Journal of Data Intelligence, Vol. 2, No. 2 (2021) 166–189

AN APPROACH TO DEVELOP MOBILE PROXEMIC APPLICATIONS

PAULO PEREZ PHILIPPE ROOSE
University of Pau, LIUPPA – T2I64600, Anglet, France
{paulo.perez-daza,philippe.roose}@univ-pau.fr

YUDITH CARDINALE
Universidad Simón Bolívar, Caracas, Venezuela
Universidad Católica San Pablo, Arequipa Perú
ycardinale@usb.ve

MARK DALMAU
E2S / University of Pau, LIUPPA – T2I64600, Anglet, France
dalmau@iutbayonne.univ-pau.fr

DOMINIQUE MASSON
Technopole Izarbel Dev1-0, Bidart, France
d.masson@dev1-0.com

NADINE COUTURE
University of Bordeaux, ESTIA Institute of Technology, Bidart, France
n.couture@estia.fr

Traditional Human-Computer Interaction (HCI) is being overpowered by the widespread diffusion of smart and mobile devices. Currently, smart environments involve daily day activities covered by a huge variety of applications, which demand new HCI approaches. In this context, proxemic interaction, derived from the proxemic theory, becomes an influential approach to implement new kind of Mobile Human-Computer Interaction (MobileHCI) in smart environments. It is based on five proxemic dimensions: Distance, Identity, Location, Movement, and Orientation (DILMO). However, there is a lack of general and flexible tools and utilities focused on supporting the development of mobile proxemic applications. To respond to this need, we have previously proposed a framework for the design and implementation of proxemic applications for smart environments, whose devices interactions are defined in terms of DILMO dimensions. In this work, we extend this framework by integrating a Domain Specif Language (DSL) to support the designing phase. The framework also provides an API, that allows developers to simplify the process of proxemic information sensing (i.e., detection of DILMO dimensions) with mobile phones and wearable sensors. We perform an exhaustive revision of relevant and recent studies and describe in detail all components of our framework.

Keywords: Domain specific language, proxemic interaction, proxemic zone, dilmo, mobile devices, graphical modelling

1. Introduction

Nowadays, the use of mobile technologies in our daily life is very common. People can interact with different contexts through electronic devices (e.g., personal mobile phones, tablets, wearable technologies, and smart-watches) to accomplish their daily tasks. Many of these tasks require a specific Human-Computer Interaction (HCI). Researchers are therefore seeking to develop new useful and enjoyable interfaces. Proxemic interaction arises as a novel concept

to improve HCI[1, 2]. Proxemic interaction describes how people use interpersonal distances to interact with digital devices[3, 4, 5], using five physical proxemic dimensions: Distance, Identity, Location, Movement, and Orientation (DILMO).

Proxemic interaction is derived from the social proxemic theory proposed in 1966 by the anthropologist Edward T. Hall[6]. Hall describes how individuals perceive their personal space relative to the distance between themselves and others. According to Hall's proxemic theory, interaction zones have been classified into four zones: (i) intimate zone, comprised between 0 and 50 cm of distance; (ii) personal zone, defined by a distance of 50 cm to 1 m; (iii) social zone, when the distance is between 1 m and 4 m; and (iv) public zone, for distances of more than 4 m. He underlines the role of proxemic relationships as a method of communication based on the distance among people. This theory has been applied to define relation and communication among people and digital devices[4].

In this context, solutions such as Toolkit[7] and ProximiThings [8](for proxemic interaction in the Internet of Things) have been proposed to support the development of proxemic interaction. However, existing tools and frameworks present limitations for implementing proxemic interaction in mobile technologies because they require special hardware devices connected to the system (e.g., a Kinect Depth sensor, which must be installed on a PC for sensing proxemic information).

Nowadays, the vast majority of smartphones and mobile devices are equipped with powerful hardware capabilities. These capabilities allow devices to process and obtain proxemic information; for example by using sensors and cameras embedded in a smartphone. In turn, it is possible to implement proxemic based mobile applications that facilitate users' contact and interaction with other people and devices in indoor and outdoor spaces, which we call smart proxemic environments. This fact, combined with the current trend of using proxemic interaction to improve HCI, has raised the need for frameworks and tools to support the development of such mobile proxemic applications.

In a previous work, we have proposed a framework[9] for the design and implementation of mobile proxemic applications, comprised by entities, whose interactions are defined according to DILMO dimensions. Our framework offers a process to define and manage all components in a proxemic environment: the interaction objects, the DILMO dimensions that govern the HCI, and the proxemic mobile applications. In this work, we extend this framework by integrating a graphical Domain Specific Language (DSL) to support the designing phase [10]. An API is integrated into the framework, that allows developers to simplify the process of proxemic information sensing (i.e., measure of DILMO dimensions) by mobile phones and wearable sensors. In this work, we perform an exhaustive revision of relevant and recent studies, to show how they cover the design and implementation phases of the development of proxemic applications. We describe in detail all components of our framework. We developed two mobile apps as proof-of-concept to demonstrate the suitability of our framework. These two mobile apps are based on HCI defined as a function of different DILMO combinations that specify different context-based infrastructures for proxemic environments based on mobile devices.

The remainder of this work is organized as follows. In Section , we outline related work on proxemic interaction. Section presents our definitions for designing proxemic environments based on a graphical DSL, followed by a description of the framework in Section . The

details of proof-of-concept and applications are presented in Section . Finally, we conclude and outline our future work in Section .

2. Related work

Proxemic concepts are based on physical, social, and cultural factors that influence and regulate interpersonal interactions [7]. In order to know how the factors should be applied to proxemic interactions for ubiquitous computing applications, Greenberg et al. [4] identified five dimensions: Distance, Identity, Location, Movement, and Orientation (we call them DILMO as an abbreviation), which are associated with people, digital devices, and non digital things. In this section, we review prior works on proxemic interaction and how they have been implemented. Afterward, we analyze the existing technical methods for the development of proxemic applications and compare them with our framework.

2.1. *Applications based on Proxemic Interactions*

There exist a variety of works that implement interactive ubiquitous applications. The common aspect to all these applications is the use of all or a subset of proxemic dimensions (i.e., DILMO). These dimensions allow applications to know absolute and relative positions of people and objects in the physical space. In this section, we describe the concept of each proxemic dimension and how they have been used by prior works.

Distance is a physical measure of separation between two entities, according to how they interact[11]. Typically, short distances allow high interactions, while long distances allow little to no interaction. For example, in [2, 7, 12, 13, 14, 15, 16], the distance is used as a parameter to assign a proxemic zone that allows the users to interact with the display or devices in different proximities. The interaction zones are also used for adapting visualizations on displays based on the users' distance relative to the screen, such as the studies presented in [17, 18, 19].

Distance can be obtained by using different techniques based on a variety of sensors to capture their values. Bluetooth Low Energy (BLE) technologies allow the device to estimate the proximity among entities by Received Signal Strength Indicator (RSSI) and Broadcasting Power value (TX power), as in [20, 21]. The work presented in [22] uses a smartphone with BLE technology in order to obtain proximity between blind persons and fixed objects. The work presented in [15], proposes the use of the body-tracking capabilities of Kinect Sensors to obtain the distance. Authors demonstrate the suitability of their proposal in an application that measures the distance between blind people and paintings, according to which it provides different background music experiences. In [17, 23], computer vision is used for measuring the distance from the device hosting the program to the user.

Identity is a term that mainly describes the individuality or role of a person or a particular object in a space [3, 7]. SpiderEyes [13] is a collaborative proxemic system that helps designers to create applications by tracking multiple people interacting in front of a display in run-time. In this particular regard, it is indispensable to have the user identification. User's interaction is based on their identity and distances with the display. The system is able to detect when users leave the field of view of the display and if they later rejoin the field of view at a different distance. With SpiderEyes, authors demonstrate the effectivity of the identification system with up to four users at the same time. This work uses a visual monitoring tool (called

Microsoft Kinect Depth Camera), that allows developers to classify which entities are being tracked and how the details of information are stored in the application for creating identities. FAMA (First Aid Mobile Application) [24] is a mobile app based on proxemic interactions that offers rescuers to obtain emergency identification (identity) of an unconscious person, as the rescuers are moving toward the injured person's proxemic zones. FAMA uses Beacon BLE technology for the identification of entities (injured people). Proxemic-Aware Controls system [25] uses identity for controlling spatial interactions between a person's handheld device and all contiguous appliances to generate an effective appliance control interface. These applications have demonstrated that identity can be used to know individuality of a person or an object.

Location describes qualitative aspects of the space, where there is interaction among fixed entities (e.g., room layout, doors) or semi-fixed entities (e.g., furniture positions) [14, 25]. Researchers have also considered the location for obtaining the user's current positions. Multi-Room Music System [26] is based on proxemic interactions that allow the user to hear the same songs playlist, while he changes his location around the house through the speakers that have been installed in the rooms. In the work presented in [15], location is used to detect events related to hands' tracking. For example, when a blind user explores a painting with his hands, the application uses the 3D coordinate system of the Microsoft Kinect Camera [27], to know the user's hand position in a specific region of the painting. In [2, 7], entities are associated with three-dimensional positions related to a fixed point that can be used for initial setting of smart environments. In such a way, it is possible to obtain the relative position among people and devices.

Movement is defined as changes of position and orientation of an entity over the time [3]. This is the case when a user walks in front of a screen or approaches it, and the content of the screen is adjusted according to the user's movements. This kind of motion can be captured by motion technology [2, 28]. Velocity changes are calculated in order to respond to the user's behavior. The movement also allow gesture recognition through smartphones or wearable technologies by employing motion sensors [8]. FAMA, a first aid mobile app, identifies potential rescuers as they move towards the injured person's proxemic zones [24].

Orientation provides the information related to the direction in which an entity is facing. It can identify the front of an entity (e.g., person's eyes, screen front). Previous work have demonstrated how a person's orientation related to a display can be used for improving user interaction [2, 3, 11, 29]. The study presented in [30], describes the use of built-in compass in mobile devices to support the process of pairing them based on the orientation. Another remarkable work is Multi-View Proxemic system [31], which considers distinct views from a single display related to the angle of orientation of two viewers. This work uses gaze detection technology that allows the active user to be identified.

We have briefly described studies that have implemented multiple proxemic applications based on DILMO. In all of these applications, the interaction objects (i.e., people and devices) are considered as entities. In other studies, entities have been implicitly managed. For example in [12], Distance, Movement, and Location dimensions have been used to implement interaction between *a user* and *a screen*; the *user* and the *screen* are (non identifiable) entities. The work presented in [31] detects the Distance and Orientation of *a user* with respect to a *single display* to generate multiple views of the information displayed. Similarly, the

applications described in [14, 15] help *visually impaired people* to explore paintings based on Distance, Movement, and Location. Thus, we conclude that depending on the application, all DILMO dimensions are not required and specific combination of them can determine different proxemic environments. Table 1 summarizes the sensors used by previous proposals to obtain proxemic information.

Table 1. List of sensors for obtaining DILMO proxemic dimensions used by previous proposals

Sensor	D	I	L	M	O
Microsoft kinect [3, 7, 11, 12, 13, 15, 16, 19, 23, 29]	✔	✔	✔	✔	✔
Vicon/OptiTrac Motion Capture [3, 11, 18, 25, 29, 31]	✔		✔	✔	✔
Leap Motion [12]				✔	
LV-MaxSonar-EZ1 [14]	✔				
SHARP GP2Y0A02YK0F [32]	✔				
Mobile Sensors (accelerometer, gyroscope, magnetometer) [2, 8, 30]			✔	✔	✔
Bluetooth BLE [2, 20, 21, 22, 24]	✔	✔			
Wi-fi RSSI signal [26]	✔	✔	✔		
Mobile computer vision [17]	✔	✔			

Computer vision is frequently employed to obtain almost the whole DILMO proxemic dimensions using a Kinect depth camera. Kinect depth camera is powerful and low cost sensor that is frequently used in previous proposal for sensing proxemic interaction. Vicon motion capture technology is very accurate in seizing people's movement and objects in which users have to carry and wear Vicon marker sensors. Notwithstanding, these sensors do not provide portability that allow developers to implement in different mobile devices.

In the next section, we present some works that have proposed tools and frameworks to support the development of applications based on proxemic interactions. We highlight their limitations and how we overcome them.

2.2. *Tools to design and develop Proxemic-based Applications*

Software design is the process to convert the user requirements into some suitable form, which supports the developer in software coding and implementation. Some existing tools for the development of proxemic applications, propose approaches to support the designing phase. SpiderEyes [13] is a system that offers a visual tool that allows designers to create collaborative proxemic applications for Wall-Sized Dislays. This work develops proxemic applications for adjusting visualizations on displays. The visualization design tool is a web application that requires setting the values to distinguish active users passing by in the background and foreground based on whether their visual attention is on display or not and only displays visualizations for active users. This system shows how people's orientation and distance to the display can support collaborative activities around large wall-sized displays. Nevertheless, it does not propose a graphical design to enable end-users the general modelling of proxemic behavior.

Ministudio [33] is a tool for designers without technical implementation skills, which can be used to build miniature prototypes for proxemic interactions in the design phase. This

tool illustrates how the proxemic dimensions can be used to define spatial relationships and interactions among people, devices, and objects using designers' familiar software and materials. Ministudio is based on tangible interaction interface, which uses miniature models on paper and projected images. This work shows that MiniStudio supports rapid designing of large and complex ideas with multiple connected components for prototyping smart environments in the design phase. However, the implementation of the tool needs specific hardware and material for creating miniature models, such as Augmented Reality marker and camera-projector and it does not offer a graphical notation to allow the designer to model proxemic behaviors. Therefore, this tool is limited to be implemented on a large scale.

In order to build proxemic environments with mobile devices, we implemented a graphical notation, described in a previous work [10]. This work provides graphical symbols that allows representing proxemic environments conformed by the proxemic zones, entities, and their DILMO dimensions. Each symbol can be described as an entity in the environment. The graphical model is part of a DSL that allows non-specialist designers to describe the proxemic interaction among physical user and devices using DILMO. The graphical DSL is proposed to help developers to understand and implement the proxemic systems.

Table 2 summarizes the current tools for supporting the design of proxemic environments. Specific hardware requirements limit the use of SpiderEyes and Ministudio for the design of mobile proxemic applications, meanwhile the DSL is more appropriate and suitable for every kind of proxemic applications.

Table 2. Tools for design proxemic enviroments

Design tool	Graphical notation	Proxemic modelling	Hardware requirement
SpiderEyes [13]	no	yes	yes
Ministudio [33]	yes	yes	yes
DSL [10]	yes	yes	no

Some frameworks have been proposed to support the development process and complement the software engineering [7, 34]. In [7], a framework called Proximity Toolkit, used to discover novel proxemic-aware interaction techniques is proposed. The framework is a guide on how to apply proxemic interaction for domestic ubiquitous computing environments. It is used to discover novel proxemic aware interaction techniques. The framework manages DILMO dimensions, which allow determining relationships between entities and people. The framework architecture has four main components: (a) a Proximity Toolkit server; (b) a Tracking pug-in; (c) a Visual Monitoring tool; and (d) an API. The Proximity Toolkit server is the central component in the distributed client-server architecture which allows multiple client devices to access the captured proxemic information. The Tracking plug-in contains two plugins: the marker based VICON and the KINECT sensor, that track the skeletal bodies and stream the raw input data of tracked entities to the server. The Visual Monitoring tool permits developers and designers to visualize tracked entities and their proxemic relationships among digital devices. The API is a collection of libraries developed in C that use spatial information and relations among objects and space for processing proxemic information from ubicomp environments. This framework allows rapid prototyping of innovative

interfaces processing proxemic information from sensors. However, the implementation of this framework requires a hardware architecture based on fixed devices (e.g., a Kinect Depth sensor and a client-server architecture) for allowing the server to process the proxemic information from appliances. This solution does not offer the mobility and portability required for implementing proxemic interactions on mobile devices or wearable technologies [8]. With the current Proximity Toolkit version, it is not possible to obtain proxemic information from the new smartphone's sensors capabilities and ensuring that users can use proxemic mobile applications on their smartphones in any place.

The work presented in [34], illustrates how the proxemic dimensions can support interaction among entities (people or objects), with a proposed context-aware framework. This framework uses mobile sensors that provide portability for sensing DILMO proxemic dimensions. However, this framework needs an active internet connection to process proxemic information from mobile sensors. This framework does not offer methods that allow developers to implement proxemic interactions based on mobile computer vision. Furthermore, neither of the two current frameworks provides an API that enables developers to process proxemic information from mobile devices sensors. Both frameworks are limited to built proxemic mobile applications, while ProximiThings [34] cannot process DILMO dimensions without a server connection. Table 3 describes constraints for building proxemic mobile applications with these two proposals.

Table 3. Proxemic frameworks constraints

Framework	Portability	Offline service	Mobile API
Toolkit [7]	Low	Yes	No
ProximiThings [34]	High	No	No

All these works demonstrate the current interest for researchers to develop tools that support the design and implementation of proxemic applications. Nowadays, smartphones and mobile technologies are powerful and offer a wide range of possibilities to improve user interaction. There are about 2 billion people with smartphones, which represents one-quarter of the global population in the world, and this trend is on the rise [35]. It is therefore of great importance to provide tools for developing mobile apps and improve HCI on mobile devices.

3. Proxemic Definitions

In this section, we present the definitions of proxemic environment and its components (i.e., entities, DILMO dimensions, and proxemic zones) that we have adapted from the proxemic studies.

- **Entity (E).** *An entity, denoted as E, represents an interaction object (e.g., a person, an object, a device), that can be univocally identified or not in a physical space.*
- **Distance (D).** *The distance, denoted as D, is the physical measure of proximity among mobile or fixed entities (E) in a physical space.*
- **Identity (I).** *The Identity, denoted as I, represents an entity (E) that has a unique identification or a specific role in a physical space.*

- **Location (L).** *The location, denoted as L, is a relative position or an absolute position of an entity (E) in a physical space.*

- **Movement (M).** *The movement, denoted as M, represents the change in measures of distance or position of an entity (E), over an interval of time in a physical space.*

- **Orientation (O).** *The orientation, denoted as O, represents the face to face alignment between two entities in a physical space.*

- **Cyber Physical System (CPS).** *A cyber physical system, denoted as CPS, represents the target entity, from which a proxemic environment (P_E) is defined. All proxemic zones (P_Z) and DILMO dimensions are measured for all other entities (E) with respect to the CPS, and the interaction among them will be defined accordingly.*

- **Proxemic Zone (P_Z).** *A proxemic zone represents a circular zone delimited by a maximum distance (radio) with respect to the CPS. There are four proxemic zones defined according to such maximum distance: $P\_Z_{intimate}$, $P\_Z_{personal}$, P_Z_{social}, and P_Z_{public}.*

- **Proxemic Environment (P_E).** *A proxemic environment, denoted as P_E, represents a set of sensors and devices attached to entities (E) that can in turn interact according to D, I, L, M, and O. It is denoted as a four-tuple, such as:*

  $P\_E = < CPS, entities, identities, distances >$

  where:

  - CPS is the target entity;

  - *entities* is a set of general interaction objects;

  - *identities* is a set of identifiable interaction objects; and

  - *distances* defines the distances that define the four proxemic zones.

- **Behaviour (B).** *A behaviour represents the change of D, L, M, and O measures or the P_Z of an entity (E or I), starting from its initial behaviour (B_0). It is denoted as a tuple $E.B_i =< E.D_i, E.L_i, E.M_i, E.O_i, E.P\_Z_i >$ (as identity of I will not change, this notation is valid for I) and is produced by a transition from the previous behaviour B_{i-1}, such that:*

  - $E.B_0 =< E.D_0, E.L_0, E.M_0, E.O_0, E.P\_Z_0 >$ *(initial behaviour);*

  - $E.B_i = \nabla E.B_{i-1} =< E.D_i, E.L_i, E.M_i, E.O_i, E.P\_Z_i >$ *(for $i \geq 1$);*

  where ∇ is a user-defined function to detect the change of measures of DLMO dimensions.

- **Action (*Action*).** *An action, denoted as Action, represents an event performed by the CPS or any other entity (E or I) in a P_E, in response to a specific behaviour (B) of an entity or group of entities.*

In this work, we have defined four proxemic zones, such as $P\_Z_{intimate}$, $P\_Z_{personal}$, P_Z_{social}, and P_Z_{public}. The P_Z's distance is defined by developers according to user requirements.

A P_E represents the system based on proxemic behaviors. The change of DILMO measures of E in a P_E, defines their behaviours. Behaviours of entities might be defined by developers through user-defined ∇ functions. For example, a ∇ function to detect movement of an entity, can be implemented based on readings from an accelerometer sensor of a smartphone. Actually, when an entity's movement is detected, the rest of DLO measures can be affected. These behaviors should be specified by developers. Hence, according to behaviours of entities, E or I, in a P_E, they can react by performing actions to comply their functionalities. It is expected that the CPS in a P_E is the interaction object that should execute most actions in a P_E; however, it is also possible to define actions for other entities, E or I, in the P_E. Actions define the specific functionalities of proxemic applications and must be defined by developers.

4. Framework to Develop Proxemic Mobile Apps

We propose a framework, based on our previous work [36], along with a simple systematic development approach, for supporting the construction of mobile proxemic apps for smart proxemic environments (P_E), based on mobile and smart wearable technologies. The systematic development process provides a guidance on how to choose DILMO combinations for developing mobile apps, with particular MobileHCI for specific proxemic environments.

4.1. *Framework Architecture*

The framework architecture is designed to allow the developer to focus on how to obtain the sensing data from the smart environment (i.e., smartphone, wearable device sensors). The contribution provided by the framework is to take advantage of current mobile technology trends related to the capabilities of gathering and processing different types of data that can be used to create proxemic applications. Our approach helps the developer with the development process and with the management of proxemic information to establish the appropriate combination of DILMO dimensions.

In order to create a mobile proxemic app and define a P_E, we propose a sequential process comprised by three single steps:

1. Design the proxemic environment according to which entities will interact based on the graphical DSL [10].

2. Define the appropriate DILMO combination to define the MobileHCI for the target mobile app to be developed.

3. Implement the mobile app, considering the technology supported by the entities in the P_E.

To support this process, the framework is composed by mainly three components aligned with each step (see Figure 1): (i) A graphical DSL module to design the proxemic environment; (ii) DILMO module to define the combination of DILMO dimensions; and (iii) an API that

supports the instantiation of the two previous mentioned components. In the following, we describe each module in detail.

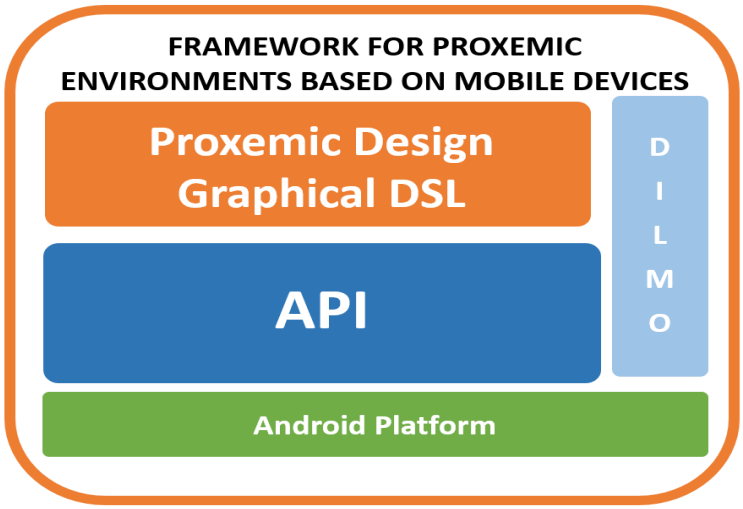

Fig. 1. Framework architecture.

4.2. *Graphical DSL module: Designing the proxemic environment (P_E)*

The graphical DSL allows the definition of the four proxemic zones and the graphical design of entities in the proxemic environment, according to user needs. The DSL is based on formal definitions and graphical notations. The DSL allows the formal and graphical representation of proxemic environments and proxemic behaviours from specific initial conditions of physical objects and entities based on DILMO dimensions. The graphical notation represents in a visual way, all components in a proxemic environment. Each symbol of graphical description can be described as entity in the P_E, which has several behaviors. The values of distances that delimit the proxemic zones are configured through the API. To define a P_E the following initial conditions should be specified:

1. Distances (radios) that define the four P_Z are provided by designers/developers according to user requirements.
2. In a P_E exists only one target entity; thus, given a $P\_E =< CPS, entities, identities, distances >$, CPS is not null.
3. The CPS represents the target interaction object; with the following properties:
   - It can be E or I;
   - The original location of CPS is denoted as $CPS.L = (0, 0)$, since proxemic zones and DILMO dimensions of all other interaction objects are determined with respect to the CPS;
   - It has a face from which its area of vision is defined according to the following angles: $\measuredangle_{MinAofV}$ and $\measuredangle_{MaxAofV}$. If $\measuredangle_{MinAofV}$=$\measuredangle_{MaxAofV}$=NULL, measn that the CPS has not face. These values are parameters provided by users or developers.

4. For all entities (E) in the system, DILMO dimensions and proxemic zones can be assigned as their initial behaviours. Formally:

   $\forall E_i \in P\_E.entities \wedge \forall I_j \in P\_E.identities,$

   $E_i.B_0 =< D_0, L_0, M_0, O_0 > \wedge$

   $I_j.B_0 =< D_0, I_j, L_0, M_0, O_0 >;$

   and L_0, M_0, O_0 can be *null*.

   It means that distance (D) is the only mandatory dimension for entities. Orientation is a dimension that only matters if the CPS has a view angle that shows the space where other entities can be detected (examples of such CPS are a screen, a smartphone using its camera).

From these initial conditions, designers can specify **conditions in a** P_E (behaviours – B) to which entities should react by executing *Actions*. Hence, from the initial conditions, proxemic *Actions* can be defined with respect to the P_Z and DILMO dimensions of entities in the P_E.

4.3. *DILMO module: Defining DILMO combination*

The framework provides a guide that allows developers to know which methods must be implemented on the API or which objects must be created from the API, according to DILMO dimensions for processing proxemic information and according to behaviours and actions identified in the previous step. This module allows developers to relate objects and entities that can interact in the proxemic environment. Hence, according to the combination of DILMO dimensions different proxemic environments can be defined, as denoted in Figure. 2. For example, a DIL proxemic environment (row 30 in Figure 2) means that Distance (D) and Location (L) are considered for Identities (I). Combination of proxemic dimensions are also valid, although the entities have not unique identification; e.g., *a person*, *a device beacon*, instead of *the smartphone's owner*, *my device beacon*.

4.4. *The API: Implementing the mobile proxemic application*

The API facilitates developers processing proxemic information and values. The API provides seven classes, called `ProxZone, DILMO, Distance, Entity, Location, Movement`, and `Orientation`, to define the P_Z, as well as to manage the different combinations of DILMO dimensions, that are required for implementing a P_E. For example, for a DIL proxemic environment, methods to identify entities (I) and to process D and L are available in the DILMO class. Thus, the API behaves as a bridge between the graphical DSL and DILMO modules.

In the current version of our framework, the API considers the extraction of DILMO values from smartphones or mobile devices based on the Android native libraries (APKs). The API provides methods that developers can implement for processing proxemic information using motion sensors and mobile computer vision cameras. The majority of current smartphones have a wide range of sensors in their hardware configuration [37], which allow the application to run proxemic apps. For example, through the BLE beacons mechanisms [38], it is possible to know the distance (D) between two mobile devices. The distance is estimated using the

#	D	I	L	M	0	MIX	Proxemic Environment
1	■					D	Physical length (D) between entities.
2			■			L	Position (L) of an interaction object (entity).
3				■		M	Motion (M) capture of an interaction object (entity).
4					■	O	Face orientation (O) between entities.
5	■	■				DI	Interaction based on proximity (D) between Identities.
6		■	■			IL	Interaction based on the physical location (L) of Identities.
7		■		■		IM	Interaction based on movement tracking (M) of Identities.
8		■			■	IO	Interaction based on face to face orientation (O) of Identities.
9	■	■	■			DIL	Interaction based on proximity (D) and positions (L) of Identities..
10	■	■		■		DIM	Interaction based on proximity (D) according to movement tracking (M) of Identities.
11	■	■			■	DIO	Interaction based on proximity (D) and face to face orientation (O) of Identities.
12		■	■	■		ILM	Interaction based on physical location (L) and movement tracking (M) of Identities.
13		■	■		■	ILO	Interaction based on face to face orientation (O) and position (L) of Identities.
14		■		■	■	IMO	Interaction based on movement tracking (M) and face to face orientation (O) of Identities.
15	■	■	■	■		DILM	Interaction based on proximity (D) and physical location (L) with movement tracking (M) of Identities.
16	■	■	■		■	DILO	Interaction based on proximity (D), location (L) and face to face orientation (O) of Identities.
17	■	■		■	■	DIMO	Interaction based on proximity (D) and face to face orientation (O) according to movement (M) of Identities.
18		■	■	■	■	ILMO	Interaction based on location (L) and face to face orientation (O) according to movement (M) of Identities.
19	■		■			DL	Physical length (D) between entities.
20	■			■		DM	Interaction based on proximity (D) according to movement tracking (M) of entities.
21	■				■	DO	Interaction based on proximity (D) and face to face orientation (O) of entities
22			■	■		LM	Interaction based on physical location (L) and movement tracking (M) of entities.
23			■		■	LO	Interaction based on face to face orientation (O) and position (L) of entities.
24				■	■	MO	Interaction based on movement tracking (M) and face to face orientation (O) of entities.
25	■		■	■		DLM	Physical length (D) between entities.
26	■		■		■	DLO	Interaction based on proximity (D), location (L) and face to face orientation (O) of entities.
27	■			■	■	DMO	Interaction based on proximity (D) and face to face orientation (O) according to movement (M) of entities.
28			■	■	■	LMO	Interaction based on location (L) and face to face orientation (O) according to movement (M) of entitites.
29	■		■	■	■	DLMO	Interaction based on proximity (D) and location (L) and face to face orientation (O) according to movement (M) of entities.
30	■	■	■	■	■	DILMO	Interaction based on all proxemic interaction dimensions.

Fig. 2. DILMO proxemic dimensions and nomenclature to describe each combination that is available through the API methods for processing proxemic information.

signal strength of the mobile device's Bluetooth signals. Another way to estimate the distance between two entities is to use computer vision (face detection). Android provide methods for detecting faces as image frame. This image frame size is used to estimate the distance between the user and the mobile phone camera through the API methods.

The API[a] is available for free downloading. It was developed in Java, hence the `jar` files are provided to be added to the Android Studio platform. Figure 3 describes the structure of the API, represented by a UML Class diagram. The main classes are described as follows:

1. The `ProxZone class` allows to define proxemic zones (P_Z) according to user requirements (i.e., user/developer decides the measures that delimit each P_Z), when this class is instantiated (i.e., by its constructor method). The constructor method of this class receives as parameters the respective maximum measures of distance D, which define each P_Z. A P_Z can be associated to one or more entities.

   Figure 4 shows an example of the `ProxZone` constructor method, in which the maximum distance, in meters, for each P_Z are specified (see Def.): $P\_Z_{intimate}$ is delimited from 0 to 0.25 meter, $P\_Z_{personal}$ is defined from 0.26 meter to 0.45 meter, P_Z_{social} is from 0.46 meter to 1 meter, and P_Z_{public} is depicted from 1.1 meters to 2 meters. Inappropriate arguments are validated in order to have valid measurements (e.g., not overlapped zones, right order of measures).

2. The `DILMO class` is useful for developing P_E. It offers the possibility of identifying the P_Z of all entities E (or identities I) which will interact in the P_E. This class allows to define relations among proxemic dimensions, according to our proposed combinations of DILMO (see Figure 2). Its main methods are:

   (a) The `setProxemicDI(String` I`, double D)` method allows assigning a P_Z to an identity (I), based on the distance (D).

   (b) The `getProxemicDI(String` I`)` method allows obtaining the P_Z of an identity (I).

   (c) The `setProxemicDIL(String` I`,double` D`, float` L`)` allows assigning a P_Z to an identity (I) based on the distance (D) and processing the location (L).

   (d) The `getProxemicDIL(String` I`)` method allows obtaining the P_Z and relative location of the identity (I).

   (e) The `setProxemicDistance(double D)` method allows assigning the P_Z to an entity, according to the distance (D).

   (f) The `getProxemicZoneByDistance()` method returns the P_Z of an entity, based on the distance.

3. The `Distance class` allows the developer to estimate the distance among identities (I). Distance (D) can be calculated by using any available method. In the current version of our API, we have integrated some methods to calculate D, based on the Android platform, such as:

[a]The API is available in https://github.com/paulocpd76/ProxemicEnt15-01-20.git/

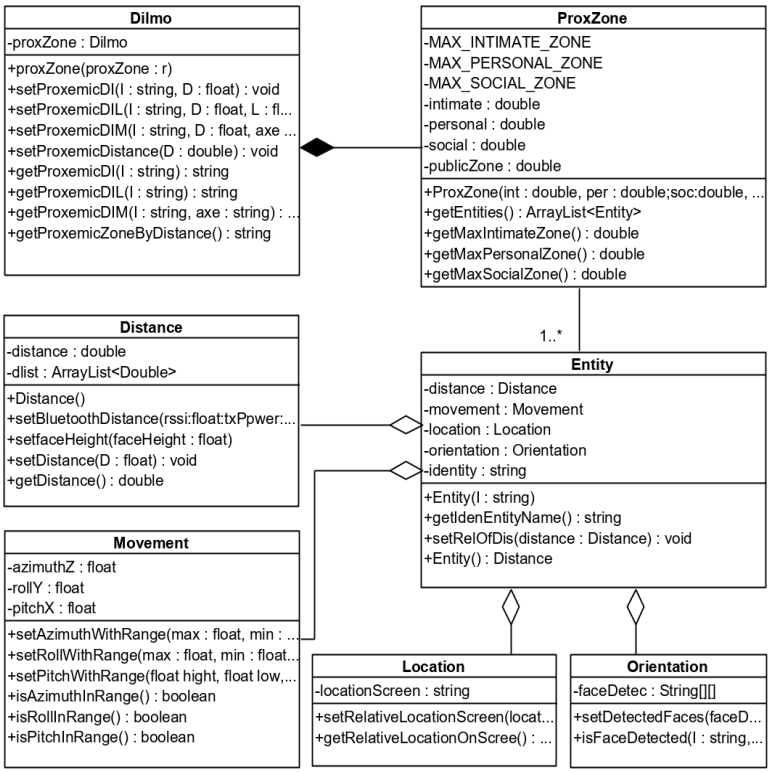

Fig. 3 UML Class Diagram of the API.

```
//Définition des zones proxémiques utilisée
proxzone = new ProxZone(0.25D, 0.45D, 1.0D, 2.0D)
```

Fig. 4. Example of ProxZone Class Constructor invocation.

(a) The `setBluetoothDistance(double rssi, double txtPower)` that allows estimating distance based on BLE.

(b) The `setFaceHeight(float faceHeight)`, which allows estimating the distance from camera using visual computing; distance is proportional to the height of the detected face.

(c) The `getDistance()` method, that allows obtaining D in meters.

4. The `Entity class` represents the interaction objects in a P_E, whose behavior is determined or will be determined according to their DILMO proxemic dimensions. This class allows the discovering of the entities on a P_E.

5. The `Location class` provides methods to manage location (L) of interaction objects (E and I). As in the `Distance class`, location (L) of entities can be calculated by any

available method. Currently, we have integrated in our API some methods to calculate L, such as:

(a) The `setRelativeLocationScreen(float L)` method, which sets the relative position on the screen of an entity E, based on the coordinates of E on the display.

(b) The `getRelativeLocationOnScreen()` method, which returns the relative position of an entity E.

6. The `Movement class` has methods that allow motion processing from the coordinate system of smart-mobile sensors (e.g., Azimuth, Pitch, Roll), such as:

   (a) The `setAzimuthWithRange(float MAX, float MIN, float value)` method which allows processing the azimuth angle of an entity E, that has been established by the developer.

   (b) The `isAzimuthInRange()` method returns true if the azimuth angle of an interaction object (E) is within a range of reference.

7. The `Orientation class` provides methods to validate the face orientation (O) of interaction objects on a P_E. Some of them are:

   (a) The `setDetectedFaces(String[][] detectedFa ces,ProxZone p)` method, that receives a collection of faces to be defined as interaction objects (E or I) in the P_E.

   (b) The `isFaceDetected(String I,String P_Z)` method, which returns true if a specific face (I) is detected in the P_Z.

5. Proof-Of-concept of our Framework

Our goal is to develop proxemic environments, from the design phase, based on mobile devices and demonstrate that our framework allows developers to build proxemic mobile applications effectively. For this purpose, we show the implementation of two mobile applications, called IntelliPlayer and Tonic, based on proxemic interactions. These apps were implemented using Android Studio platform version 3.3; however a higher Android studio version can be used. The apps[b]are available for free downloading.

Both apps have been developed by undergraduate students, as part of their final project in computer science. The developing team was integrated by four students whose average age was 21 years-old, who have developer experience using Java object-oriented programming. This project was the first challenge for them implementing Android applications and MobileHCI based on proxemic interactions.

Two training sessions of two hours each were organised for the student. The training process allows students to understand the systematic process for building proxemic mobile applications with our framework. They learned: (i) how to define each P_Z, using the graphical DSL; (ii) how to select each combination of proxemic dimension for recreating a P_E; and (iii) how to use methods and classes in the API. The development time of both applications was 64 hours by two developers over a period of 4 weeks. IntelliPlayer took 44 hours of work, while Tonic was finished in 20 hours, in the same four weeks.

[b]The APPS are available in https://https://github.com/llagar910e/

5.1. *Tonic app*

Tonic is a mobile app for playing musical notes, developed for illustrative purposes. A user's smartphone, with the Tonic app (*Tonic device*), plays and modifies different musical notes according to its movements and the distance of another smartphone (*visitor device*).

In the first step of the approach, the four proxemic zones (P_Z_{intime}, $P\_Z_{personal}$, P_Z_{social}, and P_Z_{public}) and the functionality of the application are designed. The visitor smartphone is identified (i.e., it is an I) by the *Tonic device*, which plays the sound (i.e., it is the CPS). The initial conditions that defines this P_E are:

- Distances that define the four P_Z: P_Z_{intime} from 0 mts to 0.5mts; $P\_Z_{personal}$ between 0.51 mts and 1 mts; P_Z_{social} from 1.1 mts to 2 mts; and P_Z_{public}, with 2.1 mts to 4 mts.
- The target interaction object is the *Tonic device*. Thus, the four tuple is

  $P\_E = < CPS = Tonic\ device, identity = \{I = visitor\ device\}, distances = \{0.5, 1, 2, 4\} >$;
- Initial location of the CPS: $CPS.L = (0, 0)$; it has not face;
- Initial behavior of CPS: $CPS.B_0 =< D_0 = 0, L_0 = (0, 0), M_0 = quiet >$.
- Initial behavior of I: $I.B_0 =< D_0 = 1 >$.

The volume of the sound is adjusted according to the P_Z in which I is, with respect to the CPS; and the note changes according to the movements of the CPS. Thus, from the initial conditions, the CPS can react (i.e., shows different behaviours), by executing different *Actions*, such as:

- $CPS.B_1$: $if\ \exists\ I$ in $P\_Z_{intimate}$, then $CPS.Action_1$; where $Action_1$= "increase to 25% of volume";
- $CPS.B_2$: $if\ \exists\ I$ in $P\_Z_{personal}$, then $CPS.Action_2$; ; where $Action_2$= "increase to 50% of volume";
- $CPS.B_3$: $if\ \exists\ I$ in P_Z_{social}, then $CPS.Action_3$; where $Action_3$= "increase to 75% of volume";
- $CPS.B_4$: $if\ \exists\ I$ in P_Z_{public}, then $CPS.Action_4$; where $Action_4$= "increase to 100% of volume";
- $CPS.B_5$: $if\ CPS.M = get\ up \vee CPS.M = lay\ down$, then $I.Action_5$; where $Action_5$= "increase a semitone";
- $CPS.B_6$: $if\ CPS.M = gyre$, then $CPS.Action_6$; where $Action_6$= "increase a tone for each grade";

Figure 5(a) shows the graphical representation of P_E based on the DSL.

Thus, the CPS plays and modifies a sound, by using proxemic interactions based on the P_Z, and on D, I, and M dimensions – i.e., a DIM proxemic environment according to DILMO

combination (step two of the approach). In Tonic, the distance (D) between the two devices is obtained by using BLE technology. The entity I broadcasts its identifier to nearby portable electronic devices, thus it is caught by the CPS. The volume of the sound is adjusted according to the P_Z in which I is, with respect to the CPS. To manage P_Z and D, the methods used from the API, in the third step of the approach, were those described in items 2.(b), 2.(c), and 3.(a) in Section : `ProxemicDI(String I)`, `setProxemicDIL(String I, double D, float L)`, and `setBluetoothDistance(double rssi, double txtPower)`. The musical notes are changed according to the smart phone movements. Movements are calculated based on the capabilities of the smartphone, such as accelerometer, gyroscope, compass, and magnetometer. These sensors provide information that is mainly used in the API using methods `setAzimuthWithRange(float MAX, float MIN, float value)` and `isAzimuthInRange()` described in items 6.(a) and 6.(b) in Section .

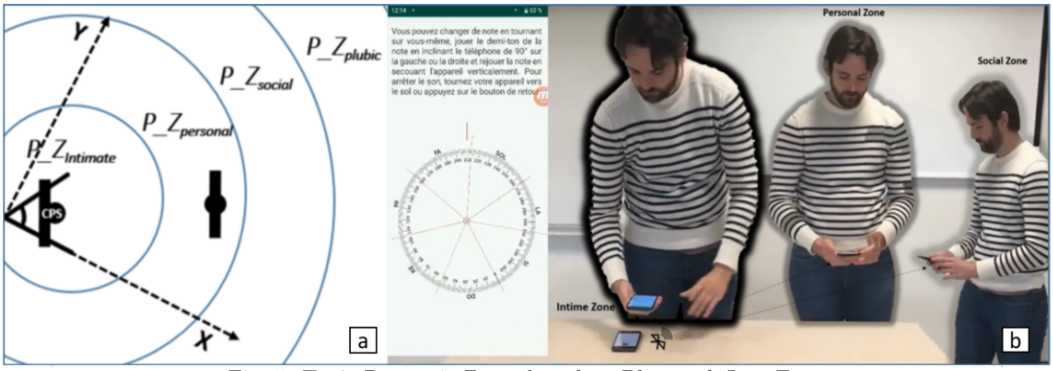

Fig. 5. Tonic Proxemic Zones based on Bluetooth Low Energy.

5.2. *IntelliPlayer app*

IntelliPlayer is a mobile application that plays a video in a smartphone (*IntelliPlayer device*) and reacts according to four proxemic zones and DLO proxemic dimensions of another entity (E_1). In the first step of the approach, the four P_Z and the initial condition of the CPS are designed:

- Distances that define the four P_Z: P_Z_{intime} between 0 mts and 0.25mts; $P\_Z_{personal}$ from 0.26 mts to 0.45 mts; P_Z_{social} from 0.46mts to 1 mts; and P_Z_{public} between 1.1 mts and 2 mts.
- The target interaction object is the *IntelliPlayer device*. Thus, the four tuple is:
  $P\_E =< CPS = IntelliPlayer\ device, entities = \{E_1\}, distances = \{0.25, 0.45, 1, 2\} >$;
- $CPS.L = (0, 0)$; it has face;
- Initial behavior of CPS: $CPS.B_0 =< D_0 = 0, L_0 = (0, 0), O_0 = 0 >$, $\measuredangle_{MinAofV} = 0°$ and $\measuredangle_{MaxAofV} = 75°$;
- Initial behaviour of E_1; Distance of E_1: $E_1.D = 2$; Orientation of E (the face of E_1 is in the area of vision of the CPS); thus, $E_1.B_0 =< D_0 = 2, O_0 = 35° >$, $E_2.B_0 =< D_0 = 2, O_0 = 45° >$.

With the distance (D) between the user (E_1) and the smartphone (CPS), IntelliPlayer determines the proxemic zone (P_Z) of E_1 (a user), with respect to the smartphone (CPS). To do so, it invokes the method `getProxemicZoneByDistance()` described in item 2.(f) in Section . IntelliPlayer automatically adjusts the volume of the video according to the P_Z in which E_1 (the user) is with respect to the smartphone (CPS): when E_1 is in P_Z_{intime}, it decreases to 25% volume of speaker; for $P\_Z_{personal}$, it increases to 50% volume; for P_Z_{social}, it increases to 75% volume; and for P_Z_{public}, it increases to 100% volume) (see Figure 6). These behaviours are designed as follows:

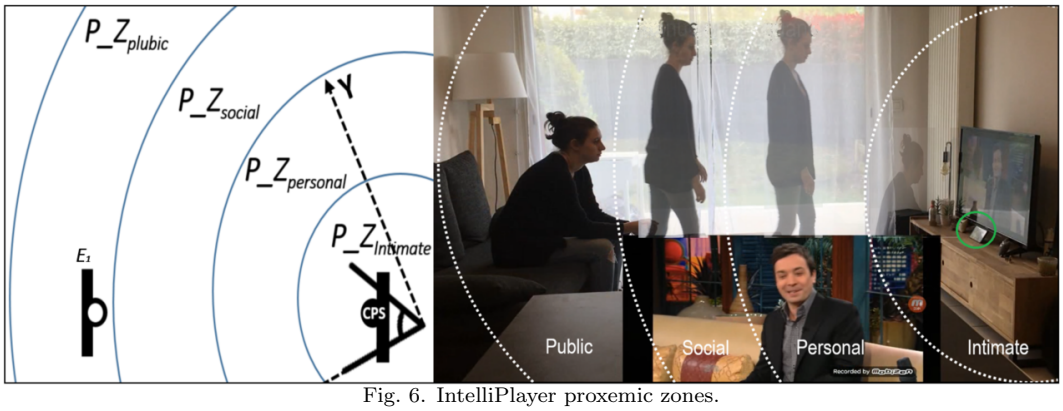

Fig. 6. IntelliPlayer proxemic zones.

- $CPS.B_1$: $if\ \exists\ E_1$ in $P\_Z_{intimate} \wedge \measuredangle_{MinAofV} \leq E_1.O \leq \measuredangle_{MaxAofV}$, then $CPS.Action_1$; where $Action_1$ = "increase to 25% of volume";
- $CPS.B_2$: $if\ \exists\ E_1$ in $P\_Z_{personal:} \wedge \measuredangle_{MinAofV} \leq E_1.O \leq \measuredangle_{MaxAofV}$, then $CPS.Action_2$; where $Action_2$ = "increase to 50% of volume";
- $CPS.B_3$: $if\ \exists\ E_1$ in $P\_Z_{social:} \wedge \measuredangle_{MinAofV} \leq E_1.O \leq \measuredangle_{MaxAofV}$, then $CPS.Action_3$; where $Action_3$= "increase to 75% of volume";
- $CPS.B_4$: $if\ \exists\ E_1$ in $P\_Z_{public:} \wedge \measuredangle_{MinAofV} \leq E_1.O \leq \measuredangle_{MaxAofV}$, then $CPS.Action_4$; where $Action_1$ ="increase to 100% of volume";

Then, in the second step, the MobileHCI is designed according to D, L, and O; thus a DLO P_E is defined. Figure 6 shows the proxemic zones that have been defined by the developer through the API and DSL. With these behaviors, we illustrate a proxemic environment using a mobile player app that reacts to the distance (D) and location (L) of a person (E_1) and his face orientation (O), with respect to the smartphone (CPS) displaying a video. The computer vision technique has been used for this purpose, based on the properties of an Android camera and through the API methods `setFaceHeight(float faceHeight)` and `getDistance()` described respectively in items 3.(b) and 3.(c) in Section . Figure 7 shows a block code of this case.

More behaviours can be designed. When a second person (E_2) is in front of the smartphone camera (I_1), the application verifies if both users are looking at the screen at the same time, as shown in Figure 8(a). The DSL describes the P_E:

```
Distance d= new Distance();
d.setfaceHeight(faces.valueAt(i).getHeight());
String id =String.valueOf(faces.valueAt(i).getId());
dilmo.setProxemicDI(id, d.getDistance());
dilmo.getProxemicDI(id);
changerVolume(dilmo.getProxemicDI(id));
//Log.i("FaceId", "Zone : "+ id + "/"+ dilmo.getProxemicDI(id));
```

Fig. 7. Block code of IntelliPlayer.

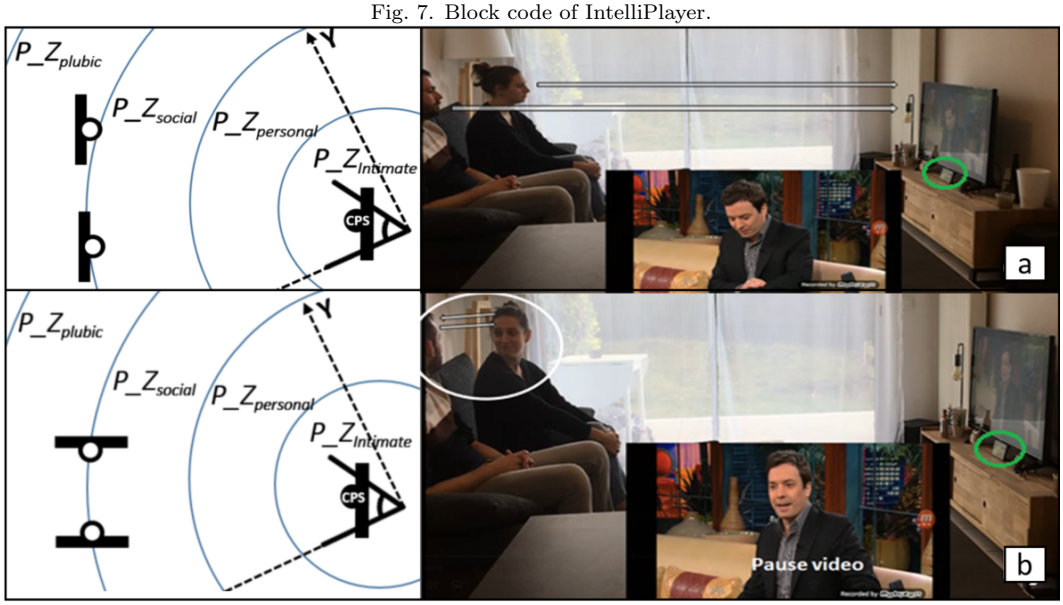

Fig. 8 Play pause video using users faces orientation.

- The four tuple of P_E:
  $P\_E =< CPS, entities = \{E_1, E_2\}, distances = \{0.25, 0.45, 1, 2\} >$;
- $CPS.L = (0, 0)$; $\measuredangle_{MinAofV} = 0°$ and $\measuredangle_{MaxAofV} = 75°$;
- Distance of E_1: $E_1.D = 2$, $E_2.D = 2$;
- Orientation of E (the face of E_1 or E_2 is in the area of vision of the CPS) thus, $E_1.B_0 =< D_0 = 2, O_0 = 45° >$, $E_2.B_0 =< D_0 = 2, O_0 = 35° >$.

Behaviours according to this scenario are presented in Figure 8(a) using he graphical DSL, and the actions are:

- $CPS.B_5$: $if\ \exists\ E_1 \wedge E_2$ in $P\_Z_{public} \wedge \measuredangle_{MinAofV} \leq E_2.O \leq \measuredangle_{MaxAofV} \wedge \measuredangle_{MinAofV} \leq E_1.O \leq \measuredangle_{MaxAofV}$, then $CPS.Action_5$ = video starts (Figure 8 (a)).
- $CPS.B_6$: $if\ E_2.O > \measuredangle_{MaxAofV} \wedge E_1.O > \measuredangle_{MaxAofV}$, then $CPS.Action_6$ = video is paused automatically (Figure 8 (b)).

The method `setDetectedFaces (String[][]detectedFaces, ProxZone p)`, item 7.(a) in Section) is used to process detected faces. In the case one user (E_1 or E_2) turns his face, the video will be paused automatically by the mobile application (see Figure 8(b)).

Another useful function of IntelliPlayer is to provide a video description that users can read on the screen according to user location (L). Figure 9 shows the description of this scenario using the DSL.

Conditions and behaviours of this P_E, are described with the DSL, as follows:

- The four tuple of P_E:

  $P\_E =< CPS, entities = \{E_1, E_2\}, distances = \{0.25, 0.45, 1, 2\} >$;

- $CPS.L = (0,0)$ $\measuredangle_{MinAofV} = 0°$ and $\measuredangle_{MaxAofV} = 75°$;

- Distance of E_1: $E_1.D = 2$, $E_2.D = 0.45$;

- Location of $E_1.L = (x_1, y_1)$; $E_2.L = (x_2, y_2)$, *face position relate to smartphone camera in pixels.*

- Orientation of E (the face of E_1 or E_2 is in the area of vision of the CPS), thus $E_1.B_0 =< D_0 = 2, O_0 = 45° >$, $E_2.B_0 =< D_0 = 2, O_0 = 35° >$.

The behaviour according to the scenario presented in Figure 9 is:

- $CPS.B_7$: if $\exists$ E_1 in P_Z_{public} $\wedge$ E_2 in $P\_Z_{personal}$ $\wedge$ $\measuredangle_{MinAofV} \leq E_2.O \leq \measuredangle_{MaxAofV}$ $\wedge$ $\measuredangle_{MinAofV} \leq E_1.O \leq \measuredangle_{MaxAofV}$, $\wedge$ $E_2.L = (x_2, y_2)$,then $CPS.Action_6$ = screen show information about the video (Figure 9).

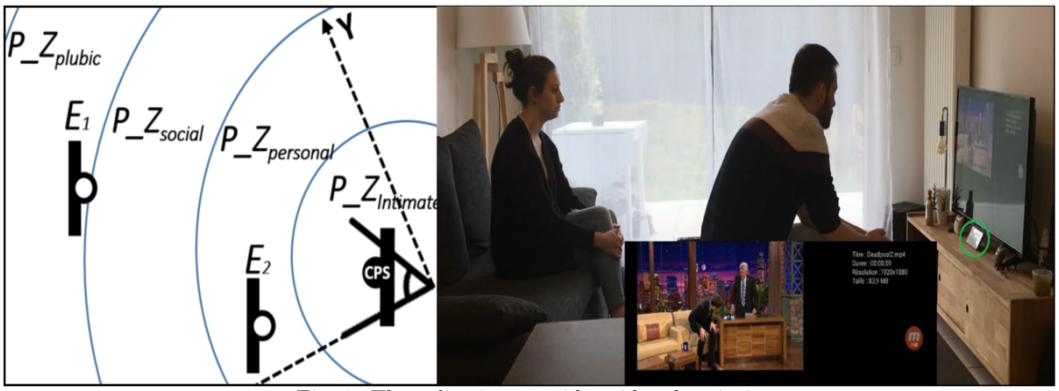

Fig. 9. The split view provides video description.

When a user (E_1 or E_2) is in the $P\_Z_{personal}$ and his orientation (O) is in front of the screen, the application can obtain the face location (L) (see `setRelativeLocationScreen(float L)` and `getRelativeLocationOnScreen()` in item 5.(a) and 5.(b) in Section) to split the screen, with the video (running) and information about the video on the right or on the left, according to the detected L. The correct use and instance of methods and classes of the API conforms the third step of the proposed approach.

5.3. *Usability evaluation*

In order to evaluate the usability of the API, we applied a survey composed of nine questions to the group of students who developed the applications described. Results of the survey

are shown in Figure 10. These results indicate that 100% of surveyed students have strongly agreed with **Q1**: *"It was easy to implement the API with Android Studio"* and **Q9**: *"The API is useful for the development of proxemic applications with Android?"*. While 95% of students endorsed **Q5**: *"The API allowed you to process information of a combination of DILMO dimensions"*, **Q7**: *"The API's method for estimating distance based on face detection was accurate"*, and **Q8**: *"The API allows you to obtain the proxemic zone when it is using Bluetooth proximity sensing"*. For **Q4**: *"The API allows you to improve your productivity for developing proxemic applications by hiding complexity"* and **Q6** *"The API allows you to create the proxemic zones quickly"*, 78% of students expressed acceptation. Finally, **Q2**: *"The documentation provides enough information to interact with the API"* and **Q3**: *"The API provides enough examples for creating proxemic applications"*, were accepted only by 53% and 61% of students, respectively.

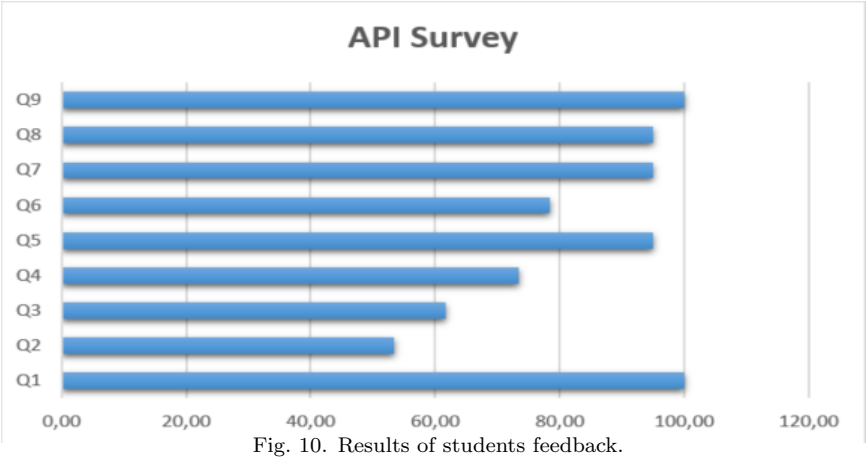

Fig. 10. Results of students feedback.

The the proof-of-concept and survey allow corroborating that the API supports development of proxemic applications and creation of proxemic environments, based on the exclusive use of mobile devices, while reducing complexity of the development process. These apps offer portable implementations that facilitate using the proxemic interaction in comparison to previous works that have used fixed platforms for similar purposes. Moreover, the survey results allow knowing aspects to develop, such as the quality of documentation and the basic examples provided.

6. Conclusion

Mobile technologies are frequently used in daily life for different activities, and their use keeps increasing. Through this work introduced a framework that includes an API for developing proxemic applications for smart environments comprised of entities whose interactions are supported by proxemic dimensions DILMO. We demonstrated the framework's effectiveness through the proof-of-concept, which details the implementation of two proxemic mobile applications developed by undergraduate students in computer science. With this framework, we provide a tool that can help developers to build novelty proxemic mobile applications using social distancing. We used a domain-specific language (DSL) to design proxemic environments. With our proposal, we hope to inspire other researchers to build more mobile

applications based on proxemic interactions in accordance with real-life needs. We aim to improve the current API for building mobile applications based on gesture interaction and proxemic interactions in a future work.

1. Frederik Brudy, Christian Holz, Roman Rädle, Chi-Jui Wu, Steven Houben, Clemens Nylandsted Klokmose, and Nicolai Marquardt. Cross-device taxonomy: Survey, opportunities and challenges of interactions spanning across multiple devices. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '19, page 562. ACM, 2019.
2. Jens Emil Grønbæk, Christine Linding, Anders Kromann, Thomas Fly Hylddal Jensen, and Marianne Graves Petersen. Proxemics play: Exploring the interplay between mobile devices and interiors. In *Proceedings of Companion Publication of the Conference on Designing Interactive Systems*, DIS '19, pages 177–181. ACM, 2019.
3. Till Ballendat, Nicolai Marquardt, and Greenberg Saul. Proxemic interaction: designing for a proximity and orientation-aware environment. In *Proceedings of International Conference on Interactive Tabletops and Surfaces*, ITS' 10, pages 121–130. ACM, 2010.
4. Saul Greenberg, Nicolai Marquardt, Till Ballendat, Rob Diaz-Marino, and Miaosen Wang. Proxemic interactions: the new ubicomp? *Interactions*, 18(1):42–50, 2011.
5. Jens Emil Grønbæk, Mille Skovhus Knudsen, Kenton O'Hara, Peter Gall Krogh, Jo Vermeulen, and Marianne Graves Petersen. Proxemics beyond proximity: Designing for flexible social interaction through cross-device interaction. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2020.
6. Edward T Hall. *The Hidden Dimension: An anthropologist examines man's use of space in private and public.* New York: Anchor Books; Doubleday & Company, Inc, 1966.
7. Nicolai Marquardt, Robert Diaz-Marino, Sebastian Boring, and Saul Greenberg. The proximity toolkit: prototyping proxemic interactions in ubiquitous computing ecologies. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 315–326. ACM, 2011.
8. Mihai Bâce, Sander Staal, Gábor Sörös, and Giorgio Corbellini. Collocated multi-user gestural interactions with unmodified wearable devices. *Augmented Human Research*, 2(1):6, 2017.
9. Paulo Pérez, Philippe Roose, Yudith Cardinale, Mark Dalmau, Dominique Masson, and Nadine Couture. Mobile proxemic application development for smart environments. In *Proceedings of the 18th International Conference on Advances in Mobile Computing & Multimedia*, pages 94–103, 2020.
10. Paulo Pérez, Philippe Roose, Yudith Cardinale, Marc Dalmau, Dominique Masson, and Nadine Couture. Proxemic environments modelling based on a graphical domain-specific language. In *2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–8. IEEE, 2020.
11. Nicolai Marquardt, Ken Hinckley, and Saul Greenberg. Cross-device interaction via micromobility and f-formations. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, UIST '12, pages 13–22. ACM, 2012.
12. Tilman Dingler, Markus Funk, and Florian Alt. Interaction proxemics: Combining physical spaces for seamless gesture interaction. In *Proceedings of the 4th International Symposium on Pervasive Displays*, PerDis '15, pages 107–114. ACM, 2015.
13. Jakub Dostal, Uta Hinrichs, Per Ola Kristensson, and Aaron Quigley. Spidereyes: designing attention-and proximity-aware collaborative interfaces for wall-sized displays. In *Proceedings of the 19th international conference on Intelligent User Interfaces*, IUI '14, pages 143–152. ACM, 2014.
14. J Antonio Garcia-Macias, Alberto G Ramos, Rogelio Hasimoto-Beltran, and Saul E Pomares Hernandez. Uasisi: a modular and adaptable wearable system to assist the visually impaired. *Procedia Computer Science*, 151:425–430, 2019.
15. Kyle Kyle, Keith Salmon, Dan Thornton, Neel Joshi, and Meredith Ringel Morris. Eyes-free art: Exploring proxemic audio interfaces for blind and low vision art engagement. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(3):93, 2017.

16. Ghare Mojgan, Pafla Marvin, Caroline Wong, James R Wallace, and Stacey D Scott. Increasing passersby engagement with public large interactive displays: A study of proxemics and conation. In *Proceedings of the International Conference on Interactive Surfaces and Spaces*, ISS'18, pages 19–32. ACM, 2018.
17. Michael Brock, Aaron Quigley, and Per Ola Kristensson. Change blindness in proximity-aware mobile interfaces. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 43. ACM, 2018.
18. Mikkel R Jakobsen, Yonas Sahlemariam Haile, Søren Knudsen, and Kasper Hornbæk. Information visualization and proxemics: design opportunities and empirical findings. *IEEE transactions on visualization and computer graphics*, 19(12):2386–2395, 2013.
19. Jo Vermeulen, Kris Luyten, Karin Coninx, Nicolai Marquardt, and Jon Bird. Proxemic flow: Dynamic peripheral floor visualizations for revealing and mediating large surface interactions. In *Proceedings of IFIP Conference on Human-Computer Interaction*, INTERACT'15, pages 264–281. Springer, 2015.
20. Augustin Zidek, Shyam Tailor, and Robert Harle. Bellrock: Anonymous proximity beacons from personal devices. In *Proceedings of International Conference on Pervasive Computing and Communications*, PerCom'18, pages 1–10. IEEE, 2018.
21. Yapeng Wang, Xu Yang, Yutian Zhao, Yue Liu, and Laurie Cuthbert. Bluetooth positioning using rssi and triangulation methods. In *Proceedings of the 10th Consumer Communications and Networking Conference*, CCNC'13, pages 837–842. IEEE, 2013.
22. Seyed Ali Cheraghi, Vinod Namboodiri, and Laura Walker. Guidebeacon: Beacon-based indoor wayfinding for the blind, visually impaired, and disoriented. In *Proceedings of International Conference on Pervasive Computing and Communications*, PerCom '17, pages 121–130. IEEE, 2017.
23. Helena M Mentis, Kenton O'Hara, Abigail Sellen, and Rikin Trivedi. Interaction proxemics and image use in neurosurgery. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI'2012, pages 927–936. ACM, 2012.
24. Paulo Pérez, Philippe Roose, Dalmau Marc, Nadine Couture, Yudith Cardinale, and Dominique Masson. Proxemics for first aid to unconscious injured person. In *Proceedings of the 30th Conference on l'Interaction Homme-Machine*, IHM'18, pages 156–162, 2018.
25. David Ledo, Saul Greenberg, Nicolai Marquardt, and Sebastian Boring. Proxemic-aware controls: Designing remote controls for ubiquitous computing ecologies. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI'2015, pages 187–198. ACM, 2015.
26. Henrik Sørensen, Mathies G Kristensen, Jesper Kjeldskov, and Mikael B Skov. Proxemic interaction in a multi-room music system. In *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration*, OzCHI '13, pages 153–162. ACM, 2013.
27. Microsoft. Cameraspacepoint structure, 2109. Microsoft Docs https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn758354(v%3Dieb.10).
28. Vicon. About vicon motion systems, 2108. https://www.vicon.com/vicon/about.
29. Ahmed E Mostafa, Saul Greenberg, Emilio Vital Brazil, Ehud Sharlin, and Mario C Sousa. Interacting with microseismic visualizations. In *Proceedings of CHI Extended Abstracts on Human Factors in Computing Systems*, HRI'13, pages 1749–1754. ACM, 2013.
30. Jens Emil Grønbæk and Kenton O'Hara. Built-in device orientation sensors for ad-hoc pairing and spatial awareness. In *Proceedings of Cross-Surface Workshop*, 2016.
31. Jakub Dostal, Per Ola Kristensson, and Aaron Quigley. Multi-view proxemics: distance and position sensitive interaction. In *Proceedings of the 2nd ACM International Symposium on Pervasive Displays*, PerDis '13, pages 1–6. ACM, 2013.
32. Katrin Wolf, Yomna Abdelrahman, Thomas Kubitza, and Albrecht Schmidt. Proxemic zones of exhibits and their manipulation using floor projection. pages 33–37. ACM.
33. Han-Jong Kim, Ju-Whan Kim, and Tek-Jin Nam. ministudio: Designers' tool for prototyping

ubicomp space with interactive miniature. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 213–224. ACM, 2016.
34. Carlos Cardenas and J Antonio Garcia-Macias. Proximithings: Implementing proxemic interactions in the internet of things. *Procedia Computer Science*, 113:49–56, 2017.
35. Lingling Gao, Kerem Aksel Waechter, and Xuesong Bai. Understanding consumers' continuance intention towards mobile purchase: A theoretical framework and empirical study–a case of china. *Computers in Human Behavior*, 53:249–262, 2015.
36. Paulo Pérez, Philippe Roose, Nadine Couture, Yudith Cardinale, Marc Dalmau, and Dominique Masson. Proxemic environments: A framework for developing mobile applications based on proxemic interactions. In *Proceedings of the 2020 Federated Conference on Computer Science and Information Systems*, pages 653–656, 2020.
37. Google. Sensors, 2109. https://developer.android.com/guide/topics/sensors.
38. AltBeacon. The open and interoperable proximity beacon specification, 2018. https://altbeacon.org/.