sensors are detecting the weather measurements within their coverage areas, based on the features specified by the user. Each sensor works following its internal sampling rate, collecting data from the environment and storing them in the database. As a continuation of the setup phase evaluation, we create 50 sensors with a sampling rate of 0.1 s (i.e., every 0.1 s, all sensors carry out their reading calculations simultaneously and store the results in the database). We evaluate the time performance of sensor reading queries considering large numbers of data tuples ranging over: 10 k, 20 k, 30 k, 40 k, and 50 k. We evaluate *export*, *search*, *refresh*, *delete*, and *undo* queries, by executing every query 10 times and computing the average execution time. Results in Fig. 28 reflect efficient simulation time, where the maximum consumed time was detected at 945 ms to *export* 50 k tuples (i.e., almost 3.7 MB) into an external CSV file. This highlights the simulation tool's time performance in running large simulation projects, and its ability to simulate complex weather environments with large numbers of sensors and weather phenomena.

6 Conclusion

This paper introduces VOWES, a Virtual Outdoor Weather Event Simulator to replicate and measure outdoor weather events and data in 3D visualizations. We design and implement an integrated knowledge graph (KG) representation for VOWES, by creating two constituent KGs: (i) Weather KG describing weather data and events, and (ii) Simulator KG describing 3D simulation components and properties, and

connecting them with the (iii) Semantic Sensor Network (SSN) KG to form an integrated structure serving as the knowledge backbone of the VOWES simulation environment. We make use of the Unity 3D engine to build and design the simulator environment and introduce special visualizations and behaviors to present weather measurements, events, and sensors as visible 3D structures with specifications controllable by the user. We also integrate the Mapbox SDK to import high-resolution world maps, and the WeatherStack API to capture real-time weather measurements and conditions, allowing for more realistic and accurate simulations. Users can provide their synthetic measurements, events, and virtual sensors to initialize the simulation exercise, and can interact with the system without requiring any previous knowledge about Unity 3D or other software simulators. We have conducted qualitative evaluations involving 13 expert and 30 non-expert testers, to assess the quality of VOWES' KGs and its simulation environment. We have also conducted performance evaluations to test VOWES loading, execution, and data search time, among other features. Results are extremely promising and highlight the system's quality and potential.

We are currently extending our KG representation and simulation environment functionalities to perform virtual indoor 3D simulations, considering the impact of outside weather conditions on indoor environments (namely indoor sensor networks, heating systems, and cooling systems). We are also investigating the impact of data collection [12, 13], and data duplication and de-duplication techniques on the quality and time performance of the simulation environment. Preliminary results in [29] highlight the need to detect and handle temporal and spatial–temporal redundancies and their significant impact in reducing processing time. In the near future, we aim to further extend our KG and simulator environment, by coupling it with regional and global climate models [56] to include pollution-related measurements (e.g., carbon dioxide concentration) and their impact (e.g., carbon footprint) on the overall environment. In the long run, we plan to investigate different machine learning models [16, 36] and evolutionary developmental techniques [2, 43], to perform weather measurement forecasting and event prediction [21, 34], as well as event-based KG evolution [30]. Forecasting, prediction, and evolution functionalities will be added as dedicated plug-and-play software-as-a-service layers on top of the VOWES simulation environment, allowing for model transparency and extensibility.

Acknowledgements We also like to thank all the experts who helped evaluate our knowledge graphs: Caetano Traina Jr., Ph.D., Full Professor, University of Sao (USP) Paulo, Brazil, William Grosky, Ph.D., Full Professor, University of Michigan (UM), USA, Agma Traina, Ph.D., Full Professor, University of Sao (USP) Paulo, Brazil, Richard Chbeir, Full Professor, University of Pau and Pays Adour (UPPA), France, George Khazen, Ph.D., Associate Professor, Lebanese American University (LAU), Lebanon, Regina Ticona, Ph.D., Associate Professor, Catholic University of San Pablo (UCSP), Peru, Fekade Getahun, Ph.D., Associate Professor, Addis Ababa University (AAU), Ethiopia, Gilbert Tekli, Ph.D., Assistant Professor, University of Balamand (UoB), Lebanon, Khouloud Salameh, Ph.D., American University of Ras AL Khaimah (AURAK), UAE, Nathalie Charbel, Ph.D., Research Engineer, Nobatek, France, Lara Kallab, Ph.D., Research Engineer, OPEN Group, France, Sabri Allani, Ph.D., Post doc, UPPA, France, and Elio Mansour, Ph.D., Post doc, UPPA, France. We are also very grateful to all non-expert testers who helped evaluate our software simulation tool.

References

1. Abbasi M (2011) An integrated platform for physical and virtual intelligent sensors. Proquest, Umi Dissertation Publishing, 102 p
2. Abboud R, Tekli J (2019) Integration of non-parametric fuzzy classification with an evolutionary-developmental framework to perform music sentiment-based analysis and composition. Soft Comput 24(13):9875–9925
3. Abebe M et al (2020) Generic metadata representation framework for social-based event detection, description, and linkage. Knowledge Based Systems, p 188
4. Albrecht J et al (2008) Geo-ontology tools: the missing link. Trans GIS 12(4):409–424
5. Angsuchotmetee C, Chbeir R, Cardinale Y (2020) MSSN-Onto: an ontology-based approach for flexible event processing in multimedia sensor networks. Futur Gener Comput Syst 108:1140–1158
6. Avancha S et al (2004) Ontology-driven adaptive sensor networks. In: International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04), pp. 194–202
7. Buyuksalih I et al (2017) 3D Modeling and visualization based on the unity game engine—advantages and challenges. In: 4th International GeoAdvances Workshop, 2017. pp 161–166
8. Chen M, Plale B (2012) From metadata to ontology representation: a case of converting severe weather forecast metadata to an antology. Association for Information Science and Technology Annual Meeting (ASIST'12), pp 1–4
9. Chun S et al (2020) Designing an integrated knowledge graph for smart energy services. J Supercomputing 76(10):8058–8085
10. Devaraju A, Kauppinen T (2012) Sensors tell more than they sense: modeling and reasoning about sensor observations for understanding weather events. Int J Sensors Wireless Commun Control 1(2)
11. Durand N et al (2007) Ontology-based object recognition for remote sensing image interpretation. Tools with Artificial Intell 1:472–479
12. Ebrahimi D et al (2019) UAV-aided projection-based compressive data gathering in wireless sensor networks. IEEE Internet Things J 6(2):1893–1905
13. Ebrahimi D et al (2018) Data collection in wireless sensor networks using uav and compressive data gathering. GLOBECOM, pp 1–7
14. Fares M et al (2019) Difficulties and improvements to graph-based lexical sentiment analysis using LISA IEEE International Conference on Cognitive Computing (ICCC'19), pp 28–35
15. Fares M et al (2019) Unsupervised word-level affect analysis and propagation in a lexical knowledge graph. Elsevier Knowledge-Based Syst 165:432–459
16. Fuentes S et al (2020) Machine learning modeling of wine sensory profiles and color of vertical vintages of pinot noir based on chemical fingerprinting, weather and management data. Sensors 20(13):3618
17. Garcia-Dorado I et al (2017) Fast weather simulation for inverse procedural design of 3D urban models. ACM Trans Graphics 36(2):21:1–21:19
18. Gomez M et al (2008) An ontology-centric approach to sensor mission assignment. In: International Conference Knowledge Engineering and Knowledge Management (EKAW), pp 347–363
19. Google Developers, Keyhole markup language (KML). https://developers.google.com/kml, 2008 (Accessed Nov. 2021)
20. Herrera RT et al (2015) Toward RDF Normalization. Inter. Conference on Conceptual Modeling (ER'15), pp. 261–275
21. Hewage P et al (2021) Deep learning-based effective fine-grained weather forecasting model. Pattern Anal Appl 24(1):343–366
22. Hoffart J et al (2013) YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. Artifitial Intell 194:28–61
23. Jain V, Mahdavi A (2016) Implementation of simulation-based virtual sensors using radiance and java. Appl Mech Mater 824:740–747
24. Jin W, Kim D (2018) Design and implementation of e-health system based on semantic sensor network using IETF YANG. Sensors 18(2):629
25. Li X et al (2019) Primitive-Based 3d building modeling, sensor simulation, and estimation. In: IEEE International Geoscience and Remote Sensing Symposium (IGARSS'19), pp 5148–5151

26. Maguitman A et al (2005) Algorithmic detection of semantic similarity. In: International Conference on the World Wide Web (WWW), pp. 107–116
27. Manola F, Miller E (2004) Resource Description Framework (RDF) primer: model and syntax specification. W3C Recommendation, 2004, http://www.w3.org/TR/rdf-primer/
28. Mansour E, Chbeir R, Arnould P (2019) HSSN: an ontology for hybrid semantic sensor networks. In: International Database Engineering and Applications Symposium (IDEAS'19), pp 8:1–8:10
29. Mansour E et al (2020) Data redundancy management in connected environments. InL International Conference on Modeling, Analysis, and Simulation of Wireless and Mobile Systems (MSWiM-Q2SWinet), 2020, pp. 75–80
30. Mao Q et al (2021) Event prediction based on evolutionary event ontology knowledge. Future Generation Compter Syst 115:76–89
31. MapBox, Mobile Maps SDK v10. https://www.mapbox.com/mobile-maps-sdk, 2021
32. Mezaris V, Kompatsiaris I, Strintzis M (2003) An Ontology Approach to Object-based Image Retrieval. In: International Conference on Image Processing (ICIP'03), 2003. Vol. 2. IEEE, II–511
33. Miller G (1990) WordNet: an on-line lexical database. Int J Lexicography 3(4)
34. Moreno R et al (2020) Seeking the best weather research and forecasting model performance: an Empirical Score Approach. J Supercomput 76(12):9629–9653
35. Open Geospatial Consortium. Geography Mark-up Language (GML). http://www.opengeospatial.org/standards/gml (January 2009)
36. Oses N et al (2020) Analysis of Copernicus' ERA5 climate reanalysis data as a replacement for weather station temperature measurements in machine learning models for olive phenology phase prediction. Sensors 20(21):6381
37. Poveda-Villalón M et al (2018) Ontological requirement specification for smart irrigation systems: a SOSA/SSN and SAREF comparison. In: International Semantic Web Conference (ISWC'18), 2018. pp. 1–6
38. Rana R et al (2010) Ear-phone: an end-to-end participatory urban noise mapping system. In: 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, 2010, pp. 105–116
39. Raskin R, Pan M (2005) Knowledge representation in the semantic web for earth and environmental terminology (SWEET). Comput Geosci 31(9):1119–1125
40. Richardson R, Smeaton A (1995) Using WordNet in a Knowledge-based approach to information retrieval. In: Proceedings of the BCS-IRSG Colloquium on Information Retrieval
41. Roussey C et al (2020) Weather data publication on the LOD using SOSA/SSN ontology. Semantic Web 11(4):581–591
42. Sagar S et al (2018) Modeling Smart Sensors on top of SOSA/SSN and WoT TD with the Semantic Smart Sensor Network (S3N) modular Ontology. International Semantic Web Conference (ISWC'18), pp. 163–177
43. Salloum G, Tekli J (2021) Automated and personalized nutrition health assessment, recommendation, and progress evaluation using fuzzy reasoning. Int J Human-Computer Studies (IJHCS) 151:102610
44. Sanders B et al (2020) Design and validation of a unity-based simulation to investigate gesture based control of semi-autonomous vehicles. In: International Conference on Human-Computer Interaction (HCI'20), vol 10, pp 325–345
45. Schlenoff C et al (2013) A literature review of sensor ontologies for manufacturing applications. In: International Symposium on Robotic and Sensors Environments (ROSE'13), pp 96–101
46. Shahid A et al (2020) Insights into relevant knowledge extraction techniques: a comprehensive review. J Supercomput 76(3):1695–1733
47. Shimizu C et al (2020) Towards a modular ontology for space weather research. CoRR abs/2009.12285
48. Sun L et al (2020) An optimised steelmaking-continuous casting scheduling simulation system with unity 3D. Int J Simulation Process Modell 15(3):213–224
49. Taddesse FG et al (2009) Relating RSS News/Items. In: Proceedings of the 9th International Conference on Web Engineering (ICWE'09), LNCS, 2009, pp 44–452
50. Tekli J et al (2018) Full-fledged semantic indexing and querying model designed for seamless integration in legacy RDBMS. Data Knowl Eng 117:133–173
51. Unity, *Architecture, Engineering & Construction* Unity, 2020. Available: https://unity.com/solutions/architecture-engineering-construction (Accessed Nov. 2021)

52. Unity, *Particle System.* Unity Documentation, available at: https://docs.unity3d.com/ScriptReference/ParticleSystem.html (Accessed Nov. 2021), 2020
53. W3C, *Semantic Sensor Network Ontology.* W3C Recommendation, 2017, www.w3.org/TR/2017/REC-vocab-ssn-20171019/ (Accessed Nov. 2021)
54. W3C, *Extensions to the semantic sensor network ontology,* 2020, www.w3.org/TR/vocab-ssn-ext/ (Accessed Nov. 2021)
55. Wang R et al (2020) Portable interactive visualization of large-scale simulations in geotechnical engineering using Unity3D. Adv Eng Softw 148:102838
56. Wang Y et al (2018) An efficient parallel algorithm for the coupling of global climate models and regional climate models on a large-scale multi-core cluster. J Supercomput 74(8):3999–4018
57. Wazir H, Annaz F (2015) Applicability of virtual reality in the study of environmental stress. Appl Mech Mater 741:209–214
58. WeatherStack, *Real-Time & historical world weather data API.* https://weatherstack.com/, 2021
59. Zhang D et al (2019) Knowledge Graph-based image classification refinement. IEEE Access 7:57678–57690
60. Zhong J et al (2008) Progress for ontology of fractures and faults. In: AAAI Spring Symposium: Semantic Scientific Knowledge Integration, pp 114–115
61. Zigon B et al (2018) Interactive 3D simulation for fluid-structure interactions using dual coupled GPUs. J Supercomput 74(1):37–64

Authors and Affiliations

Hamza Noueihed[1] · Heba Harb[1] · Joe Tekli[1]

✉ Joe Tekli
joe.tekli@lau.edu.lb

Hamza Noueihed
hamza.noueihed@lau.edu

Heba Harb
heba.harb@lau.edu

[1] Department of Electrical and Computer Engineering, School of Engineering, Lebanese American University (LAU), Byblos 36, Lebanon