Designing Guided User Tasks in VR Embodied Experiences

Hui-Yin Wu, Florent Alain Sauveur Robert, Théo Fafet, Brice Graulier, Barthélemy Passin-Cauneau, Lucile Sassatelli, Marco Winckler

Designing Guided User Tasks in VR Embodied Experiences (Preprint)

HUI-YIN WU, Centre Inria d'Université Côte d'Azur, France
FLORENT ROBERT, Centre Inria d'Université Côte d'Azur, CNRS, I3S, France
THÉO FAFET, Université Côte d'Azur, Polytech, France
BRICE GRAULIER, Université Côte d'Azur, Polytech, France
BARTHELEMY PASSIN-CAUNEAU, Université Côte d'Azur, Polytech, France
LUCILE SASSATELLI, Université Côte d'Azur, IUF, CNRS, I3S, France
MARCO WINCKLER, Université Côte d'Azur, Inria, CNRS, I3S, France

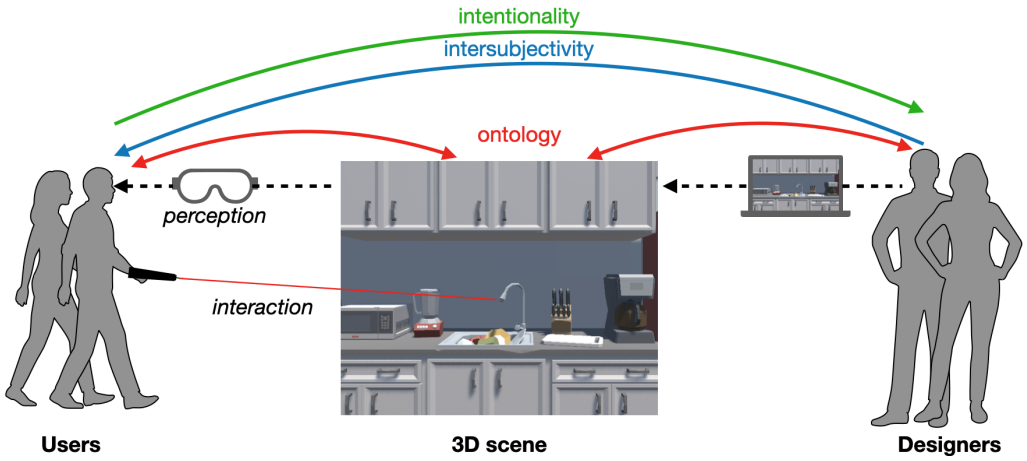

Fig. 1. According to Dourish [14], in the creation of interactive embodied experiences, gaps of perception are introduced in : (1) **Ontology**: between the representation of the scene and the perceptions of the User and Designer, (2) **Intersubjectivity**: between the communication of the goals and constraints of the task from the Designer to User, and (3) **Intentionality**: between the user's intentions and perceptions and the designers interpretations of the user experience.

Virtual reality (VR) offers extraordinary opportunities in user behavior research to study and observe how people interact in immersive 3D environments. A major challenge of designing these 3D experiences and user tasks, however, lies in bridging the inter-relational gaps of perception between the designer, the user, and the 3D scene. Based on Paul Dourish's theory of embodiment, these gaps of perception are: **ontology** between the scene representation and the user and designer perception, **intersubjectivity** from designer to user in task communication, and **intentionality** from the user's intentions to the designer's interpretations.

EICS'22, June 21–24, 2022, Nice, France

ACM ISBN XXX-X-XXXX-XXXX-X/XX/XX…$15.00
https://doi.org/XX.XXXX/XXXXXXX.XXXXXXX

We present the GUsT-3D framework for designing *Guided User Tasks* in embodied VR experiences, i.e., tasks that require the user to carry out a series of interactions guided by the constraints of the 3D scene. GUsT-3D is implemented as a set of tools that support a 4-step workflow to (1) annotate entities in the scene with names, navigation, and interaction possibilities, (2) define user tasks with interactive and timing constraints, (3) manage scene changes, task progress, and user behavior logging in real-time, and (4) conduct post-scenario analysis through spatio-temporal queries using ontology definitions. To illustrate the diverse possibilities enabled by our framework, we present two case studies with an indoor scene and an outdoor scene, and conducted a formative evaluation involving 6 expert interviews to assess the framework and the implemented workflow. Analysis of the responses show that the GUsT-3D framework fits well into a designer's creative process, providing a necessary workflow to create, manage, and understand VR embodied experiences of target users.

CCS Concepts: • **Human-centered computing → Systems and tools for interaction design**; **User studies**; • **Computing methodologies → Virtual reality**; **Ontology engineering**.

Additional Key Words and Phrases: Embodied experiences, interactive task modeling, virtual reality, user experience analysis

ACM Reference Format:
Hui-Yin Wu, Florent Robert, Théo Fafet, Brice Graulier, Barthelemy Passin-Cauneau, Lucile Sassatelli, and Marco Winckler. 2022. Designing Guided User Tasks in VR Embodied Experiences (Preprint). In *EICS'22: ACM SIGCHI Symposium on Engineering Interactive Computing Systems.* ACM, New York, NY, USA, 24 pages. https://doi.org/XX.XXXX/XXXXXXX.XXXXXXX

1 INTRODUCTION

The rising popularity and growth in graphics rendering capabilities has facilitated the adoption of Virtual Reality (VR) to address professional training and simulation needs as diverse as cockpit training [26], engineering education [1], rehabilitation of disabilities [36, 37], prevention of occupational hazards [29], and many more. On the one hand, the *Designer* – the team or individual that creates embodied VR experiences – can realize strong control over the content being presented to the *User* of their system. On the other hand, the equipment – headset, controllers, and external sensors – not only offer natural and rich interactions, but also register user behavior (e.g., movement of the head and hands) that characterize the embodied user experience.

Being able to gain insight into user behavior, perception, and efficacy in a task is key to understanding the user experience and measuring the usability of an interactive system. This has been addressed in research in user-centric design and task modeling [3]. However, in the design of complex interactive systems, gaps of perception may appear in any pairwise relation between the designer, user, and the system itself. In VR, the freedom of movement to visually explore and interact with an immersive 360° environment further widens these inter-relational gaps between the designer's desired user experience, the affordances of the 3D scene in VR, and how the user intends to act and react throughout the experience. Dourish's theory of embodiment [14] identifies three major gaps of perception for embodied experiences (as depicted in Figure 1):

- Ontology: gap between the representation of the scene and the perceptions of the user and designer,
- Intersubjectivity: gap in understanding between the designer to the user when communicating the goals and constraints of the task, and
- Intentionality: gap between the user's intentions and perceptions and the designers interpretations of the user experience.

Contributions: To address these gaps of perception, we propose the GUsT-3D framework to define a consistent set of vocabulary used throughout the design, implementation, and running of *Guided User Tasks (GUTasks)* in a VR embodied experience. The framework enables a workflow consisting of 4 steps: (1) the annotation of various entities in the scene, including navigation spaces, interaction possibilities, object properties and relations, (2) the definition of *GUTasks* and their constraints for completion, (3) running the *GUTask* scenario with real-time interaction, logging, and validation of the task, and (4) analysis of user experience through the construction of spatio-temporal queries and scene graph visualization. This workflow is implemented as a set of tools in Unity to

create and design embodied, interactive experiences in VR. The export of the scenario in GUsT-3D representation contains sufficient information to re-create the entire 3D scene, with all the original geometric, interactive, and navigational properties. Conversely, the *GUTasks* can be exported and imported into a new scene holding the same semantic properties required by the task as the original scene. We validate these contributions with two case studies and a formative evaluation through expert interviews with professional Unity users. The implementations are released as an open toolkit [1].

The rest of the paper continues as follows: the existing work is reviewed in Section 2 with our positioning. We then describe our framework and its implementation in Sections 3 and 4. Section 5 demonstrates the feasibility of our approach illustrated by two case studies covering an indoor escape game scene and an outdoor road crossing scene. The findings of a formative evaluation with six expert interviews are then presented in Section 6 to validate our approach. Finally, we discuss the perspectives for this work in the Conclusion (Section 7).

2 RELATED WORK

We present here the existing work for creating and managing interactive user experiences in 3D virtual environments related to the concept of *Guided User Tasks*. We focus specifically on domain-specific frameworks for specific use cases that provide tools for the creation and management of user experience in 3D, and ontologies for representing interactive 3D applications.

2.1 Frameworks for user experiences in 3D

Designing a fully-integrated workflow for the creation of VR experiences is complex, often composed of a combination of multiple tools, as demonstrated by Vergara et al. [34] who propose such a workflow for the design of applications in engineering education, coupled with suggestions of suitable tools for each step of the workflow. Górski [18] specifically addressed the issue of open access to knowledge within hard-coded industrial VR applications, and proposed a knowledge-based engineering approach where 3D content, interaction possibilities, and logic connections between objects can be configured outside of the VR application. This framework focuses strongly on the creation of the VR application that allows flexible modification on the exported knowledge representation of the application, and further, the runtime-loading of the content – whether visual or audio – from the knowledge representation. The entity-component-system [28] has been popularly used for managing interactions of 2D interfaces, as well as gaming applications.

Recently, Speicher et al. [31] developed a system for creating extended reality experiences (both augmented and virtual reality) from paper prototypes. The workflow consists of hand-crafting the scene from paper materials, capturing the scene with a 360°camera, and importing it into their application. A "Wizard" can then add and manage interactive events and animations based on the user's screen-based gestures. For augmented reality, Gottschalk et al. [19] also proposed a product-user-environment framework for e-commerce in the interior design and furniture sector, with a specific focus on modelling product features (e.g. textures and components), user models to interact with the product, and the configuration of the product in the environment. Another work of interest is that of Li et al. [22] who generate 3D training scenes for wheel chair navigation in VR. The scenes are defined by a number of distance constraints between objects in the scene as well as properties of the wheel chair trajectories (e.g., distance, rotations, path width). However, there is no semantic labeling or scene understanding available for the generated scenes, nor interactive task management.

2.2 Ontologies for creating and representing interactive 3D environments

The most widely known domain specific language for 3D content is X3D [8], formerly known as VRML. It is an XML-based syntax that focuses on the information on the 3D model itself (e.g. vertices, texture, material),

[1] CeCILL licence number will be added to the final version of the paper.

its 3D coordinates (e.g. position, rotation), and animation as trajectories over a specific time frame. A detailed review of similar ontologies for 3D scenes has been done by Flotyński [16], but they have a primary interest in 3D modeling applications and not the creation of 3D interactive experiences. A number of extensions to X3D have allowed it to incorporate more contextual, multi-hierarchical, and ontological representations of scenes [27], as well as incorporate sensors and controller input for virtual applications [2]. Buffa and Lafon [9] have also demonstrated its use for an online interactive 3D warehouse application. Another extension by Flotyński et. al allow the logging of user interactions [15] in VRML applications.

Our work has been strongly inspired by ontologies for robotics and procedural content generation. The RSG-DSL [6] was developed under the Eclipse framework in order to represent robot world model as a scene graph, which allows the attachment of functional blocks to the scene graph to establish the robot's workflow. Beßler et. al proposed an ontology for modeling affordance in robotics tasks [4] coupled with the capabilities for logical reasoning and question answering. The Scenic language [17] was developed for machine learning applications in 3D environments to generate datasets from a high-level scene description to train perception systems such as those for autonomous driving. Similarly, Plan-It [35] allows the generation of 3D indoor scenes from a high-level scene graph definition. Liu et al. [23] introduce the idea of *scene programs* that allow the generation of 3D images from layout descriptions, and conversely reason about layouts of objects from images. For game applications, the GIGL language was recently introduced [12] to procedurally generate game maps such as for dungeons in RPG games.

One of the first formal representations of user interactions for virtual reality was the Interactive Cooperative Objects (ICO) description proposed by Navarre et al. [25]. The work decomposes gestures into a granular graph of states to define selection, rotation, and movement of objects at a low level for specific interactive devices, using a chess game as case study. Vanacken [33] introduces the concept of tasks by designing the "concept" datatype that can be attached to objects to label their interaction possibilities. In this work *task* refers to an interaction technique (such as selecting an object) that is triggered by an event, and results in a state change. More recently, a number of ontologies have been proposed for the semantic modeling of virtual environments [13, 30] with the goal to improve the richness of information about the virtual environment in a database for multi-agent systems.

Our positioning: While many frameworks and ontologies have excellent properties for designing 3D content, they present a number of limitations when considering embodied experiences: (1) reasoning about the user experience within the scene is mostly limited to the geometric properties of 3D objects [8, 27] or interaction in a single usage scenarios [19], (2) they have a strong focus on the 3D geometry, and those that include interactions are defined at the level of the device and interface (e.g. a mouse, controller, gesture) [2, 18, 25, 28, 33] instead of for user tasks (e.g. "take an object", "move to location"), and hence do not allow systematic task validation (3) there is no temporal representation of the world and user state changes, and (4) while some works allow the generation of 3D images or scenes [12, 22, 23, 31, 35], these are not semantically annotated, which firstly, do not have customizable interactive possibilities or user task management, and secondly do not allow a faithful re-creation of a specific scene with task and interactivity information included. A full comparison of the relevant ontologies and frameworks are summarized in Table 1.

In our framework, we set out from a knowledge-based approach to represent and manage the user experience. The central idea is to provide a set of consistent vocabulary throughout all phases of a workflow to create an embodied experience, including the scene design and annotation, user task description, logging and validation of real-time tasks, and post-scenario analysis. This facilitates the smooth transition of scenes, assets, tasks, and interpretations between different steps of a complex workflow for VR experience creation [34]. The export of a GUsT-3D scene includes the geometry of entities in the scene, allowing the rebuilding of the entire 3D scene – with all the original 3D model, interactive, task creation, and navigational properties – in Unity simply through re-importing the file. Similarly, *GUTasks* can be exported and loaded for a new scene with the same ontology

Table 1. Summary table for comparing related contributions including their focus (ontology or framework), application scenarios, and other observations. A positioning summarizing limitations in existing work is indicated for ontology and frameworks. System implementations (with source code) were found for [4, 8, 12, 19, 28, 31].

Focus	Related work	Application scenario	Comments
Ontology	[2, 8, 9, 15, 27]	Web 3D; e-commerce, other varied applications	X3D and its extensions, as reviewed by [16], primarily for lower-level model and mesh representations
	[4]	robotics	an affordance model for Q&A systems
	[6]		robot world model for robotic reasoning
	[12, 17, 23, 35]	games and learning systems	Primarily for procedural content generation
***Positioning on ontology**: existing ontologies are currently not able consider a human agent fully immersed in the 3D scenario, with high level of interactivity, and specific tasks to carry out.*			
Framework	[31]	virtual and augmented reality	wizard of oz system with medium fidelity paper prototyping for more simple scenes and limited interactivity
	[19]	augmented reality; e-commerce	
	[18]	industrial VR applications	knowledge-based approach
	[22]	rehabilitation for wheelchair users in VR	optimization approach to generating constrained 3D scenes
	[28]	2D interfaces; based on entity-component system	
***Positioning on ontology**: existing frameworks are designed with targeted scenarios in mind, with no scene representation with an ontology, and do not address intersubjectivity communication nor analysis of intentionality during and after the user experience.*			

properties, in addition to the one for which it was originally designed, to conduct the same interactive task in different, and even dynamically generated, scene layouts.

3 OVERVIEW

In this paper we propose a framework for creating 3D embodied experiences, shown in Figure 2 and comprised of three main components:

- **GUsT-3D**: representing the **ontology** of the scene, implemented in JSON. It is used to represent the export of the annotated 3D scene, including all of the geometric properties, interactive and navigation possibilities, and their interrelations as a spatio-temporal scene graph. It also serves as the vocabulary set used consistently with the *GUTask* definition and user logging.
- ***GUTasks***: representing the **intersubjectivity** between the designer and the user, also implemented in JSON using the same vocabulary defined in GUsT-3D. Allows the definition of user tasks to be carried out in the VR scenario, their interactive and completion constraints, and management and validation in real-time
- **Logs and query language**: representing the user **intentionality**, implemented in LINQ [24]. Allows the designer to construct spatio-temporal queries on any annotated object or user in the 3D scene.

The implementation of this framework as a unified toolkit in Unity allows the realization of a typical workflow for the design of embodied experiences involving four steps: (1) the design of the 3D scene and annotation of various properties (e.g., interaction, navigation) and calculation of the scene graph (relations between objects), (2) the definition of *GUTasks* that the user should carry out, their constraints and interaction design, (3) the running of the task scenario with real-time guidance, task validation, and logging, and (4) post-scenario analysis of the scene evolution and user behavior through query and visualization tools. The detailed task model for this workflow is depicted in Figure 3.

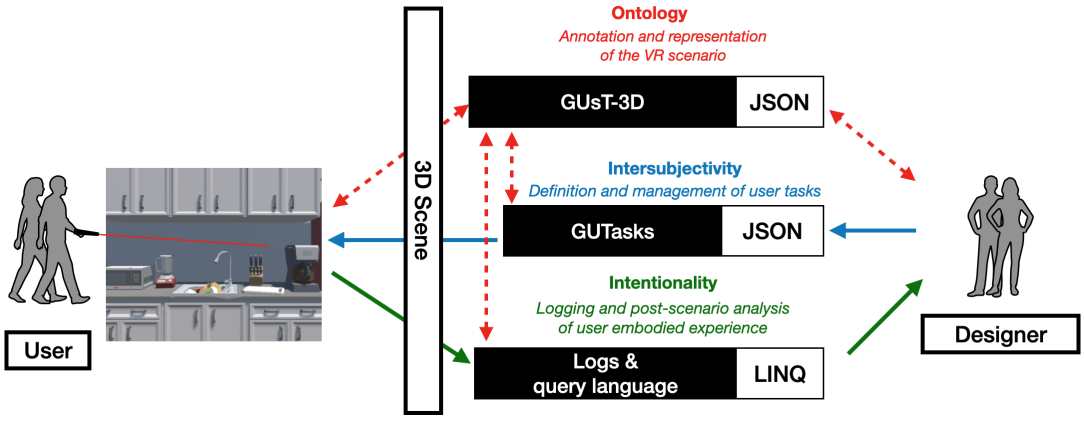

Fig. 2. **3-Component framework** for designing *GUTasks* in 3D virtual environments based on embodiment theory: the GUsT-3D vocabulary which is the full representation of the scene *ontology*, the definition and management of GUTasks communicated between the Designer and User for *intersubjectivity*, and the logging and query of user experience to allow Designers to analyze user *intentionality*.

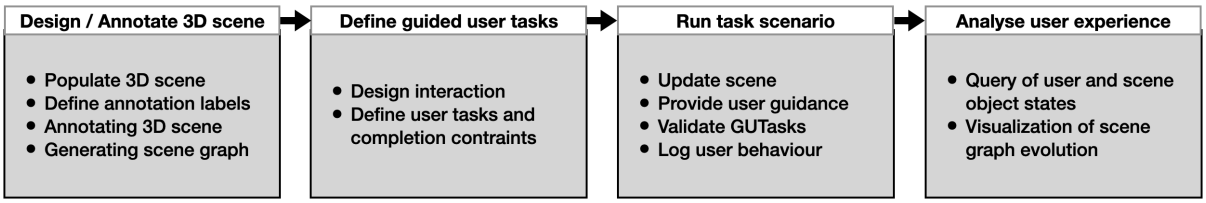

Fig. 3. **4-Step workflow** to create *Guided User Tasks* in 3D environments involves: design and annotation of the 3D scene, defining *GUTasks*, running the task scenario, and post-scenario analysis of the user experience. Each step of the workflow contains a number of sub-tasks that the designer may need to do.

The following section will introduce the tools of this framework and how it integrates into a design workflow for embodied experiences.

4 GUTASK WORKFLOW AND SYSTEM IMPLEMENTATION

This section details the implementation of our framework for the creation of 3D embodied experiences, shown in Figure 2. We first introduce each of the three elements of the framework corresponding to Dourish's theory of embodiment – the GUsT-3D ontology (Section 4.1) for annotation and scene graph visualization of entity relations, definition and management of intersubjectivity (Section 4.2) through *GUTasks*, and the analysis of user behavior for intentionality (Section 4.3) through query and scene graph analysis. We then summarize how this framework fits into the task model depicted in Figure 3.

4.1 Ontology: the GUsT-3D representation

We refer to the elements in the scene as entities, which include all objects, terrain, and the user. The first step is for the designer to establish the ontology of the 3D scene. This includes (1) defining and assigning various

properties (e.g., interactive, navigational) to entities in the scene, (2) defining the constraints for interactions between entities, and (3) generating a global view of entity relations within the scene.

4.1.1 Layers for defining and annotating entities. We introduce the idea of *layers*, which represent a category of an object in the scene and/or its interactive properties.

Currently four categories of *layers* are provided in our framework: navigation, object, interactive, and environment layers.

Navigation layers: structure the space in which the user can move, and defines all the navigation possibilities and constraints within the 3D scene. Three types of navigation layers are currently defined:

- `ground`: the *navigable spaces* (i.e. a contiguous space where the user can move around without constraints), such as a room or the strip of sidewalk of a street
- `entryways` such as doors and passages that serve as connections between these navigable spaces, and
- `obstacles` such as walls that partition these spaces.

The designer can define new layers that inherit the `ground` property to create a named navigable space. Figure 4 shows the visualization of three navigable spaces: `garage`, `kitchen`, and `bedroom`.

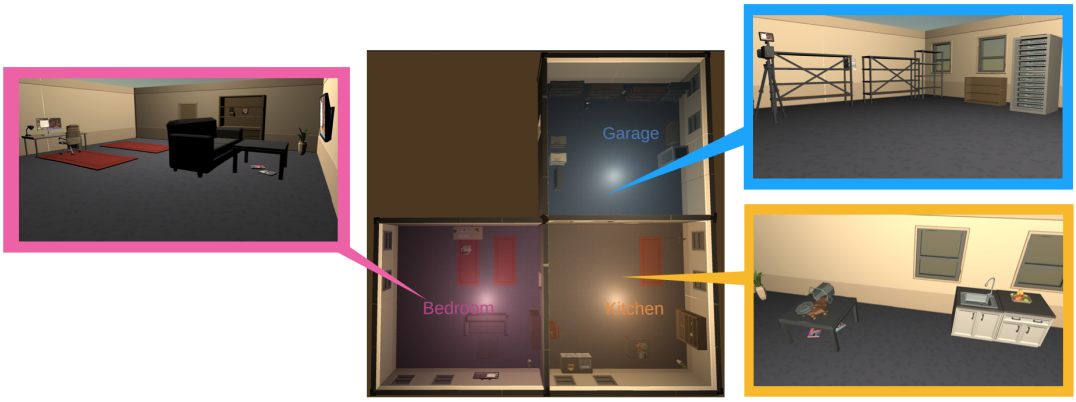

Fig. 4. The possible *navigable spaces* within the 3D scene are visualized in our Unity implementation. This example indoor scene has three navigable spaces: the Garage (blue), the Kitchen (yellow), and the Bedroom (pink). A user viewpoint is shown for each navigable space.

Interactive property layers: describe the relationship between two or more entities in the scene, and indicate how these entities can interact within the 3D scene. These properties exist :

- Between two scene objects: `support`, indicating that an object can support another; `placed-on` the opposite, signifying whether an object can be placed on another `support` object; `container` indicating if an object can contain another object;
- Between an object and user: `movable`, indicating whether an object can be taken and moved around by the user; `actable`, indicating whether the object can be activated, such as a light switch, button, or a remote control.

The default layers are attached to Unity scripts that manage the outcome of an interaction. The designer can easily define new interactive layers and attach scripts to them to enable other types of interaction.

Object layers: layers that have structural or interactive functionalities within the scenes – furniture that can serve as supports or containers, props that can be moved and manipulated. They can inherit other layers such as to take on interactive properties.

Environment layers: are used to annotated entities within the 3D scene that do not change the topology nor structure of the 3D scene, but through environmental parameters, allow users to change the perceptions of the environment. For example, all scenes have a `camera`, which can be extended to be a perceptive agent, such as the user. The camera is often used to represent the user's visual range (i.e. field of view), current position, movement direction, and gaze. Another layer is `light`, which contributes to environmental lighting of the scene.

The flexibility of our layer approach is that objects can be assigned multiple layers, which can also inherit other layers, and thus gaining their annotated properties. For example a `shelf` layer can be defined as inheriting both `support` (e.g. placing a book on it) and `container` since it can hold other smaller items on its shelves. The `ground` layer also inherits `support`, since it has to support furniture and other items.

4.1.2 Interactive constraints. While the definition of an interactive property layer only indicates an interaction possibility, certain constraints to the interaction may exist. For example, one may wish to enforce that during the interaction with certain objects, the target must be in the visual field and within reaching distance of the user.

In order to allow such a reasoning, basic properties of object relations must be defined from lower geometric properties. Take *distance* as an example. Two points p_1 and p_2 have an Euclidean distance of $dist(p_1, p_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$. However, for user experience, the concept of distance can vary based on the object's size, shape, and function. Moreover, we typically use relative terms such as *near* or far. Thus, the definition of vocabulary for user experiences will need to be relative to pairs of objects and users, and the designer needs a mechanism to flexibly redefine them.

Within GUsT-3D, the designer can also redefine these constraints by directly using the pre-defined layers. In GUsT-3D a constraint is a tuple of three entries:

- the (*Source*,*Target*) pair of the interaction, with layers.
- *Criterion* on under which the interaction can take place, which is a boolean *evaluation* on a *property* for a certain *value*. The property can be further of three types:
  (1) Individual property: **a)** a geometric property (i.e., *position*, *rotation*), **b)** the localization of the entity (within a navigable space), and **c)** the layer which the entity belongs to,
  (2) Relational property: **a)** the Euclidean *distance* or designer added definitions such as *near* or far based on the Euclidean distance, **b)** the relative position between entities including *above*, *below*, *contains*, and *holding* (e.g., in the case the user is holding an object), and **c)** visibility constraints between the camera and another entity such as *visible*, *occluded*, and $facing$.
- *State_change* resulting from the interaction

Using this vocabulary, we can then define an interaction constraint, such as the following for *take_object*:

```
near{ "type":"evaluation","property":"distance","eval":"<=","value":2,"unit":"meter
    "},
take_object{
    "type" : "constraint",
    "Source" : "user",
    "Target" : "movable",
    "Criteria":[{"near":["Source","Target"]},{"visible":["Source","Target"]}]
}
```

These are currently limited to relations between at most two entities, but we can imagine incorporating lambda functions to have more complex constraints.

4.1.3 3D Scene graph generation. One of the advantages of having entities in the 3D scene annotated with an ontology is to extract and represent the relations between entities as a scene graph. The scene graph becomes a model that can be used to describe the state of the system from a given perspective in a moment of time. To do this, we take the annotated scene, layer definitions, and the defined interactive constraints, and calculate the scene graph in three steps:

(1) calculation of *navigable spaces*. Ground tiles labelled with the same location are grouped together to define a contiguous space, as shown in Figure 4.
(2) calculation of the navigational graph based on the positioning of `entryways` between the navigable spaces, and
(3) reasoning of *interactive constraints* between all pairs of entities to extract the interactive relations

These are then exported as a `.dot` file which can be visualized as a graph where the nodes denote various entities or entity groups, and edges denote their relations. An example of a scene graph calculated from the annotated scene can be found in Figure 5.

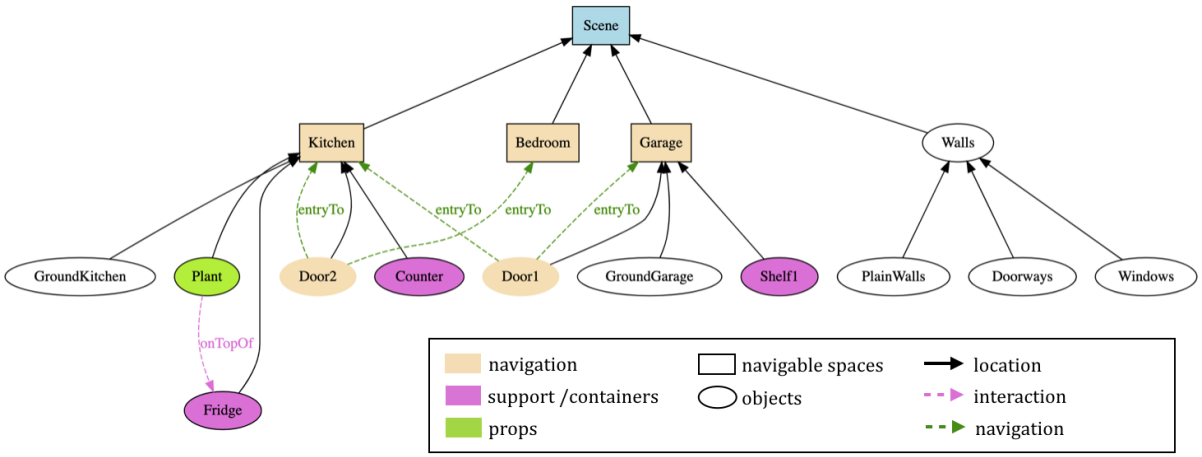

Fig. 5. A partial scene graph generated from the scene to show the relations between various entities in the scene, including navigable spaces and entryways, movable props, objects that serve as support or containers, and other elements such as ground and wall entities. The hard edges denote location relations, whereas dashed edges of different colours indicate interactive and navigation relations between entities.

4.2 Intersubjectivity: defining and managing real-time *GUTasks*

Setting out from an annotated 3D scene, the designer must now define the target tasks for the user to carry out, which we term *Guided User Tasks (GUTasks)* that can be managed in real-time.

4.2.1 GUTask definition. Based on the grammar for interactive constraints and state changes, defining a *GUTask* comprises of specifying a list of constraints for the success of the task. Specifically, each constraint is composed of a time frame, and the desired state to be achieved. Assuming the task scenario is run from time t_0 to t_n, expressed in first order logic, a state can be evaluated at :

- a specific time $State(t_m, [parameters])$
- at some arbitrary point during the task $\exists m(State(t_m, [parameters]))$, or
- at all time steps during the task $\forall m < n \rightarrow State(t_m, [parameters])$

Suppose we would like to define a *GUTask* that requires the user to place a book on the bedroom table, assuming the book cannot move on its own and that it is not already at the target location, we can deduce that the user has to carry out a series of actions such as locating the book, taking it, locating the bedroom and subsequently the table, then finally placing the book on the table. As such the definition of this *GUTask* would be:

```
"move_book":[
    {"evaluation_time":"endOfScenario","State":{"localization":["book","bedroom"]}},
    {"evaluation_time":"endOfScenario","State":{"placed\_on":["book","table"]}}
]
```

The task can also be even more constrained, requiring the user to do multiple actions or achieve multiple states.

It is important to note that the definition of a *GUTask* is independent of any specific scene geometry or configuration: the same *GUTask* could be applied to two different scenes containing objects and entities annotated with the same *layers*. However, if the scene does not contain the objects or layers defined in the *GUTask*, then the *GUTask* is not relevant to the scene, and thus there is no way for the user to achieve the task.

4.2.2 Real-time GUTask management. Our framework also provides easy ways to record user behavior, and manage and validate *GUTasks* when the scenario is run.

The first such mechanism is the recording of logs, at a granularity that can be set to n logs per second in a Unity configuration window. Two types of logs are included: `LogO` created for objects, and `LogU` for the user. Each newly launched instance of the scene creates a separate log file. Each log contains the following information:

- **Movement** of the position and rotation of the entity,
- **State Changes** of any defined interactive property (from Section 4.1.2). Since these can be numerous, the designer can define a "watchlist" of those that should be logged, and
- **Interaction** that results in a state change to a scene entity, as a target-source pair.

Part of the JSON schema for the user logs is as below:

```
{"title":"Scenario Log",
  "definitions":{
    "LogU":{
      "properties":{
        "timestamp":{"description":"Current timestamp","type": "integer"},
        "position": {"description":"in world coordinates", "$ref":#Vector3"},
        "item": {"description":"User-held items", "type": "array"},
        "localization": {"description":"Occupied navigable space", "type": "string"},
        ...}},
    "LogO": {
      "properties":{
        "timestamp":{"type": "integer"},
        "position":{"description":"in world coordinates","$ref": "#Vector3"},
        "interactions": {"description":"Interaction details", "type": "array", "items":{"$ref":"#Interaction"},
        "localization": {"description": "Occupied navigable space", "type": "string"}
        ...}},
    "Interaction": {
```

```
      "properties":{
        "Isource":{"description": "Interaction source", "type": "string"},
        "Itype": {"description": "Type of interaction", "type": "enum"}
  }}},
  "properties": {
    "StartTimestamp": {"type": "string"},
    "UserLogs":{ "type": "array", "items":{"$ref": "#LogU"}},
    "ItemLogs":{ "type": "array", "items":{"$ref": "#LogO"}}
}}
```

These logs contain sufficient information to re-calculate the scene graph for each time step, updating localization and interaction states. The scene graph would allow one to deduce as much fine-grained information as all the entities visible to the user, as well as high-level changes such as an object going from being `placed_on` another object to being held by the user, which is key to analyzing the user experience, presented in the next section.

4.3 Intentionality: reasoning on and visualizing user experience through spatio-temporal queries

The query component of the framework is designed to allow the construction of diverse queries about the scene using a fixed set of vocabulary. Queries can be made on any entity or layer name. The syntax for possible queries under our framework is shown in Figure 6

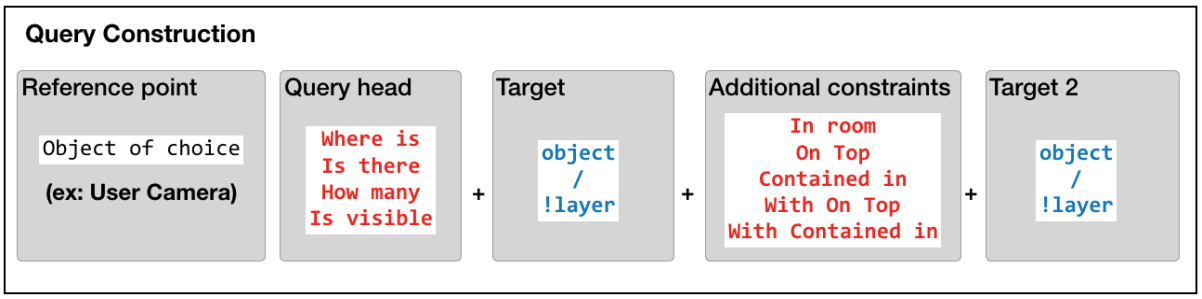

Fig. 6. The syntax of the query is composed of keywords from the vocabulary defined in GUsT-3D including a minimum of three keywords – the reference point (usually the user camera), query head, and target – and potentially two additional keywords that can further limit the search to entities that fulfil specific location or interaction constraints. The result of a runtime query can be seen in Figure 10.

Queries can be constructed on entities in real-time or on the logs post-experience. In real time, entities that match the query are highlighted in the 3D scene. On the logs post-experience, the designer can use this same component to construct spatio-temporal queries, and to observe at each timestamp the temporal evolution of positions of objects, the user, and the interactions that take place within the scene between these entities.

This query component thus allows the designer or observer to monitor all of the entities of interest in the 3D scene. Object and path highlighting aids the designer in clearly defining the objects and layers related to the *GUTasks*, as well as add suitable indications which can be used to guide the user in completing the task.

4.4 Workflow and system integration in Unity

We can now combine the framework and the workflow from Figures 2 and 3 into Figure 7 to show how each element of the framework introduced in this section intervenes with the designer workflow, and the flows of information between the Designer, the User, and the 3D scene.

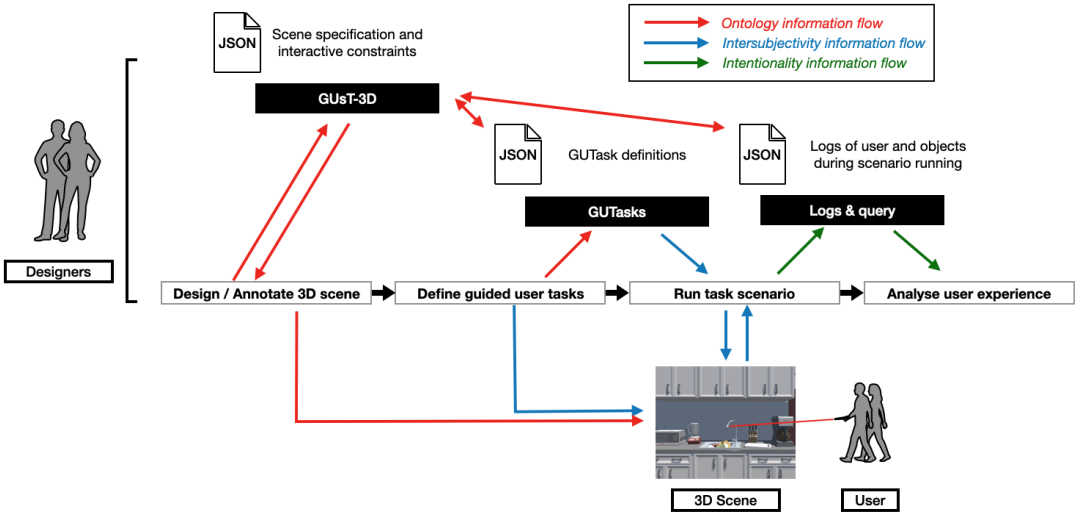

Fig. 7. The combined framework and task model to indicate how each element of the framework intervenes on the workflow of creating a 3D embodied experience. The arrows indicate the flow of information on the three aspects of embodied experiences: ontology (red), intersubjectivity (blue), and intentionality (green)

This framework is realized as a set of editor interfaces in Unity. It realizes our 4-step workflow of the design of an embodied user experience depicted in Figure 3. We detail below concretely how the framework integrates into Unity for each step of this workflow.

4.4.1 Design / Annotate of 3D scene ontology. Scene annotation provides the first step to having a context-aware 3D environment that allows one to make meaningful inferences as to where the user is, what actions they are carrying out and how, and from there analyze the perceptions and intentions of the user. The annotation is facilitated through two means: first, the designer defines the layers for use in a layer configuration window (Figure 8), indicating how layers are inherited, as well as the units and constraints linked to the layers. Then by selecting one or more entities in the scene, they can assign layers to the selection through a point-and-click interface.

We designed an interface that allows the easy addition of layers, interactive constraints, and units. The design schema for the interface is shown in Figure 8, which also demonstrates how layers are structured hierarchically to allow layers inheriting properties of other layers.

The GUsT-3D annotated scene can be exported in JSON format, using Unity's built-in serialization functions. In the exported file, each object's geometric properties (position, rotation, and scale) and layer information are included. We provide an interpreter that allows the recreation of the exact same scene from a GUsT-3D JSON file. The interpreter for the JSON scene representation is implemented as a parser in C# with direct control in the interface to load, update, or export the current configurations.

When the scene has been fully annotated, the navigable spaces and scene graph are calculated with an interpreter for the layer relations and the interactive constraints. The navigable spaces are calculated based on connectivity between the ground tiles and the designated entryways. All entity properties and relations are then encoded by color and style respectively, and exported to a .dot file that provides a scene graph visualization as depicted in Figure 5.

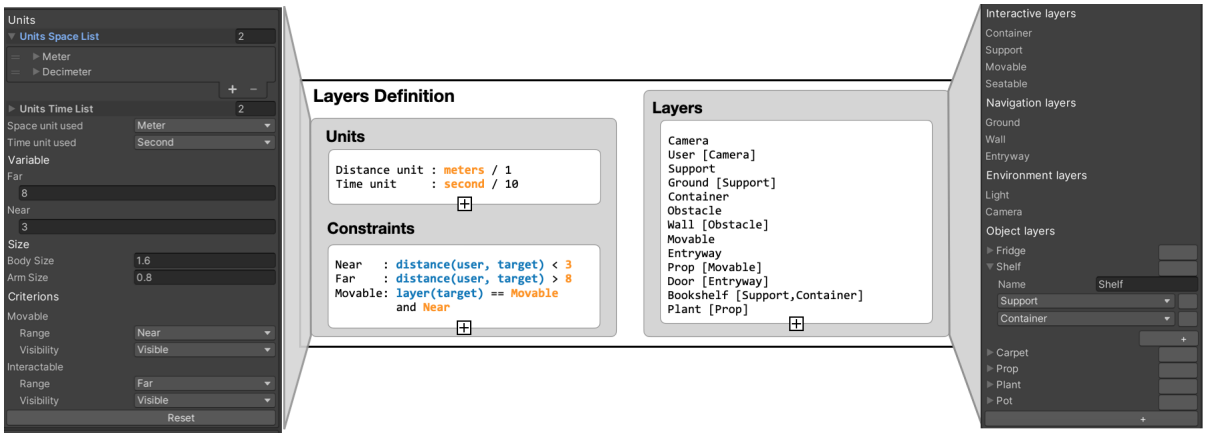

Fig. 8. Schema of the layers definition interface, which allows the scene designer to add and edit the units, constraints, and layers either through point-and-click or importing a configuration file. The actual interface we designed based on the schema is shown on the two sides.

4.4.2 Define GUTasks for intersubjectivity. The next step of the workflow is to establish the intersubjectivity through defining *GUTasks* for the scenario to be communicated from the designer to the user, based on the pre-defined ontology. The interface component in Unity for specifying *GUTasks* is shown in Figure 9, allowing the easy modification of the task goal, constraints, as well as user guidance in the form of text or path indications that can be communicated to the user at specific time points when they have difficulty accomplishing the task.

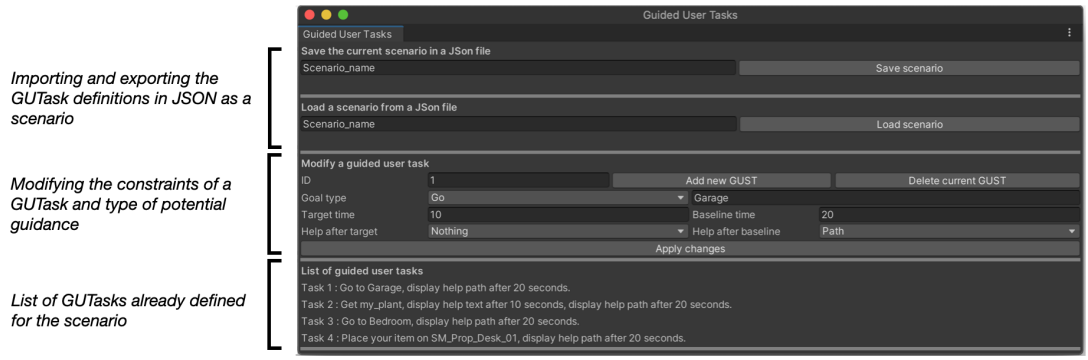

Fig. 9. This interface allows designers to define *GUTasks*, add and edit task constraints, view the list of existing tasks. The tasks can be easily loaded and exported in the GUsT-3D representation for reuse in various scenarios.

The *GUTask* list is also saved in a JSON file in a separate file from the scene representation, using the same vocabulary defined in the GUsT-3D. They use the same layer and annotations from GUsT-3D ontology, which means the designer can easily create and import new *GUTask* lists for the same 3D scene, allowing the same scene to be reused easily for different task scenarios.

4.5 Run task scenario

To run the scenario with interaction, each interactive modality (e.g., navigation, interacting with objects) is bound to an input, such as keyboard or joystick. Collision meshes are added around obstacles such as walls to prevent users from going through them. Interactive constraints such as distance to interaction for an object are evaluated.

Fig. 10. Here we show the window to construct a simple query using the syntax from Figure 6 that counts the number of props within a near distance (3 meters) of the player camera. The two views show the highlighting in the scene from the player viewpoint (left) and the Unity scene view from above (right).

In real time, queries can be composed to search for specific scene entities. These queries to the interface as depicted in Figure 10, are translated in LINQ queries using our C# scripts, which will retrieve all the scene objects that fulfil the query constraints. The implementation then highlights all potential entities of query by modifying the object material with a colored outline, which can be used as indications to the user or designer. While the scenario is running, changes to the scene including the user's camera movement (moving in the scene and head rotation) are recorded, accomplishing of the *GUTasks*, as well as the interactions with objects, and saved to a log file in JSON format. Feedback is also provided to the user to either guide them through the tasks, or indicate the success of the task by adding in-scene text and drawing path line indicators.

4.5.1 Analyze user experience. The analysis of user experience is facilitated through the loading of the aforementioned user logs, in which various information is then visualized. This visualization involves composing queries on specific entities in the scene. The Query Log window allows one to load a log, and drag the time slider to enumerate all the state changes for the specific entity. At the same time, the scene graph is generated for each time stamp, and can be viewed directly in the log window. These elements are shown in Figure 11.

5 CASE STUDIES

We present here two case studies – an indoor and outdoor scene – to show the wide range of possible 3D embodied experiences that can be created with our system and its link to the framework based on embodiment theory.

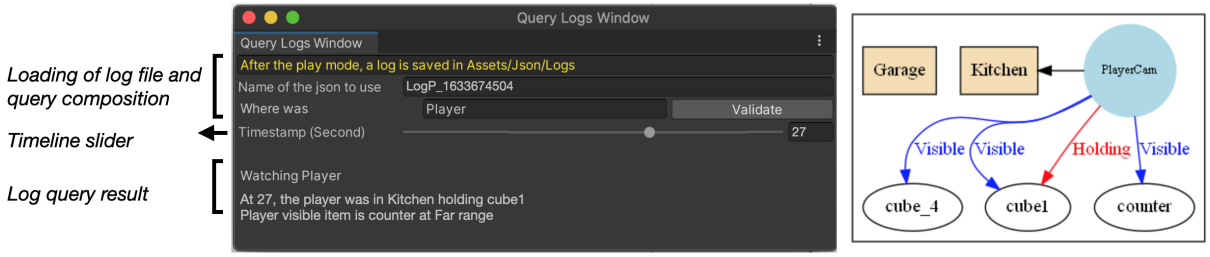

Fig. 11. Window for querying the logs of a previously run scenario, including loading a log file, a slider to show all the state changes recorded to an entity, and the descriptive text on the state at a time point. A generated scene graph of the user's interaction state at a specific time point is shown on the right, indicating, for example, visibility of entities and objects being held.

5.1 Indoor scene: navigation and object interaction

The first case study uses the indoor scene of Figure 4 with three navigable spaces – Garage, Kitchen, and Bedroom – separated by walls, and connected by doors. In an escape game, users commonly have to look around to find indices in the scene, collect objects into their inventory, then use these objects to interact with other entities in the scene. We design a simple *GUTask* for this scene that is similar to an escape game task. The user must (1) locate and retrieve a stack of books from the Bedroom, and (2) navigate to the Garage and place them on the bookshelf to reveal a hidden passage.

The annotation of the scene ontology involves the navigable spaces (ground tiles for *Bedroom*, *Garage*, and *Kitchen*), and the assigning of object and interaction layers to other entities (*movable* such as book stacks or *support* like bookshelves). Multiple elements can be selected and annotated simultaneously with the desired layers and localisation using the interfaces we designed in Section 4.4. A previously annotated scene that was exported in the GUsT-3D representation can also be loaded using the interface, which facilitates the loading of ontologies already pre-defined by other designers.

Next, we define the *GUTask* that facilitates the intersubjectivity communication between the designer and user on what the user should do in the scene. The user starts out in the Bedroom where the stack of books is. Assuming the task is carried out between time t_0 and t_n, the constraints for correctly carrying out this *GUTask* described in first-order logic include :

(1) User at some point must be in the bedroom, near enough to the stack of books, the stack of books must be visible from the user point of view, and the user must interact with the stack of books:
$\exists m(Localisation(t_m, User, Bedroom) \wedge Near(t_m, User, BookStack) \wedge$
$Visible(t_m, User, BookStack) \wedge Interact(t_m, User, Take, BookStack))$
(2) At the end, the user must be in the Garage and the stack of books must be on the garage bookshelf:
$Localisation(t_n, User, Garage) \wedge onTopOf(t_n, BookStack, GarageBookshelf)$

The navigation paths and interactions for the two constraints of the *GUTask* are depicted in Figure 12 in two parts following the above constraints: (1) locating and retrieving the stack of books, and (2) navigating to the Garage to place the stack of books. In the first constraint, there is also a precondition that the stack of books must both be near to the user, and in the visual field of the camera, to be interacted with. Through user logs, we can also observe which objects the user tries to interact with before finding the correct ones, and how much time the user takes to navigate the environment. It is an example of how a simple *GUTask* can be defined in a complex 3D scene with various interactive constraints.

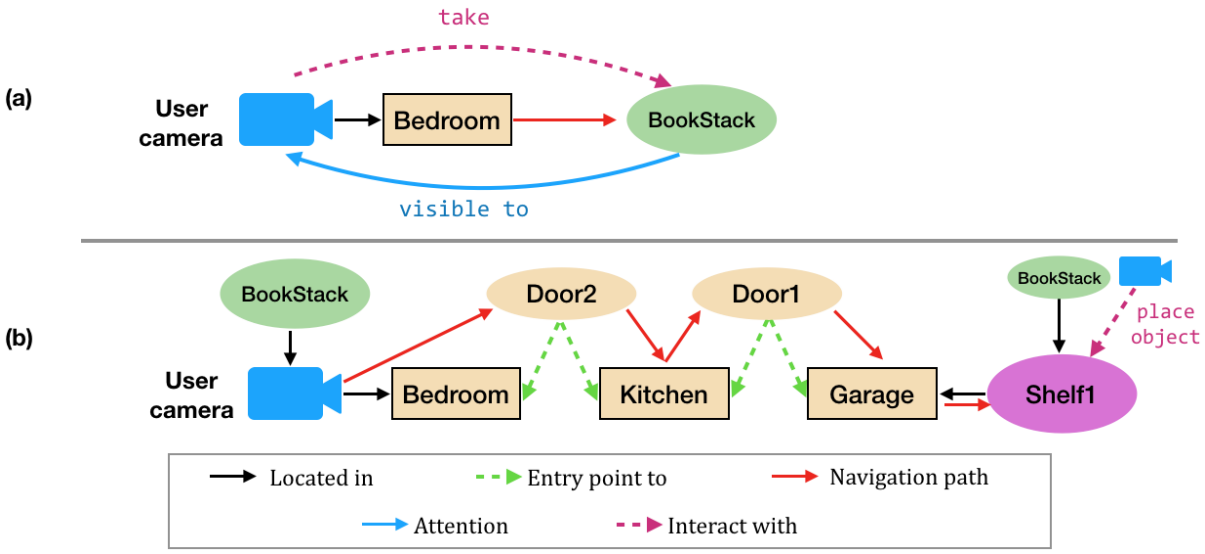

Fig. 12. Given the *GUTask* of placing a stack of books on the bookshelf in the garage, this figure depicts the interaction and navigation paths of the user for two constraints: (1) locating and retrieving the stack of books, and (2) navigating to the garage, and placing the stack of books on the bookshelf. We use the same colour and shape representation of entities as Figure 5. It is worth noting that the interaction with the stack of books in (1) requires evaluating an attention constraint – that the object has to be in the user's visual field.

Finally, we show the example analysis of the user intention using a query to locate an object related to the GUTask, as depicted in Figure 13.

5.2 Outdoor scene: traversing a traffic crossing

The second case study involves a pedestrian crossing as depicted in Figure 14 with a road lined by two sidewalk strips, and connected by a pedestrian crossing.

The *GUTask* defined for this scene is that the user must cross from `Sidewalk2` to `Sidewalk1` in security, involving two sub-constraints: (1) crossing through the pedestrian crossing and not anywhere else on the road, and (2) verifying that the crossing light is green before and during crossing.

The first step is the annotation of the scene ontology. This scene only includes a small number of terrain tiles for different navigable spaces (the two sidewalks, road, and crossing), as well as the traffic light and the timing of its state changes between red and green. The resulting calculation of navigable spaces is shown in Figure 14.

The second step in the workflow is to define the *GUTask* to communicate intersubjectivity from the designer to the user. The initial position of the camera is on `Sidewalk2`. Assuming the task is carried out between time t_0 and t_n, the constraints for correctly carrying out this *GUTask* described in first-order logic include :

(1) User does not at any point appear on the Road:
$\forall m < n \rightarrow \neg Localisation(t_m, User, Road)$
(2) User must at some point look at the light when approaching the crossing from Sidewalk2:
$\exists m(Localisation(t_m, User, Sidewalk2) \wedge Near(User, Crossing) \wedge LookAt(User, Light))$
(3) User can only be on the crossing when the light is green:
$\forall m(Localisation(t_m, User, Crossing) \rightarrow State(t_m, light, green))$

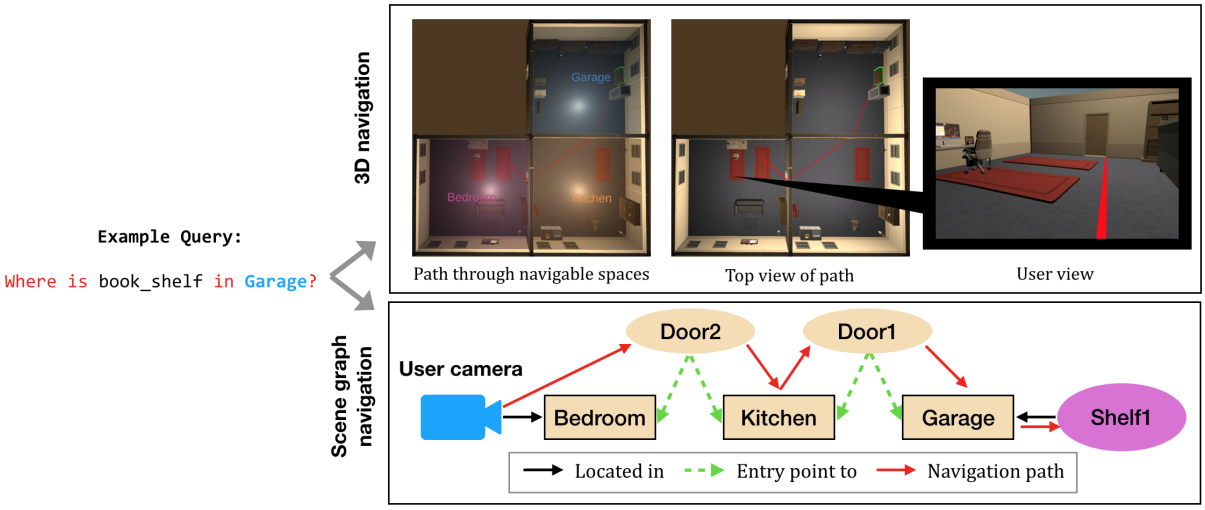

Fig. 13. This figure shows the result of the query to locate a bookshelf in a specific location. In the results, we show both the navigation path as visualized in the 3D scene with red indications as well as object highlighting. Below is the proposed navigation path represented as a subset of the scene graph, using the same colour and shape representation of entities as Figure 5.

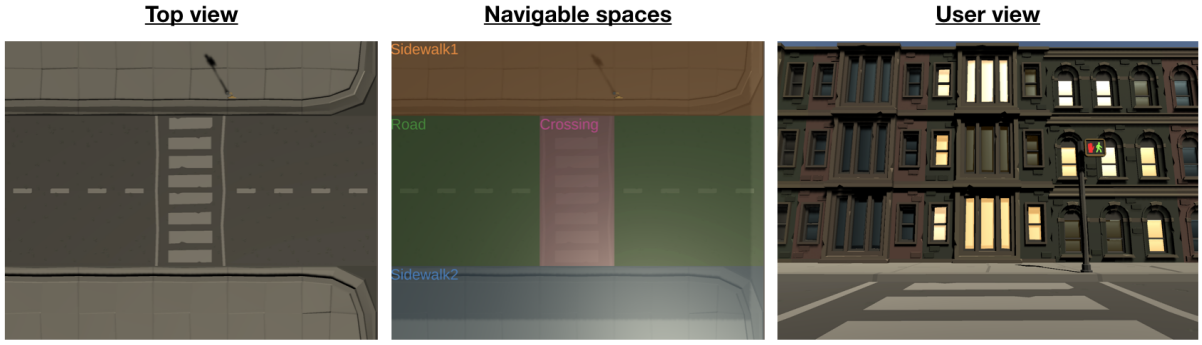

Fig. 14. The second case study features a basic road crossing scene in an open space. The scene has four navigable spaces: a road and a crossing that overlap, and two sidewalks. From left to right, this figure shows the top view, the navigable spaces, and a viewpoint from the user camera.

(4) The user should be on Sidewalk1 at the end:
$Localisation(t_n, User, Sidewalk1)$

This *GUTask* further involves the element of user attention as an active constraint, requiring the user to consciously direct their attention towards the traffic light while approaching the crossing to ensure a safe crossing, which in addition to navigation, involves also an "attention path" of the user's gaze. Thus the validation of this task would involve a navigational and attention path as depicted in Figure 15. During the experience, the camera position and orientation is dynamically evaluated and recorded to ensure that all required constraints for this *GUTask* are met.

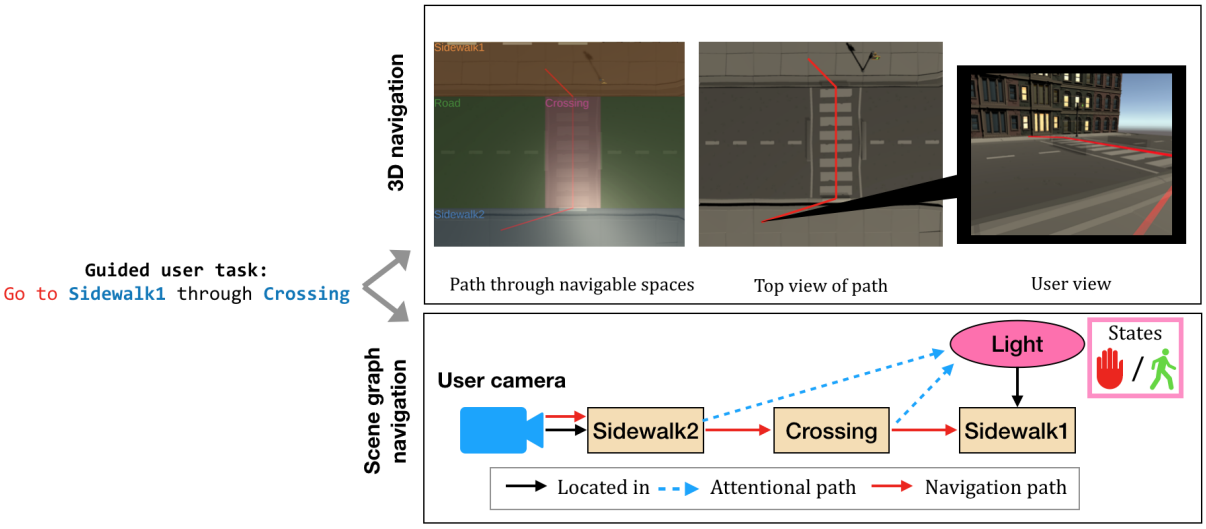

Fig. 15. Given the *GUTask* of crossing the street, the previewed navigation path demands the user to approach the point between `Sidewalk1` and the Crossing. Here however, the user is required to look up and check the status of the traffic light, thus resulting in a guided "Attention Path" that the user needs to fulfill in order to validate the task.

Through generated user logs, further insights into user intention can be drawn, such as whether the user continued to pay attention to the light during the crossing, the variation of walking speed on different navigable spaces (e.g. the user walks faster on the crossing in order to ensure timely arrival on the other sidewalk).

6 EVALUATION

We assessed our framework through a formative evaluation, which was initially used in education for instructional design [5, 7] and later extended and popularized in human-computer interactions by Hix and Hartson [20] to describe evaluations in formative stages of design [32]. As emphasized by Carol et al. [11], formative evaluations provide observations and recommendations that can be immediately used to improve the design of the product or service, and refine the development specifications. For that, a typical formative evaluation addresses questions such as: what are the usability issues in our implementation? Do users understand the workflow? Does the system comply with recognized usability principles? The results are typically qualitative instead of summative, focusing on the needs of the design team including developers, designers, project managers, and other members. Using structured and in-depth interviews, formative evaluations can assess these aspects of our workflow [10].

With this evaluation, our primary aim is to identify gaps between the goals of our framework and our current approach to the system implementation, which is done primarily through feedback from expert interviews, and also collecting specific points to improve on for the implementation before continuing to adapting the system to a professional use case. This formative evaluation allows us to obtain feedback from professional users who would potentially play the role of *Designer* in our workflow, and to improve our system implementation. For that purpose, we have run a series of semi-structured interviews. The case study of the indoor scene (navigation and object interaction) was used as illustrative support.

6.1 Participants

We recruited 6 expert users (3 male, 3 female) from a convenience sample, which is around the recommended number of participants for this type of formative evaluation [21]. In age, 4 participants are between 18-29, 1 participant between 30-39 years old, and 1 participant between 40-49. One of the participants is a Master's student, 3 participants have a Master degree and are currently working in the domain of extended (virtual and augmented) reality, and 2 are currently conducting a PhD in the mixed-reality domain. Overall, the average years of experience in the development of immersive 3D applications is 3 years (min 1, max 7). Concerning the experience with software available in the market, all six participants reported to have experience with Unity, with some having experience in other software including Blender (5), Unreal Engine (4), Maya (1), Voxel (1), iClone (1), Character Creator (1), 3D Exchange (1), and Tinkercad (1).

6.2 Procedure

The interviews were organized in four main steps: presentation of the study, profiling of participants, presentation of our workflow and implementation to develop the indoor scenario, and debriefing. We structured our interviews in a similar way proposed by Burmester et al. [10], by presenting the usage of our application, allowing participants to freely express their positive and negative feedback, and collecting qualitative feedback on specific points to gain concrete advice for future versions. Due to the Covid-19 pandemic, the interviews were performed by videoconference and lasted about an hour. For the purposes of the evaluation, data was collected anonymously and participants have given consent to record the interviews.

The presentation of the steps to develop the indoor scenario was at the core of the interview. At first, we have provided an image of a scene composed of 3 rooms : a kitchen, a bedroom and a garage. Participants were then asked to assess and rate the perceived usability for performing the following steps in the workflow in link with embodiment theory:

- S1 - Scene annotation (Ontology): involves creating new layers, and adding and annotating an object to an existing scene
- S2 - Creating an interactive scenario (Intersubjectivity): involves defining the *GUTasks* that the user should carry out in the scenario (i.e. go to the bedroom, then take the lamp placed on the desk, go to the kitchen, and then place the lamp on the table)
- S3 - Running the scenario (Intersubjectivity): involves the user carrying out the defined *GUTasks* in real-time, with logs of the scene changes and user behavior recorded
- S4 - Analysis of user experience (Intentionality): involves querying and visualizing the log output of the scenario in order to understand the user experience

It is important to note that S1, S2, and S4 correspond to steps that require designer input in Unity using GUsT-3D, while S3 is performed by the end-user (the person experiencing the VR scenario and carrying out tasks in the 3D scene) and is mostly automated for the designer such that they only make observations about how users, in real-time, perform the interactions created in the scene.

We have created videos showing the real use of the tools with detailed information so that participants could identify usability problems in running scenarios. The use of the videos also allowed to perform the evaluation remotely, which was a necessary condition during the lock down period imposed by the Covid-19 pandemic. Given the remote context of the interviews, we could not directly observe the usage of our system implementation. Instead, we played usage videos of the system for each step of the workflow, and we asked the participants to elaborate on what they perceived as positive and negative at each step of the workflow. More specifically, they were prompted to provide detailed comments with respect to: perceived usability, learning curve, efficacy, and flexibility of the system. For steps S1, S2 and S4, we asked participants to provide a score on a 1 (poor) to 5 (very good) Likert scale on their perceived usability, learning curve, efficacy, and flexibility of the workflow.

Additionally they could also respond "I don't know" A video was also played for S3, but since no designer intervention was required, participants were not asked to score this step. Hereafter we summarize the qualitative feedback from expert participants on each step of the workflow.

6.3 Results

In the analysis of results of the interviews, we look at the two parts: scores on various metrics for each step, and the verbal feedback on pros and cons of our workflow. With the selective study pool, our analysis focuses on the qualitative understanding of how our framework and workflow would improve their design process, and the limitations it imposes.

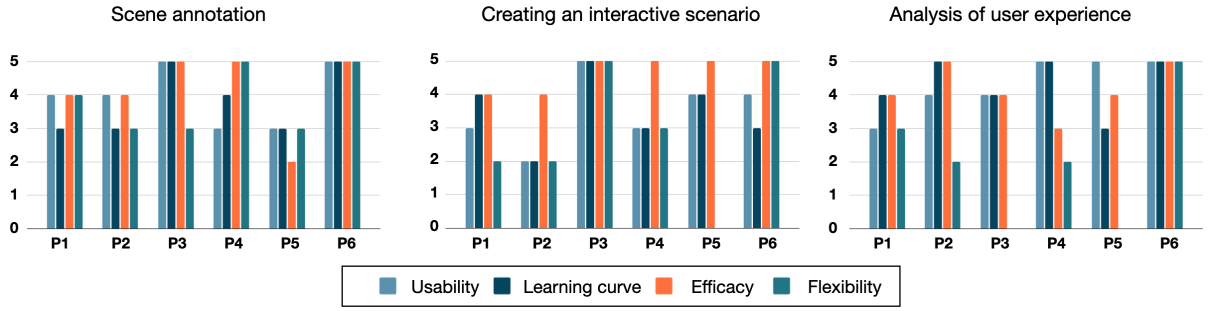

Fig. 16. For the three steps in the workflow, the responses of each participant on the usability, learning curve, efficacy, and flexibility of the implementation. The step "Running the Scenario" does not have these metrics, since it is mainly composed of the automatic processes that launch the *GUTasks* and record logs of the experience. The missing responses indicate where participants responded "I don't know".

An overview of these scored evaluations received from the expert interviews in steps S1, S2, and S4 is summarized in Figure 16, showing the responses of each participant on each metric for the three workflow steps. Overall, the participants found that the workflow implemented in Unity made their design process much more efficient, and on average the responses to the individual steps were all positive, though some more than others. Participants with more experience in 3D development were more critical of the flexibility of the interfaces. The missing responses on flexibility are where the participants responded "I don't know". On further inquiry, they indicated that they felt it depended on the goal and scene they were trying to create. We also observe that the more difficult step of creating and managing an interactive scenario presented a steeper learning curve, but with much improved efficacy in the designer's process.

We also analyzed detailed qualitative feedback from the interviews, as summarized in Table 2. Overall, participants provided more positive than negative comments for the general workflow. They also provided many suggestions for improving the usability and the user interface in all the workflow steps. A number of excerpts of the interviews are particularly illustrative of the type of feedback we received for different steps. On S1, suggestions from participants P1, P2, P4 and P5 mention that the interface should *"Centralize different features in a single window"*. P6 who works in the domain of music, notably appreciated how the annotations with the GUsT-3D ontology *"can be used with any 3D scene or environment"*, but also mentioned their needs for annotating sound in their applications as well as for usage in other game engines such as Unreal, which are currently not supported by our implementation. On step S2, participants commented that *"The labels of the UI tools could be more easy to understand"*. P3 commented on S3 *"Display tasks with color depending on the guidance type, task goals, and times would help to better see the tasks"*. Another interesting comment on S3 from P4 was *"I don't need to code*

Table 2. Participants were asked to elaborate the positive and negative points of each step of the workflow and its implementation. Here we summarize the qualitative feedback from expert participants on each step, with the IDs of the participants that provided each point in parenthesis.

Workflow step	Positive feedback	Negative feedback
S1 Scene annotation *(Ontology)*	- Time gain (P1, P2) - Convenient for non-developer (P2, 3) - Allows real-time testing (P2) - Easy to learn (P1, 3, 4, 6) - Any new object can be added (P2, 5) - Has properties and pathfinding on objects (P4,5,6) - Ontology can be used with any 3D scene (P6)	- Unclear path indications (P4, 5, 6) - Complex syntax for query language (P2) - Too many separate windows (P1, 2, 4, 5) - No way to add and annotate sound properties (P6) - Can't be used with other software like Unreal (P6)
S2 Creating an interactive scenario *(Intersubjectivity)*	- Easy to learn (P1, 2, 3, 5, 6) - Time gain (P2, 3, 4) - Convenient for non-developer (P2, 3) - Easy creation of task list (P4) - Good management of user guidance (P5) - Recap of all tasks is easy to understand (P1) - All information in one place (P1)	- Limited to tasks available in the tools (P2, 5) - No parallel tasks (P5) - Names and labels in UI are not clear (P1, 4, 6)
S3 Run task scenario *(Intersubjectivity)*	- Easy to learn (P1, 2) - Provides a color-coded timer (P2) - Provides useful indications to the user (P3, 5)	- Unclear object names (P1, 4) - Text on the screen is hard to read (P3, 6) - Don't know how to interact with objects (P4) - Cannot change UI display (P2) - Current time and time unit is not displayed (P1)
S4 Analysis of user experience *(Intentionality)*	- Scene graph is easy to understand (P4) - Displays information over time (P3, 5, 6) - Information is useful for various analyses (P5, 6) - Shows interactions between user and scene (P4) - Easy to learn (P2) - Scene graph generation is automatic (P1, 2)	- Limited visualization methods (P3, 4, 6) - Lack of a global view (P4) - Hard to read when there are too many items (P2, 5) - UI lacks auto-complete/autoplay (P1, 4) - Cannot export scene graph as an image (P3) - Scene graph elements cannot be customized (P2, 5)

just click and select, and progress is automatically saved", which is a convenient feature of the implementation. As for the step S4, the participant P5 put himself/herself in the shoes of the end-user and suggested that the designer should *"Ask the user how much guidance they want, and when'* , which definitely would help customize the guidance to be provided by *GUTasks* specified in the GUsT-3D representation, and thus improve the user experience. In addition to the suggestions, many of the negative comments focused on properties of the user interface design, which would be easy to change. For example, in S4 the comment *"Cannot export scene graph as an image"*, while we do not yet have this functionality, the .dot file used for the scene graph is a well-known format that can be easily exported into an image file. The main negative comment associated with S4 concerns the analysis of user interaction. Participants P3, P4, and P6 mentioned that the interface only provided basic visualizations, where for example, P6 suggested navigating the data not just on a temporal slider, but also spatially

to see the state and movement changes for each navigable space. Participant P4 also mentioned the system's *"Lack of global view"*. These comments suggest the strong interest and current needs for robust and flexible solutions on analyzing and visualizing user experience metrics – an ongoing and important challenge. Another positive remark from P1 and P2 was the workflow's potential to *"Record and generate scene graphs automatically"*.

The majority of the negative feedback focusing on the user interface, and not on the framework itself. The results of such as formative evaluation confirm the needs of Designers for an integrated workflow for the creation and management of interactive user experiences in VR, demonstrates the added-value of the framework, and also provides valuable suggestions for improvements. We can thus confidently seek the deployment of our framework in practice after addressing usability issues.

The results of the formative evaluation have thus evaluated the anticipated use of our developed tools and workflow on measures of usability, learnability, efficacy, and flexibility. Whilst the results are quite positive with respect to how users perceive the system, only by deploying the system to a real use case can we evaluate the real usability (e.g., performance with specific tasks) and user experience, which were not addressed in this formative study.

7 CONCLUSION

In this paper we have presented a novel framework based on Dourish's embodiment theory that presents multiple capabilities to design, run, and analyze embodied experiences in VR. This involves notably the development of the GUsT-3D representation to establish scene ontology, the *Guided User Tasks (GUTask)* definitions to create and run user tasks, and tools for the query, analysis, and visualization of scene and user behavior evolution. The framework is implemented and we assessed its usage through two case studies and a formative evaluation involving 6 structured expert interviews. Results show that the GUsT-3D framework and corresponding workflow implementation integrate well into the Designer's process to create, manage, and analyze VR embodied experiences of target users.

We are planning a number of future directions for this work. To improve the system flexibility, we are looking into:

- the creation and loading of a custom ontology that can change and adapt the user experience in the 3D scenario,
- the definition of constraints on ranges of values or probabilistic models to represent uncertainty to support the design of more diverse or complex *GUTasks*, and
- providing flexible query compositions to reason on varied elements of user behavior and perception in the post-scenario analysis, with adapted visualizations that can better help understand usability and user experience as a whole [3].

In addition, we believe the scene graph generation using the ontology opens exciting pathways to support model-based evaluations of embodied VR scenarios. The feedback from the expert interviews in the formative evaluation have provided valuable insights to challenges towards designing visualization and analysis tools to better understand the embodied user experience. In the longer term, we can envision this work in the context of managing interactive tasks for multi-user scenarios, or in multi-agent systems. Finally, a direct step already in motion is the deployment of this framework to a practical use case, which will allow us to observe how the workflow performs with expert designers. This is a current and strong need in areas such as neuroscience and cognitive science, to create serious games for rehabilitation and evaluate training efficacy.

REFERENCES

[1] Abdul-Hadi G Abulrub, Alex N Attridge, and Mark A Williams. 2011. Virtual reality in engineering education: The future of creative learning. In 2011 IEEE global engineering education conference (EDUCON). IEEE, Amman, Jordan, 751–757.

[2] Johannes Behr, Patrick Dähne, and Marcus Roth. 2004. Utilizing X3D for Immersive Environments. In Proceedings of the Ninth International Conference on 3D Web Technology (Monterey, California) (Web3D '04). Association for Computing Machinery, New York, NY, USA, 71–78. https://doi.org/10.1145/985040.985051

[3] Regina Bernhaupt, Célia Martinie, Philippe Palanque, and Günter Wallner. 2020. A Generic Visualization Approach Supporting Task-Based Evaluation of Usability and User Experience. In 8th International Conference on Human-Centered Software Engineering - IFIP WG 13.2 International Working Conference, HCSE 2020 (Lecture Notes in Computer Science book series (LNCS), Vol. 12481), IFIP : International Federation for Information Processing (Ed.). Springer, Eindhoven, Netherlands, 24–44. https://doi.org/10.1007/978-3-030-64266-2_2

[4] Daniel Beßler, Robert Porzel, Mihai Pomarlan, Michael Beetz, Rainer Malaka, and John Bateman. 2020. A Formal Model of Affordances for Flexible Robotic Task Execution. In ECAI 2020. IOS Press, Santiago de Compostela, Spain, 2425–2432.

[5] Paul Black and Dylan Wiliam. 2009. Developing the theory of formative assessment. Educational Assessment, Evaluation and Accountability (formerly: Journal of Personnel Evaluation in Education) 21, 1 (2009), 5–31.

[6] Sebastian Blumenthal and Herman Bruyninckx. 2014. Towards a Domain Specific Language for a Scene Graph based Robotic World Model. arXiv:1408.0200 [cs.RO]

[7] Carol Boston. 2002. The concept of formative assessment. Practical Assessment, Research, and Evaluation 8, 1 (2002), 9.

[8] Don Brutzman and Leonard Daly. 2010. X3D: extensible 3D graphics for Web authors. Elsevier, San Francisco, CA, USA.

[9] Michel Buffa and J-C Lafon. 2000. 3D virtual warehouse on the WEB. In 2000 IEEE Conference on Information Visualization. An International Conference on Computer Visualization and Graphics. IEEE, London, UK, 479–484.

[10] Michael Burmester, Marcus Mast, Kilian Jäger, and Hendrik Homans. 2010. Valence method for formative evaluation of user experience. In Proceedings of the 8th ACM conference on Designing Interactive Systems. ACM, Aarhus, Denmark, 364–367.

[11] John Carroll, MARK SINGLEY, and Mary Beth Rosson. 1992. Integrating theory development with design evaluation. Behaviour & Information Technology - Behaviour & IT 11 (Sept. 1992), 247–255. https://doi.org/10.1080/01449299208924345

[12] Tiannan Chen and Stephen Guy. 2018. GIGL: A domain specific language for procedural content generation with grammatical representations. In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, Vol. 14(1). AAAI, Edmonton, Canada, 8 pages.

[13] Pierre Chevaillier, Thanh-Hai Trinh, Mukesh Barange, Pierre De Loor, Frédéric Devillers, Julien Soler, and Ronan Querrec. 2012. Semantic modeling of virtual environments using mascaret. In 2012 5th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS). IEEE, Costa Mesa, CA, USA, 1–8.

[14] Paul Dourish. 2004. Where the action is: the foundations of embodied interaction. MIT press, USA.

[15] Jakub Flotyński and Paweł Sobociński. 2018. Logging Interactions in explorable immersive VR/AR applications. In 2018 International Conference on 3D Immersion (IC3D). IEEE, Brussels, Belgium, 1–8.

[16] Jakub Flotyński and Krzysztof Walczak. 2017. Ontology-Based Representation and Modelling of Synthetic 3D Content: A State-of-the-Art Review. In Computer Graphics Forum, Vol. 36(8). Wiley Online Library, Poznań, Poland, 329–353.

[17] Daniel J Fremont, Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Alberto L Sangiovanni-Vincentelli, and Sanjit A Seshia. 2019. Scenic: a language for scenario specification and scene generation. In Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation. Association for Computing Machinery, New York, NY, USA, 63–78.

[18] Filip Górski. 2017. Building virtual reality applications for engineering with knowledge-based approach. Management and Production Engineering Review 8(4) (2017), 63–73. https://doi.org/10.1515/mper-2017-0037

[19] Sebastian Gottschalk, Enes Yigitbas, Eugen Schmidt, and Gregor Engels. 2020. Model-based product configuration in augmented reality applications. In International Conference on Human-Centred Software Engineering. Springer, Eindhoven, The Netherlands, 84–104.

[20] Deborah Hix and H. Rex Hartson. 1993. Developing user interfaces : ensuring usability through product & process. New York : J. Wiley. http://archive.org/details/developinguserin00hixd

[21] Harry Hochheiser Jonathan Lazar, Jinjuan Feng. 2010. Research Methods in Human-Computer Interaction - 2nd Edition. Wiley, United States.

[22] Wanwan Li, Javier Talavera, Amilcar Gomez Samayoa, Jyh-Ming Lien, and Lap-Fai Yu. 2020. Automatic Synthesis of Virtual Wheelchair Training Scenarios. In 2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR). IEEE, Atlanta, GA, USA, 539–547.

[23] Yunchao Liu, Jiajun Wu, Zheng Wu, Daniel Ritchie, William T. Freeman, and Joshua B. Tenenbaum. 2019. Learning to Describe Scenes with Programs. In International Conference on Learning Representations. ICLR, New Orleans, USA, 13 pages. https://openreview.net/forum?id=SyNPk2R9K7

[24] Erik Meijer, Brian Beckman, and Gavin Bierman. 2006. Linq: reconciling object, relations and xml in the. net framework. In Proceedings of the 2006 ACM SIGMOD international conference on Management of data. ACM, NY, New York, 706–706.

[25] David Navarre, Philippe A. Palanque, Rémi Bastide, Amélie Schyn, Marco Winckler, Luciana Porcher Nedel, and Carla Maria Dal Sasso Freitas. 2005. A Formal Description of Multimodal Interaction Techniques for Immersive Virtual Reality Applications. In Human-Computer Interaction - INTERACT 2005, IFIP TC13 International Conference (Lecture Notes in Computer Science, Vol. 3585), Maria Francesca Costabile and Fabio Paternò (Eds.). Springer, Rome, Italy, 170–183. https://doi.org/10.1007/11555261_17

[26] Matthias Oberhauser and Daniel Dreyer. 2017. A virtual reality flight simulator for human factors engineering. Cognition, Technology & Work 19, 2 (2017), 263–277.
[27] Fabio Pittarello and Alessandro De Faveri. 2006. Semantic description of 3D environments: a proposal based on web standards. In Proceedings of the eleventh international conference on 3D web technology. ACM, New York, NY, USA, 85–95.
[28] Thibault Raffaillac and Stéphane Huot. 2019. Polyphony: Programming Interfaces and Interactions with the Entity-Component-System Model. Proceedings of the ACM on Human-Computer Interaction 3, EICS (2019), 1–22.
[29] Rafael Sacks, Jennifer Whyte, Dana Swissa, Gabriel Raviv, Wei Zhou, and Aviad Shapira. 2015. Safety by design: dialogues between designers and builders using virtual reality. Construction Management and Economics 33, 1 (2015), 55–72.
[30] Julien Saunier, Mukesh Barange, Bernard Blandin, Ronan Querrec, and Joanna Taoum. 2016. Designing adaptable virtual reality learning environments. In Proceedings of the 2016 Virtual Reality International Conference. ACM, New York, NY, USA, 1–4.
[31] Maximilian Speicher, Katy Lewis, and Michael Nebeling. 2021. Designers, the Stage Is Yours! Medium-Fidelity Prototyping of Augmented & Virtual Reality Interfaces with 360theater. Proceedings of ACM Human-Computer Interactions 5, EICS, Article 205 (May 2021), 25 pages. https://doi.org/10.1145/3461727
[32] Sandra Trösterer, Elke Beck, Fabiano Dalpiaz, Elda Paja, Paolo Giorgini, and Manfred Tscheligi. 2012. Formative user-centered evaluation of security modeling: Results from a case study. International Journal of Secure Software Engineering (IJSSE) 3, 1 (2012), 1–19.
[33] Lode Vanacken, Chris Raymaekers, and Karin Coninx. 2007. Introducing semantic information during conceptual modelling of interaction for virtual environments. In Proceedings of the 2007 workshop on Multimodal interfaces in semantic interaction. ACM, New York, NY, USA, 17–24.
[34] Diego Vergara, Manuel Pablo Rubio, and Miguel Lorenzo. 2017. On the design of virtual reality learning environments in engineering. Multimodal technologies and interaction 1, 2 (2017), 11.
[35] Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X Chang, and Daniel Ritchie. 2019. Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. ACM Transactions on Graphics (TOG) 38, 4 (2019), 1–15. Publisher: ACM New York, NY, USA.
[36] Haojie Wu, Daniel H Ashmead, Haley Adams, and Bobby Bodenheimer. 2018. Using Virtual Reality to Assess the Street Crossing Behavior of Pedestrians with Simulated Macular Degeneration at a Roundabout. Frontiers in ICT 5 (2018), 27.
[37] Hui-Yin Wu, Aurélie Calabrèse, and Pierre Kornprobst. 2021. Towards accessible news reading design in virtual reality for low vision. Multimedia Tools and Applications 80 (2021), 1–20.