Programming natural interfaces through the combination of smart phone sensors

Javier Hernández-Aceituno
Dpto. Ingeniería Informática y de Sistemas
Universidad de La Laguna
San Cristóbal de La Laguna, Spain
jhernaac@ull.edu.es

Isabel Sánchez-Berriel
Dpto. Ingeniería Informática y de Sistemas
Universidad de La Laguna
San Cristóbal de La Laguna, Spain
isanchez@ull.edu.es

***Abstract*—One of the most striking aspects of learning how to program smart phone applications is the possibility of producing interfaces based on motion and basic human physical skills, instead of bing limited to the usage of keyboards and mice. The present work describes a series of practical exercises, designed to provide students with the required skills to access and process the information generated by smart phone sensors, and to combine them to create natural interfaces.**

***Index Terms*—Intelligent interfaces, natural interfaces, smart phones, programming, sensors.**

I. Introduction

One of the main objectives of the design of computer application, especially regarding those destined to operate predominantly under mobile platforms such as tablets or smart phones, is the usage of interfaces that are not limited to conventional user interaction methods, such as keyboards or computer mice.

A *natural interface* is that which allows the user to interact with a system or application through gestures, motions and basic human skills, such as gaze and speech, in a way that feels simple and comfortable o the user, without requiring any training. This kind of interfaces has been broadly studied and applied o several research fields [1]–[4].

Students' interest in developing this kind of interfaces is motivated by the large amount of sensors which are available in mobile devices, such as gyroscopes, accelerometers, global positioning, cameras and microphones, and the simplicity of modern programming languages regarding their integration in any kind of application.

It is therefore imperative to teach university students of Computer Science degrees how to use the development tools which are available to them and which allow them to create computer applications with natural interfaces.

There currently exists a trend that favors he usage of natural interfaces as teaching support, especially in cases where traditional methods are unavailable or ineffective. The Telematics Faculty of Colima University (Mexico) has developed gesture and tactile interfaces for special education and he physical activation of elementary school students [5]. The University of Zaragoza has also developed a visual detection system for early childhood education, based on shape and color recognition trough a camera [6]. The University of La Coruña performed an analysis of possible designs in augmented reality as a learning interface [7].

The present work is based on the quarterly subject "Intelligent Interfaces", belonging to the Computation itinerary of the Computer Science degree of University of La Laguna. This subject teaches the basics and technologies of the analysis and design of interactive experiences, virtual reality, augmented reality and natural interfaces, organized in the following epigraphs:

1) Basics and technology of the analysis and design of interactive experiences
2) Virtual reality and augmented reality
3) Natural interfaces
4) Affective and emotional interaction

The skills with which it provides students are the following:

- General skills

CG4	Ability to define, evaluate and choose hardware and software platforms for the development and execution of systems, services and computer applications.
CG6	Ability to conceive and develop computer systems and centralized or distributed architectures integrating hardware, software and networks.
CG9	Ability to solve problems with initiative, decisiveness, autonomy and creativity. Ability to know how to communicate and transmit knowledge and skills of the Computer Engineering Technician profession.

- Itinerary specific skills

C44	Ability to develop and evaluate interactive systems and complex information presentations, and their application in solving design problems in person–computer interaction.

- Transversal skills

T7	Ability of effective communication (expression and comprehension), both oral and written, with special emphasis in technical documentation drafting.
T10	Ability to quickly integrate and efficiently work in unidisciplinary teams and collaborate in multidisciplinary environments.
T21	Ability of critical, logical and mathematical reasoning.

T23 Abstraction skills: ability to create and use models that represent real situations.

Being eminently practical, the valuation of the "Intelligent Interfaces" subject is based on weekly individual lab projects and the group development of two prototypes, one based on virtual reality and the other in one of he remaining techniques that are explained during the term.

We must point out that the "Intelligent Interfaces" subject does not deal with the physical functioning of mobile device sensors, but only with their usage in programming applications with natural interfaces. However, the University of La Laguna offers other subjects within its Computer Engineering Degree, such as Computational Robotics [8], through which students may complement their learning with detailed hardware-level explanations of the inner functioning of sensory devices.

A. Evaluation

The importance of practical exercises in the subject is reflected on the strategies followed for its evaluation. During each practical lab session involves an exercise that requires the development of a minimally functional application, in which a specific element is required to be used. This way, all exercises can be grouped within one of the following categories:

- Learning 3D graphics programming through Unity
- Development of applications using sensors:
  - Development of VR applications
  - Development of AR applications
  - Development of applications involving sensor-based interaction

Every week, the tasks to be solved by students are graded, making up to 50% of the final grade of the subject. Furthermore, all the learned techniques and technologies must be applied to the development of a VR application prototype. The theme of the prototype is free for students to decide and, although generally students choose to develop a videogame, they have sometimes chosen immersive simulations, such as "space mining simulation using reactive agents" (2017–2018 term). They are also asked for the application to integrate some of the additional sensors, such as the accelerometers or the microphone, as part of the user interaction aspect.

All these works are produced by teams of at most three people, so there is a special emphasis on the usage of collaborative project development tools. This aspect is especially important in the development of skill T10, which is considered basic in the daily work of any Computer Engineer. On the other hand, from the point of view of the teacher, grading becomes easier as it allows them to trace the input of each member of the group.

While there are no restrictions regarding which tool to use, the usage of *Git* and the *Github* platform, along with the collaborative development tool *Collaborate* provided by Unity, is recommended. Regardless of which tool each team choses to use for their collaborative work, all prototypes must be delivered through a link to Github, with the whole project code and a document which includes:

Fig. 1. Google Cardboard frame

- Accomplished milestones
- Problems and difficulties faced during the project, both solved and simplified when unfeasible
- Record of team decisions and project tracking

The grade of the prototype development makes up to 20% the final grade of the subject; the presentation of the project makes up to another 20%.

II. Methods

The practical lessons of this subject are based on the development of Unity3D applications [9], using C# language, generally for their execution on smart phones with an Android operative system. After an introduction to the implementation of Unity3D code, students are weekly introduced to different tools to access and process the information provided by smart phone sensors, and their combination to create natural interfaces.

Generally students are not imposed a strict description of what application they must program very week. Instead, they are given freedom to use the acquired information creatively, producing customized applications which adhere only to a small amount of requisites, related to the usage of the different sensors.

A. Virtual reality

For their first assignment, students become familiarized with the Google Cardboard Virtual Reality package [10]. Using the accelerometers of a smart phone mounted on a cardboard frame, it is possible to create a set of very low-cost virtual reality goggles, which are really easy to program (Fig. 1).

The integration of Google Cardboard functionalities within a Unity application is relatively simple and is explained in detail in the web sites of both projects. Generally, it suffices to include the *GoogleVRForUnity_*.unitypackage* package in the resource list of the application, which produces a stereoscopic view aligned to the relative orientation of the mobile device in which it runs (Fig. 2).

The main difficulty when using this resources derives from the constant changes in Google's virtual reality application

Fig. 2. Stereoscopic view

development framework. Since this technology began being used in practical teaching–learning tasks in the “Intelligent Interfaces” subject, during the 2015–2016 term, there have been variations in the development API. During the 2015–2016 term, the Google Cardboard package was available for Unity; during the 2016–2017 term, Google launched DayDream, their first attempt to produce a mobile device which would improve the VR experience.

During the 2017–2018 term, Google updated their VR framework, unifying both devices, Cardboard y Daydream, within the same Unity package. In both terms, the change happened just one month before the period in which the subject is taught, which made all the teaching material of previous terms obsolete. Furthermore, counting exclusively on Google documentation creates an additional difficulty, since the developer community does not have the time to produce other resources. There is however the advantage of the great productivity the usage of the Unity scene editor allows for, since it gives students the chance to obtain playable, highly immersive, low-cost VR application prototypes.

This kind of applications, which is generally highly appealing to users, entails the risk of lessening the experience of the user due to the discomfort caused by the effect known as *simulator drowsiness*. For this reason, one of the laboratory exercises involves studying and practicing the design recommendations offered by Google Team, regarding VR applications.

On one hand, considerations that must be taken into account in order to avoid drowsiness are provided, all of which are expected to reduce the discrepancy between the real perception of the user and the events happening within the virtual world. On the other hand, there are guidelines to orient the user during this new form of interaction, with which they are not yet familiar, unlike conventional PC or mobile phone applications. Along this line, the laboratory exercise regarding the usage of grids and raycast-based scene object interaction is especially important. Grids are a visual object that lets the user know when their line of sight has focused on (selected) an element of the scene that can be interacted with.

Fig. 3. Screen shot of application “Zombie Timer α”, which integrates the use of the mobile device compass.

B. Position sensors

Afterwards, students learn how to manually access the values returned by the accelerometers, the inner compass and the global positioning system (GPS) of the device. The development code for all these elements is held within the *Input* class of the *UnityEngine* package, which must be imported to the application.

The output data of the inner compass are given by the numeric floating-point (*float*) attribute *Input.compass.trueHeading*, which provides the orientation of the device in respect to the geographic North Pole. Although this value is enough for most functional applications, the *Input* package also provides the orientation in respect to the magnetic North Pole, given by attribute *Input.compass.magneticHeading*, and raw geomagnetic information, measured in microteslas, given by object *Input.compass.rawVector*. In order to access all these values, the application must initialize the localization system of the device, by calling function *Input.location.Start()* (Fig. 3).

The accelerometer information of the mobile device can be easily accessed at any moment, as a vector-type (*Vector3*) object *Input.acceleration*. *Vector3* data contain three *float* attributes, labeled as x, y, z which, for accelerometers, indicate the device acceleration in each Cartesian axis (Figs. 4 and 5).

Finally, the global positioning system of the device provides its coordinates, as *latitude*, *longitude* and *altitude* values, as attributes of object *Input.location.lastData*. These attributes are only updated once function *Start()* of object *Input.location* is called.

Function *Start()* has two optional arguments: the desired precision and the minimum update distance, both measured in meters and with a default value of $10\,m$. The value of attribute *Input.location.status* tells if the service is active (*LocationServiceStatus.Running*), and should be checked before reading *Input.location.lastData*. Once the localization service is done being used, it should be stopped by calling function

Fig. 4. Screen shot of application “VR Combat Arcade”, which uses the mobile device accelerometers as control input

Fig. 5. Screen shot of application “GRESBALL”, which uses the mobile device accelerometers as control input

Input.location.Stop().

C. Audiovisual input sensors

In the next exercise, students are taught how to access the microphone and camera of their mobile devices through code. Their data are processed in following exercises (Fig 6).

Unity can obtain sound from the microphone of the computer or mobile device through function *Start (input, loop, duration, frequency)* of class *Microphone*, which returns an *AudioClip* object and whose parameters are:

- *input*: Name of the input device, given as a character *string*. If empty, the default device will be used.
- *duration*: Maximum duration in seconds of the *AudioClip* object, given as an integer (*int*).
- *loop*: Boolean value (*bool*) which tells if, once the maximum duration is reached, the recording should loop back to the beginning of the *AudioClip* object or just stop.
- *frequency*: Sampling frequency, given as a *float* value, which defines the quality of the produced audio.

To stop the recording, function *Microphone.End (input)* must be called.

The simplest way to obtain images from the mobile phone camera or the computer webcam is through Unity’s *WebCamTexture* class, which converts the data provided by those devices into 2D textures. To see the recorded images, the

Fig. 6. Screen shot of application “Program for the Investigation of Pursuing Objectives”, which integrates the usage of the microphone to interact with characters.

WebCamTexture object must be associated to the visual texture attribute *mainTexture* of a material, followed by calling the *Play()* function of the object. Calling function *Pause()* will pause the reproduction and function *Stop()* will stop it. In the practical exercise, students were tasked with using all of these functions, in such a way that isolated frames could be captured and stored as still images.

D. Speech recognition

Unity offers some speech recognition tools, although for now their usage is limited to devices equipped with the Windows 10 operative system [11]. To use these tools, package *UnityEngine.Windows.Speech* must be imported. Although this package offers several methods to recognize words and sentences through the microphone of the device, only classes *KeywordRecognizer* y *DictationRecognizer* were proposed in the “Intelligent Interfaces” subject, due to their lower complexity.

Class *KeywordRecognizer* attempts to match audio, captured from the microphone of the computer or mobile phone, to a given list of keywords. It is possible to keep several of these objects active at the same time, as long as their sets of keywords contain no common elements.

To initialize an object of this class, constructor *KeywordRecognizer (List)* must be called, where the *string* array *List* contains all the keywords to be matched. An object of this class can be enabled and disabled through its functions *Start()* and *Stop()* respectively. Students were told about the importance of calling the *Dispose()* function when done using this type of objects, in order to free their resources.

The *KeywordRecognizer* object detects speech once associated to a listener *void function (data)*, where the *PhraseRecognizedEventArgs data* parameter contains:

- the obtained phrase *data.text*, as a *string*;
- the confidence level *data.confidence*, as a *Windows.Speech.ConfidenceLevel* enumerate, whose values can be *High*, *Medium*, *Low* and *Rejected*;
- the duration of the phrase *data.phraseDuration*, as a *TimeSpan*;
- the start time *data.phraseStartTime*, as a *DateTime*;

- the semantic meaning *data.semanticMeanings*, as a *SemanticMeaning[]*. This attribute holds no use for *KeywordRecognizer*, but it does for other more complex speech recognition classes, such as *GrammarRecognizer*

. Once the listener function is built, it is associated to the *KeywordRecognizer* object using the sentence *object.OnPhraseRecognized += function*.

Additionally, class *DictationRecognizer* tries to transcript the audio captured from the microphone of the computer or the mobile phone, not requiring a list of keywords. An object of this class can also be enabled or disabled through its *Start()* and *Stop()* functions respectively. Again, it's important to call function *Dispose()* when done using the object to free its resources. A *DictationRecognizer* object expects to be associated to four listener functions:

- *void object.DictationResult (string text, Windows.Speech.ConfidenceLevel confidence)*, called whenever a phrase is recognized;
- *void object.DictationHypothesis (string text)*, called whenever the captured phrase is updated;
- *void object.DictationComplete (Windows.Speech.DictationCompletionCause cause)*, called whenever the dictation is interrupted, where the values of the enumerate *cause* parameter can be the following:
  - *Complete*, if the dictation ended correctly;
  - *AudioQualityFailure*, if the dictation stopped due to bad audio quality;
  - *Canceled*, if the dictation was canceled or the application was paused;
  - *TimeoutExceeded*, if the expected capture time was exceeded;
  - *NetworkFailure*, if the dictation was interrupted due to a network failure;
  - *MicrophoneUnavailable*, if the dictation was interrupted because no microphone was detected;
  - *UnknownError*, in any other case.
- *void objeto.DictationError (string error, int result)*, called when an error is detected during the dictation.

In all cases, the listener functions must be associated to the corresponding attribute of the *DictationRecognizer* object using the sentence *object.attribute += function*.

E. Augmented reality

Finally, students are tasked to train an application to recognize specific visual elements and use them in augmented reality [12]. To do so, they must simply add the *vuforia-unity-xx-yy-zz.unitypackage* development package to a Unity project and include the *ARCamera* prefab object, which replaces the main camera of the scene (*MainCamera*). Afterwards, students must add a database that relates objective images, to be captured by the camera of the device, with 3D objects to be shown on the scene.

F. Evaluation signature

The evaluation signature of the subject is defined to take into account each of the aspects that are important in this work:

- **Models used in the scene –** Scale: Simple, Geometrical, Complex without animation, Complex with animation, Also including scenery: terrain, skybox, ...
- **Complexity of the actions of the characters –** Scale: The characters only move through the game, The characters move and interact with elements of the scene, Some character interacts in different ways with some of the objects of the scene, All of the former and some character interaction produces special effects, such as particle systems.
- **Variety of characters –** Scale: Static objects along with the player, Dynamic objects such as enemies, All of the former and several types of non-static characters other than the player, All of the former and the behaviour of each type of character towards the player is different.
- **Originality –** Scale: Low, Medium, High
- **Development complexity –** Scale: The code is exclusively based off prefabs from the asset store and the standard asset package, Along the prefabs the student implements update scripts over at least two objects in the scene, All of the former and Unity component events are implemented, an event-based GameController is implemented based on events including user-defined events.
- **Code quality –** Scale: Low-quality undocumented code with no logical object hierarchy, Properly documented code, Properly documented code with a coherent collection of objects, All of the former plus whether properties and methods should be public or private is considered and the efficiency regarding performance in mobile phones is taken into account.
- **Immersion level –** Scale: Low, Medium, High
- **Interaction with Cardboard –** Scale: Only head motion interaction is considered, The magnetic button is used, All of the former and a Gamepad or another interaction method is used.
- **VR interaction quality –** Scale: Head motion is followed at all time, Grid recommendations are followed, other VR application recommendations are followed to avoid discrepancies between physical perception and the virtual world.
- **Teamwork –** Scale: The report only includes unconnected tasks, The report shows both tasks and a work discussion, The report shows tasks and several work discussions.
- **Individual work –** Scale: The student shows a low grasp of VR application development in their individual answers: they mistake concepts, they cannot answer accurately, they do not identify code segments; The student shows an average grasp of VR application development in their individual answers: they mistake advanced VR concepts, they cannot accurately answer advanced VR questions; The student shows a high grasp of VR application development: they answer advanced VR questions accurately, they correctly identify all implemented scripts.
- **Other sensors –** Scale: No other sensors are used, Simple usage of microphone or accelerometer, Complex usage of

either microphone or accelerometer or simple usage of both, Complex usage of both sensors.

III. Results y conclusions

The practical lessons of the "Intelligent Interfaces" subject allow students to creatively develop their own natural interface applications, by using a combination of cell phone sensors. These exercises introduce them to the code required to use the device compass, accelerometers, global positioning system, camera and microphone, in virtual or augmented reality applications. These elements are the building blocks of several design techniques for natural interfaces, such as speech and image recognition, that can be used in the development of complex adaptive applications.

The main obstacle resides in the intrinsic difficulty of the usage of technologies that are still evolving, since their frameworks change between terms or are yet unavailable for several kinds of devices. However, the main advantage the presented methodology brings is the motivation that students feel towards the usage of these technologies, by applying production tools to create functional prototypes in a very short time.

At the end of the 2017/2018 term, all 37 enrolled students were requested to share their opinion regarding the subject in an on-line survey. Their responses are shown below:

- Which of all of the projects and exercises of this subject do you consider to have helped your formation the most?
  - Seminars/debates (0%)
  - Oral project presentation (7.14%)
  - Laboratory exercises (92.86%)
- What difficulties did you face when carrying out the projects/exercises?
  - Lack of information (23.08%)
  - Lack of knowledge regarding which sources to use (23.08%)
  - Lack of guidance provided by the subject teachers (7.69%)
  - The work load was too high (0%)
  - Amount of time given for development and delivery (7.69%)
  - Difficulty understanding the meaning of the given tasks (23.08%)
  - Teamwork difficulty (15.38%)
- Grade the following exercise concepts from the point of view of learning how to program natural interfaces and sensor-based applications. (1 Inadequate. 2 Hindered learning. 3 Irrelevant. 4 Eased learning. 5 Fundamental.)
  - C# language (4.14)
  - Unity 3D (4.43)
  - Virtual Reality on mobile phones (4.00)
  - Google Cardboard Framework (3.43)
  - Vuforia (3.29)
  - Accelerometer (3.79)
  - Compass (3.69)
  - Microphone (3.93)
  - Camera (4.00)
- Grade the following exercise concepts from the point of view of motivating learning to program natural interfaces and sensor-based applications. (1 Completely discouraging. 2 Somewhat discouraging. 3 Irrelevant. 4 Motivating. 5 Completely motivating.)
  - C# language (3.71)
  - Unity 3D (3.86)
  - Virtual Reality on mobile phones (3.86)
  - Google Cardboard Framework (2.86)
  - Vuforia (2.86)
  - Speech recognition in Unity (3.57)
- Grade the following exercise concepts from the point of view of acquiring new knowledge that might be useful in the professional future. (1 No new knowledge. 2 Insufficient new knowledge. 3 Acceptable new knowledge. 4 Noticeable increase in knowledge. 5 Acquired completely new knowledge)
  - C# language (3.71)
  - Unity 3D (4.14)
  - Virtual Reality on mobile phones (4.00)
  - Google Cardboard Framework (3.71)
  - Vuforia (3.64)
  - Accelerometer (3.93)
  - Compass (3.93)
  - Microphone (4.00)
  - Camera (4.00)
- Grade the following exercise concepts regarding the difficulty they entailed when programming natural interfaces and sensor-based applications. (1 Excessive difficulty. 2 High difficulty. 3 No difficulty. 4 Eased the development. 5 Greatly eased the development)
  - C# language (3.46)
  - Unity 3D (3.36)
  - Virtual Reality on mobile phones (3.14)
  - Google Cardboard Framework (2.79)
  - Compass (3.07)
  - Microphone (3.07)
  - Camera (3.29)
  - Speech recognition in Unity (2.86)
- Write any suggestion regarding learning about natural interfaces and sensors programming that you believe would help improve the syllabus. (Among the suggestions that students provided, the following stood out:)
  - Reducing the theoretical load of the subject, favoring a greater amount of practical exercises.
  - Teaching Blender [13] as a modeling tool.
  - Computer equipment more capable of taking the graphical load the exercises require.
  - More emphasis on the explanation of virtual reality on mobile phones.
- Write your personal valuation of learning how to program natural interfaces by developing a virtual reality application for mobile phones.

- Except for very few outliers, the general opinion regarding the subject was highly positive. Students valued its usefulness, since it is the subject of the degree that focuses the most on graphical tools. The claim to have enjoyed developing games, as it allowed them to satisfactorily observe the results of their work.

Students provided mostly positive feedback regarding the mainly practical approach of the "Intelligent Interfaces" subject. They appreciated learning about the C# language and the Unity 3D platform, and felt motivated towards the chance of creating their own virtual reality applications, although they showed some disappointment regarding the Google Cardboard Framework and the usage of Vuforia. A reduction in the practical load of these elements could therefore be proposed, in favor of an introduction to the Blender modeling tool.

References

[1] G. Fischer and B. Reeves, "Beyond intelligent interfaces: Exploring, analyzing, and creating success models of cooperative problem solving," Applied Intelligence, vol. 1, issue 4, pp. 311–332, May 1992, Kluwer Academic Publishers

[2] A. Malizia and A. Bellucci, "The Artificiality of Natural User Interfaces," Magazine Comm. ACM, vol. 55, issue 3, pp. 36–38, March 2012, ACM, New York, NY, USA

[3] T.M. Alisi, A. Del Bimbo and A. Valli, "Natural interfaces to enhance visitors' experiences", IEEE MultiMedia, Vol. 12, issue 3, pp. 80–85 Sept. 2005

[4] R. Francese, I. Passero and G. Tortora, "Wiimote and Kinect: gestural user interfaces add a natural third dimension to HCI," AVI '12 Proc. Int. Working Conf. Advanced Visual Interfaces, pp. 116–123, May 2012, ACM New York, NY, USA

[5] P. Santana, "Interfaces Naturales de Usuario - La Experiencia de la Universidad de Colima," Software Guru, vol. 43.

[6] J. Marco, E. Cerezo and S. Baldassarri, "Desarrollo de interfaces naturales para aplicaciones educativas dirigidas a niños," VIII Congreso Internacional de Interacción Persona Ordenador, pp. 79–82, 2007.

[7] J. Videla, A. Sanjun, S. Martnez and A. Seoane, "Diseo y usabilidad de interfaces para entornos educativos de realidad aumentada," Digital Education Review, vol. 31, pp. 61–79, June 2017.

[8] R. Arnay, J. HernndezAceituno and E. Gonzlez, "Teaching kinematics with interactive schematics and 3D models," Computer Applications in Engineering Education, vol. 25, pp. 420–429, 2017, Wiley Periodicals.

[9] Motor de videojuegos multiplataforma Unity3D (unity3d.com)

[10] Plataforma de realidad virtual Google Cardboard (vr.google.com/cardboard)

[11] Paquete de reconocimiento de voz para Windows10 en Unity3D (UnityEngine.Windows.Speech)

[12] Paquete de realidad aumentada Vuforia para Unity3D (developer.vuforia.com)

[13] Programa de modelado 3D Blender (blender.org)