J Multimodal User Interfaces (2015) 9:105–119
DOI 10.1007/s12193-014-0172-1

ORIGINAL PAPER

Multi-modal natural interaction in game design: a comparative analysis of player experience in a large scale role-playing game

Pedro Alves Nogueira · Luís Filipe Teófilo · Pedro Brandão Silva

Received: 13 November 2013 / Accepted: 9 December 2014 / Published online: 22 January 2015

Abstract Previous work on player experience research has focused on identifying the major factors involving content creation and interaction. This has encouraged a large investment in new types of physical interaction artefacts (e.g. Wiimote™, Rock Band™, Kinect™). However, these artefacts still require custom interaction schemes to be developed for them, which critically limits the number of commercial videogames and multimedia applications that can benefit from those. Moreover, there is currently no agreement as to which factors better describe the impact that natural and complex multi-modal user interaction schemes have on users' experiences—a gap in part created by the limitations in adapting this type of interaction to existing software. Thus, this paper presents a generic middleware framework for multi-modal natural interfaces which enables game-independent data acquisition that encourages further advancement on this domain. Furthermore, our framework can then redefine the interaction scheme of any software tool by mapping body poses and voice commands to traditional input means (keyboard and mouse). We have focused on digital games, where the use of physical interaction artefacts has become mainstream. The validation methods for this tool consisted of a series of increasing difficulty stress tests, with a total of 25 participants. Also, a pilot study was conducted on a further 16 subjects which demonstrated mainly positive impact of natural interfaces on player's experience. The results supporting this were acquired when subjects played a complex commercial role-playing game whose mechanics were adapted using our framework; statistical tests on the obtained Fun ratings, along with subjective participant opinions indicate that this kind of natural interaction indeed has a significant impact on player's experience and enjoyment. However, different impact patterns emerge from this analysis, which seem to fit with standing theories of player experience and immersion.

Keywords Multi-modal · Natural interfaces · Videogames · Kinect · Game controller · Pose recognition

P. A. Nogueira · L. F. Teófilo (✉) · P. B. Silva
Department of Informatics Engineering, FEUP,
Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias,
s/n, 4200-465 Porto, Portugal
e-mail: l.teofilo@fe.up.pt

P. A. Nogueira
e-mail: pedro.alves.nogueira@fe.up.pt

P. B. Silva
e-mail: pedro.brandao.silva@fe.up.pt

P. A. Nogueira· L. F. Teófilo
LIACC, Artificial Intelligence and Computer Science Lab,
University of Porto, R. Campo Alegre, 1021,
4169-007 Porto, Portugal

P. B. Silva
INESC Porto, Instituto de Engenharia de Sistemas e Computadores,
Rua Dr. Roberto Frias, 378, 4200-465 Porto, Portugal

1 Introduction

Videogames and multimedia applications have initially tried to convey increasingly immersive experiences through increased character and environment believability, having in recent years started to dedicate their attention to the interaction artefacts (e.g. the WiiMote™, Kinect™, Move™ and Guitar Hero's controller) [1]. Traditionally, the player is forced to press an arbitrary button combination, which corresponds to a mapped action in the game world. Often, these combinations are standardized (e.g. using the WASD keys to move the game character in the game world) or rely on cultural conventions ('*R*' key for reloading a weapon, '*F*' key for turning on the flashlight). Controller-type artefacts allow players a physical mean to interact with the game

world. Sometimes, controllers assume a physical shape that resembles the events in the game, such as the Rock-Band™ instruments (musical instrument like controllers for a music playing game). Otherwise, natural interaction devices (e.g. Kinect™, voice recognition) allow players to control their avatars by acting as if they were actually performing the task within the game.

Despite natural interaction devices lacking physical artefacts, they allow players a wider range of interaction methods. They also offer the possibility of integrating them in the natural interaction, thus proving a powerful tool for interaction research.

An awarded example of this physical medium/natural interaction fusion is the interactive virtual-reality environment Osmose, by Char Davies [2]. In her experiment, Davies merged a kinaesthetic interaction scheme with traditional virtual reality technologies (a head-mounted display and 3D surround sound) to provide the physical medium. In her own words: "(Osmose) shuns conventional hand-based modes of user interaction, which tend to reduce the body to that of a disembodied eye and probing hand in favour of an embodying interface which tracks breath and shifting balance, grounding the immersive experience in that participant's own body" [2]. Davies' results show that some test participants had strong emotional reactions to the whole experience, suggesting that applications reporting high immersion levels (e.g. videogames), coupled with suitable kinaesthetic interaction schemes can drastically increase the enjoyability and sense of emotional engagement of said application. The creation of applications resorting to natural (or kinaesthetic) interaction thus enables more engaging experiences that, in turn may capture the interest of a larger audience and facilitate player engagement along the four factors proposed by Lazzaro [3]:

- Hard Fun: Enjoyment derived from overcoming complex challenges.
- Easy Fun: Enjoyment derived from high immersion levels though a game that provides full cognitive absorption.
- Altered States: Enjoyment obtained through one's internal reactions to the game's visceral, behaviour, cognitive or social aspects.
- The People Factor: Players use the game as a mechanism for social experiences, such as competition, team work or social bonding.

Regardless of the increased investment by game designers in crafting more engaging experiences, the effects of natural interaction in these types of experience has not yet been thoroughly researched. One of the reasons behind this gap is that, although multiple natural interaction techniques exist, there is also a wide lack of adequate development and prototyping tools for multi-modal natural interaction. Since introducing new controllers, sensors or peripherals requires, in most cases, a direct intervention on the source code, the experimentation process becomes difficult if not impossible to achieve. An alternative consists in handling the input of the new devices externally and afterwards map them to supported ones, such as keyboard and mouse.

We have developed such a solution in our own multimodal natural interface framework to quickly define and test the impact that multi-modal natural interaction schemes (NIS) had on players' user experience (UX) ratings. In the study presented in this paper we were interested in combining natural (body), speech and physical artefact interaction, so the framework was built to support these types of devices. Despite this, in hindsight of the limitations of current work, it was developed with a modular design to facilitate the future inclusion of additional devices.

Summarising, this paper's contributions are threefold:

1. Development of a framework capable of facilitating testing novel interaction schemes in an expeditious fashion, while merging multiple input modalities transparently;
2. Validate the aforementioned framework in respect to its usability and accuracy;
3. Investigate whether augmenting traditional gameplay interaction methods with this new kind of multi-modal natural user interaction techniques has a significant impact on user experience ratings and preference ratings.

Throughout this paper each of these contributions are discussed in detail. Thus, the paper is structured as follows: Sect. 2 discusses the related work in the areas of natural interaction methods, movement detection and speech recognition. Sections 3 and 4 rather briefly describe the framework's conceptual design and individual component implementation, respectively. Section 5 presents the framework's validation regarding its pose and speech recognition accuracies, as well as usability tests. Section 6 compares a custom-designed augmented interaction scheme to a traditional one in a case-study commercial videogame (Skyrim) by analysing players' user experience and preference ratings of both versions of the game. Finally, Sect. 7 presents our final conclusions, along with relevant future work.

2 Related work

2.1 Natural interaction methods

As previously mentioned, traditional interaction models in videogames resort to button combinations, implemented through keyboard and mouse schemes, only recently shifting to dedicated and natural controllers. However, users are still limited to the designed (native) interaction scheme (IS), not being able to redefine or change it altogether. The most

well-known and complete existing solution is the Flexible Action and Articulated Skeleton Toolkit (FAAST[1]), which is a middleware software solution that aims at simplifying the integration of full-body controls with games and virtual reality (VR) applications [4]. FAAST is able to detect manually defined poses and map each one to a set of keyboard or mouse inputs. The most recent version of FAAST introduces gesture recognition by following the approach on [5] by setting spatial joint constraint rule sets and extending these with temporal constraints which indicate how much time is needed for a pose to be recognized. The main issue of this framework is that the task of specifying gestures has to be performed manually, in a trial-and-error approach, becoming potentially time-consuming especially if the system is to be used by several different users—it must be configured individually to each user. FAAST, despite its usefulness only lies on its restriction to skeleton detection sensors—it is not multimodal. However, as we explain and demonstrate throughout this paper, several tasks in games are currently very difficult to perform by just using body motion detection. One good example is moving forward or backwards—which is unfeasible since it would require very large rooms, and body position detection at larger distances.

This lack of a standard and usable framework for the deployment of natural interaction schemes leaves individuals researching them with two options. To resort to a Wizard of Oz approach, simulating a non-working prototype [6], which in many cases is not possible (e.g. playing a game or most real-time activities), or to build his own custom solution from the ground up [7,8]. The latter is often the only available approach, requiring a huge commitment in terms of time and effort, while also limiting this research field to people versed or with access to people versed in computer science. Additionally, it also stifles the growth rate of the field and its adoption by the public, contributing to a decrease in popularity.

2.2 Movement detection

Recent approaches in reliable movement detection have introduced marker-based systems [9], accelerometers [10, 11], physiological sensors [12] and carbon-based strain measurement [13]. While these systems are, in general, accurate they are expensive due to the necessary complementary dedicated hardware; and intrusive, by requiring the user to wear the sensors or markers. Some of them also do not measure all of the relevant motions (e.g. strain sensors often do not measure torsion) or provide enough accuracy (e.g. cell phone-grade accelerometers).

With the introduction[2] of the Kinect™, movement detection has become cheap and unobtrusive, alas with some inaccuracy as some of our preliminary tests showed a Gaussian fluctuation of nearly 7 cm on the X and Y planes when the subject was idle. Nevertheless, its inexpensive hardware, open source SDK and wide availability encouraged its use in this study. Despite providing a spatial representation of the user's skeleton, the Kinect™ does not support custom pose or movement recognition. This is an issue that has been vastly studied by the scientific community [14–17]. Being a complex problem, most solutions do not work in real-time or have limited tracking capabilities, which motivated the development of the presented lightweight pose detection method.

2.3 Speech recognition

Speech recognition (SR) is also a complex problem with a multitude of approaches [14,18–20]. The main issue with SR is that it requires a database of recognized phonemes and words, which is difficult to create on the fly, as each instance also requires considerable feature extraction and training. Another pressing issue is that it is difficult to identify various sound sources robustly, as well as differentiate from actual sound sources (speakers) and noise. This issue has been tackled by Shih [20], but has yet to be implemented in commercial software. While proving itself resistant to the first issue, the Kinect™ is extremely vulnerable to the two foremost ones at medium distances (∼2 m, the distance required for optimal movement recognition [21]).

Microsoft's Speech API (SAPI) is a widely used package with native support that already provides an extensive database for the English language and features various runtime optimizations, which motivated its choice as our speech recognition engine.

2.4 Other multimodal frameworks

Our work focuses on multimodal frameworks and their applicability for conceiving natural interfaces to increase usability in computer games. However, despite their popularized use in games, the applications of natural interfaces are far more comprehensive. One notable example is the OpenInterface platform.[3] OpenInterface [22] is a system that allows for specifying complex interaction schemes through flow diagrams. Examples of OpenInterface applications include 3D exploration of anatomic models, audio software control or image navigation. Open Interface does not natively supports body motion detection. This support can be provided

[1] When we cite the FAAST framework throughout the paper we always refer to the version available on early 2012.

[2] The Kinect was introduced by a tech-demo game called MILO project in 2009, which explored a completely new interaction paradigm.

[3] OpenInterface is available at http://www.openinterface.org/.

by the third party software like KOI[4] (Kinect Open Interface) which allows for the recognition of 11 gestures. Considering the development of games' natural interfaces, the main issue about OpenInterface is that the definition of interfaces is too complex to easily adapt them to computer games. Moreover, the kind of mechanics that usually games present is very different from other applications, where the user usually impersonates an avatar that interacts with the virtual world.

Natural interfaces can also be extended by affective computing, by using the recognition of human emotions to control games or other software systems. One such example is the *Semain* API [23] which provides a complete framework to develop emotion-oriented systems.

2.5 Related works

In [24] the authors conduct a study using a commercial videogame (Mario Kart Wii) designed to measure the effects of natural interaction techniques in player experience. In their empirical study, they found that the non-natural interaction techniques significantly outperform their more natural counterparts. This effect can, hypothetically, be caused by several related factors:

- Bad game mechanic design (i.e. the match between the game actions and their natural interaction techniques were mismatched) [24].
- Latency, the temporal delay between user input and the system feedback. These can lead to overcorrections of initial inputs which, due to the time lag, fluctuate much more wildly than the more responsive non-natural input [24].
- Player's lack of expertise in using the natural interfaces, which in turn might have been amplified by their bad design, thus increasing the learning curve.

In [25], the authors developed and tested a hands-free videogame that used the player's facial positioning and expressions to steer a drunker game character through a series of obstacles. In their study, the authors found that although the non-natural interaction was rated higher in terms of input preciseness, the natural interaction variant of the game enhanced the overall positive aspects of the user experience. Whether this was due to a novelty effect is not analysed by the authors, but hinted at a possible factor in the analysis, which requires a long-term study. The authors also suggest that a high level of player arousal may be a key factor in more positive player experiences.

A multimodal multiplayer tabletop gaming system is presented in [26], where players are able to play several commercial strategy games using voice commands and multi-finger tactile inputs.

A similar study to [24] is presented in [27], where players' emotional experience derived from questionnaires, interviews and player action logging were compared between three different input modalities. This study reported only motion control (i.e. no speech control schemes) and report similar findings to [24], with the addition that the novel interaction schemes seem to play an important role in encouraging more inexperienced players to interact with the system; perhaps due to a novelty effect or a contrast with more experienced players who prefer to continue using the already learned (non-natural) scheme.

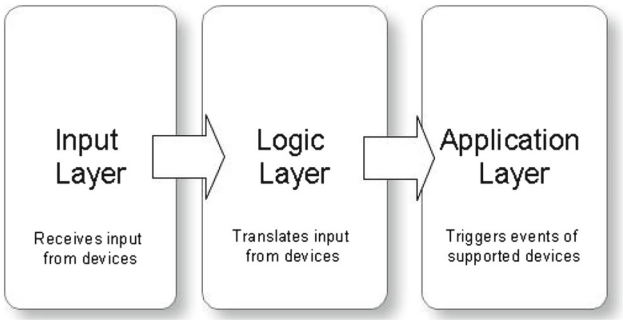

Fig. 1 GeMiNI's conceptual architecture

3 Conceptual framework description

The Generic Multi-Modal Natural Interface (GeMiNI) is a framework meant to support an easy introduction and configuration of any computer compatible peripheral device to work as a game input. It acts as an abstraction layer between device events and the game's default controls. This allows users to experience new interaction methods not originally supported or even been devised by the game's developers. As an example scenario: a first-person shooter game, designed for mouse and keyboard input, could be enhanced using voice commands to trigger actions such as issuing orders to squad members or body poses for crouching, walking or setting traps.

GeMiNI's architecture is conceptually composed of three layers, as depicted in Fig. 1. First, the input is captured on the input layer, generating an event type. Different devices with different software drivers (for example, cameras) may output the same type of event (e.g. a captured video frame). The logic layer then translates these into commands that are recognized by the game, according to a user defined IS. More precisely, this is achieved by mapping each event to the game's original controls (e.g. keyboard shortcuts). Lastly, the application layer is responsible to assure that, while in-game, the game actions are invoked when the corresponding events are triggered.

[4] Kinect Open Interface is available at http://koi.codeplex.com/.

4 Implementation

The GeMiNI framework is not conceptually restricted to any specific input device. Still, addressing all possible interaction technologies has neither been considered feasible nor relevant at this point of our research. Instead, as a first step, three kinds of device technologies and corresponding events have been considered:

- Microsoft Kinect camera, for detecting body poses.
- Embedded microphones for capturing voice commands.
- Nintendo Nunchuk for capturing auxiliary key inputs.

For this first approach, these technologies were deemed diverse enough to generate a wide variety of IS. Also, bearing in mind their popularity, maturity and affordability, they were considered the ones where gamers would be more familiarized with and eager to experiment on.

Bearing in mind that the employment of these devices poses challenges by themselves, it is important to address more technical details behind their architecture, operation and integration within the framework concept.

The language of choice for development was Microsoft .NET C# 4.0, due to its ease of development and the quality and availability of APIs to deal with the proposed devices.

The next section addresses more technical details behind the integration of said devices with the proposed architecture's concept and operation.

4.1 Architectural integration

The integration of the three interaction device technologies has been performed according to the conceptual architecture explained in the previous section (see Fig. 2). Furthermore, to support an easy addition of new features, each layer is divided into independent modules. For each new device, a component has been built, which makes use of a specific, of-the-shelf driver or library to receive and transform the device input into the mentioned event categories.

For the implemented devices, three components have been developed in the input layer (see Fig. 2).

- The Skeleton Module uses version 1.5 of the Microsoft Kinect SDK to process a human skeleton structure from the Kinect camera input, and feeds it to the Pose Recognizer module in the logic layer.
- The Speech Module uses the Microsoft SAPI to capture audio from the microphone, and feeds it to the Speech Recognizer in the logic layer.
- Nunchuk Module resorts to the open-source library WiimoteLib and feeds Nunchuk inputs to the general Input Manager in the logic layer.

The logic layer then processes the input events using the following components:

- Pose Recognizer: Uses our algorithm based on predefined spatial constraints to detect skeleton poses, and passes an identifier of the pose to the Input Manager;
- Speech Recognizer: Uses the Microsoft SAPI to recognize a designated vocabulary from the audio input, and sends the identified words to the input manager;
- Input Manager: Processes specific identifiers, such as corresponding to keystrokes (from the Nunchuk), to pose (from the pose recognizer) and pronounced words (from the speech recognizer). Then it translates these identifiers to specific game actions, according to a predefined mapping description.

Lastly, the application layer is connected to the logic layer and is composed of two distinct modules: The configuration GUI, which allows an expeditious and intuitive configuration of all the necessary parameters of the architecture's components through a graphical user interface; and the external application, which is executed simultaneously with the framework and reacts to the translated GeMiNI events.

4.2 Architectural components

In this sub-section we describe the various components that constitute the proposed framework by outlining their features, performance ratings and possible limitations.

4.2.1 Pose recognition

The pose recognition module of our framework is responsible for assessing a series of spatial constraints between body joints in order to identify one or more poses configured by the framework's user. The body joints data (Cartesian position) is provided by the Kinect cameras contains the position of 20 distinct points (or joints) from the detected human body (see Fig. 3). Each pose identification is exclusive i.e. several poses can be identified at the same time. To illustrate this, consider the following example of poses:

- *Punch Pose* : this pose can be easily defined by introducing a constraint that the hand joint is in front of (Z coordinate value is higher) the elbow and the shoulder joints.
- *Crouch Pose* : this pose can be defined by introducing constraints that the Hip joints are bellow (Y coordinate value is lower) the Knee joints.

The simultaneous identification of the example poses could be useful for a fighting game: the user can make the avatar give a punch, crouch to avoid opponents or give a

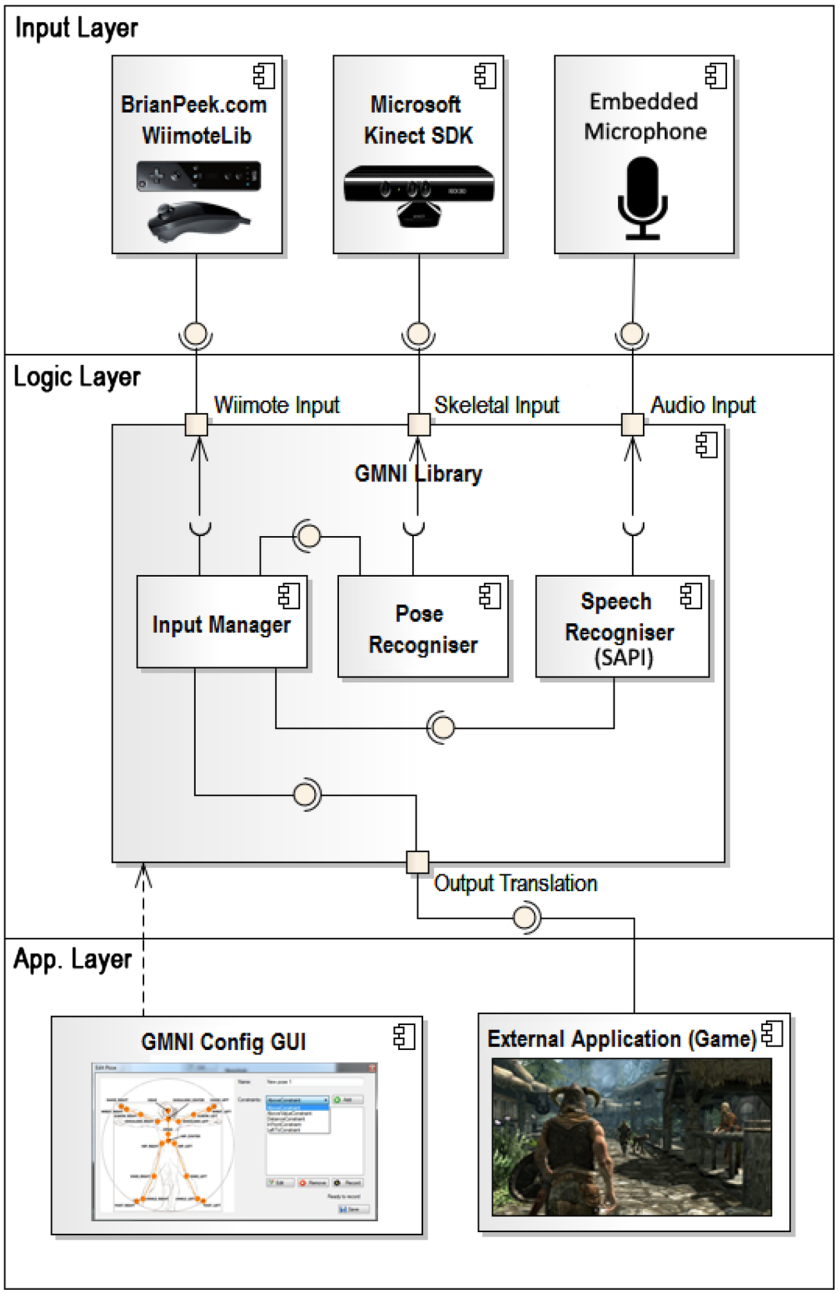

Fig. 2 GeMiNI implemented architecture

punch while crouched. Moreover, since the constraints (as explained later) are expressed in relative measures between joints, the detection of these poses does not depend on the absolute position of the user and his or her body dimensions and, therefore, not requiring individual system training for each particular user.

Each point is imbued with a semantic identifier indicating the body part and a spatial reference in three-dimensional Cartesian space, relative to the cameras. This introduces the possibility to perform queries concerning the relative location of any body part towards another and its 'absolute' location with the Kinect as the origin. The used coordinate system is shown in Fig. 3.

Kinect supports several users at the same time. The information about the body joint position of a given user provided by the Kinect API is temporally tagged. Thus, at any moment for user each joint position $\vec{p}$ can be defined as $\partial_{Id,t,u} = \vec{p} = (p_x, p_y, p_z) \in \Re^3$ and where Id is the semantic identifier of the body part and can be one of the following: {'SPINE', 'HEAD', 'WRIST_RIGHT', …} (see Fig. 3).

Saving a pose by specifying precise absolute location of a joint is neither handy nor feasible, since that it greatly limits possibilities and oversees variations that occur naturally in body posing. Instead, in this approach, a pose P is specified through a finite set of combinations of n spatial constraints $\bar{c} := (c_1, c_2, \ldots, c_n)$, which are verified on body member

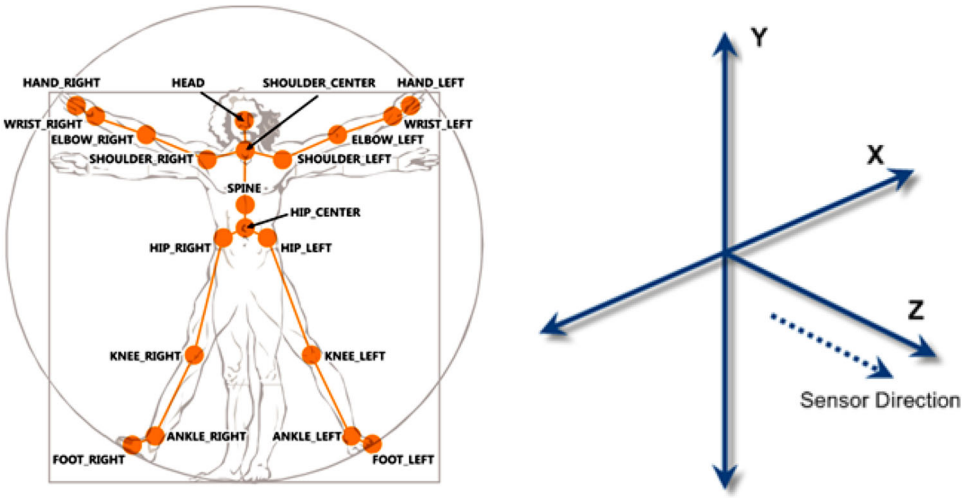

Fig. 3 *At left*: player body joint disposition recognized by the Kinect SDK. Taken from: http://i.msdn.microsoft.com/dynimg/IC534688.png. *At Right*: Kinect Camera coordinate and axis system. Adapted from: http://i.msdn.microsoft.com/dynimg/IC534689.png

dispositions. To recognize a pose, all constraints must be verified (e.g. for n = 3 the proposition $c_1 \wedge c_2 \wedge c_3$ must be true).

One example of pose is to verify if someone is standing on one foot. For this, it is only necessary to consult the Y-coordinate values of the feet joints to check for any prominent differences. Another example: to check if the hands are touching, it is only necessary to calculate the distance between them. This approach allows for a greater flexibility on defining poses.

Our framework recognizes poses in a given instant of time. To simplify the description of the implemented constraints, let us consider a fixed user u_k in a fixed instant time t_c being therefore $\vec{w}_{Id} = \partial_{Id,t_c,u_k}$ the position vector for that user and that instant for a given body joint.

The following constraint types were implemented:

- *Distance*: Imposes minimum and the maximum Euclidean distance between joints. Let two joints with *Id* equal to A and B where $A \neq B$ and the minimum distance as *min* and the maximum distance as *max*, the constraint that must be satisfied is:

$$\min \leq \|\vec{w}_A - w_B\| \leq \max$$

- *InFront*: Defines whether a joint is in front of another, by comparing the values of the Z coordinates. Let A and B be two distinct joints. If we want $A$to be in front of B, the constraint that must be satisfied is:

$$\vec{w}_{Az} \geq \vec{w}_{Bz}$$

- *LeftTo*: Defines whether a joint is to the left of another, by comparing the values of the X coordinates. Let A and B be two distinct joints. If we want $A$to be to the left of B, the constraint that must be satisfied is:

$$\vec{w}_{Ax} \leq \vec{w}_{Bx}$$

- *AboveOf*: Checks if a joint is located above another, by comparing the values of the Y coordinates. Let A and B be two distinct joints. If we want $A$to be above B, the constraint that must be satisfied is:

$$\vec{w}_{Ay} \geq \vec{w}_{By}$$

- *AboveValue*: Checks if the Y coordinate of a joint is located above a certain threshold. Let A be a joint and C the constant threshold value. The constraint is satisfied if:

$$\vec{w}_{Ay} \geq C$$

This constraint can be similarly implemented for the other axis.

- *AngleRestriction*: Imposes minimum and maximum angle between two joints. Let A and B be two distinct joints, $\theta_{\min}$ the minimum angle and $\theta_{\max}$the maximum angle. Thus, the constraint is satisfied if:

$$\theta_{\min} \leq \cos^{-1} \frac{\vec{w}_A \cdot \vec{w}_B}{|\vec{w}_A| \cdot |\vec{w}_B|} \leq \theta_{\max}$$

In order to calibrate the pose detection, each constraint can be configured by fine-tuning the corresponding parameters. Still, single constraints may lead to unwanted body pose recognitions since they impose rather broad definitions. In order to reduce this ambiguity it is possible to resort to more than one constraint simultaneously, so as to define a pose. It is important to stress that this constraint-based approach has proven to be fast, so as to allow real-time detection, with an approximate 16–25 ms delay, maintaining the game's frame rate.

4.2.2 Pose recording

The correct choice and parameterization of the spatial constraints, to define a certain pose, can become a difficult endeavour. For this reason, an alternative to this manual trial-and-error approach has been created. GeMiNI provides completely automatic constraint and parameter definition though pose recording. The Kinect skeleton feed was used to analyse the body's main motion axes during a short (5 s) training phase and infer the relevant constraints and parameters. The method works by the set of joints located in the body's main motion axes (knees, feet, spine, neck, hands and head). It then applies a peak removal method through a median filter over a 1.5 s sliding window (0.5 s overlap). This is done to remove random fluctuations from the recorded signal, as we found the Kinect camera has $\sim$7 cm fluctuations on the X and Y planes. The method proceeds to analyse the relations between each possible (distinct) joint pair. For each of these joint pairs, we consider all the possible linear combinations of axes (i.e. X, Y, Z, XY, YZ, XZ and XYZ) and consider that a relevant movement was seen if the maximum observed difference between the joint pair in the recorded data is bigger than a set threshold (15 cm in our experiments). Finally, the restriction set is defined as the relevant axis combinations in the analysed joint pairs.

4.2.3 WiiMote communication

The use of the WiimoteLib to access the Nunchuk inputs has greatly eased the key interpretation, releasing from the need of additional processing in a separate logic layer component. The choice of a Nunchunk driver implementation was motivated by its button diversity (it has both normal buttons and a D-Pad), popularity and compact form. This was considered a good practical alternative for 2D movement or camera control in 3D applications.

4.2.4 Speech recognition

Speech recognition features were implemented, allowing any (pronounceable) word or sentence recognition. The system works at a maximum optimal distance of 2 m and performs speech recognition with a 1–2 s delay. Issues found with SAPI included various user identification and noise cancellation, with some sounds from the environment sometimes being misinterpreted (false positives) as voice commands.

4.2.5 Input management and simulation

The defined poses, speech commands or external game device outputs can be mapped to a combination of both mouse and keyboard events. Regarding keyboard invocations, there are three event possibilities:

- Key press, corresponding to a single keystroke;
- Key hold, equivalent to holding the key down for certain period;
- Repeated key press, describing a series of repeating single keystrokes with a certain, configurable frequency.

Likewise, mice controls can be simulated in the following manners:

- Movement, simulating horizontal and vertical displacements;
- Button press and hold, following the keyboard example.

The distinction between button *press*, *hold* and *repeated press* are important to address the nature of the intended activities. Certain actions, such as "Toggle Inventory" or "Interact with Object" are executed by pressing the intended keys once. When the corresponding pose is detected, the key press is simulated once and only repeated if the users quit the pose and form it again.

Other actions, like "Shoot" or "Raise Shield" are performed while a key is held down. Similarly, the key hold simulation starts when the user's pose is detected, being only stopped when the users' pose is changed.

Finally, actions such as "Swing Weapon" or "Punch" represent examples that typically require one key stroke per execution, but that must be rapidly sequenced in order to be effective. While such a "manual" approach can be very easily accomplished using keyboard or mouse buttons, executing the same behaviour in such a fast manner is harder—if not impossible—to achieve by actually performing the corresponding poses. For these cases, the repeated press simulation allows the pose's mapped input to be triggered automatically with a user-defined frequency, should the player maintain the pose (e.g. the game character will continuously punch the enemy while the player holds the "punch" pose).

4.2.6 Advanced GUI

Configuring the framework to support new constraints, devices and key mappings requires still a considerable amount of work. While at a first step simple configuration files were used, a more intuitive and powerful method was considered necessary, especially to integrate with the pose recording features. Having that in mind, a graphical interface was designed using Windows Forms (Fig. 4), introducing a simple, yet more effective mean for performing experiments on different test cases, as well as users.

Among other features, this GUI introduced the following facilities:

- Creation of "projects" to address different game configurations;

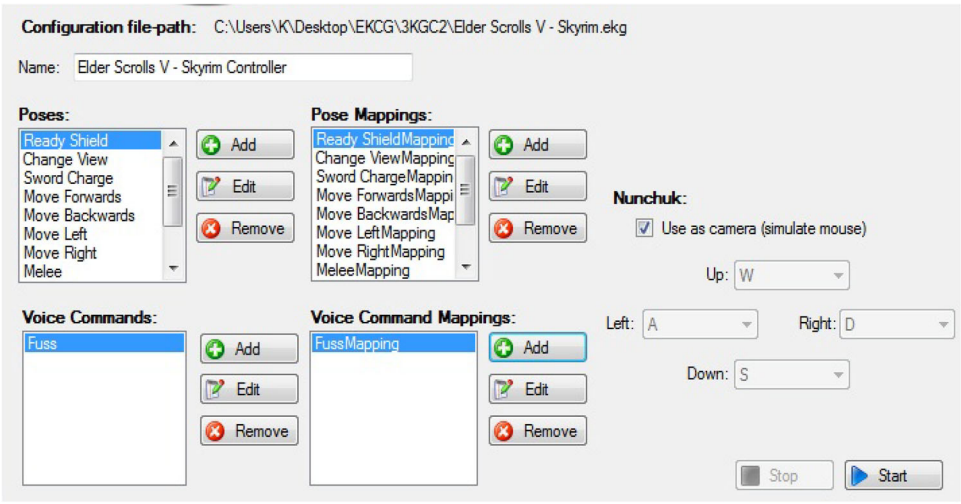

Fig. 4 The GMNI graphical user interface

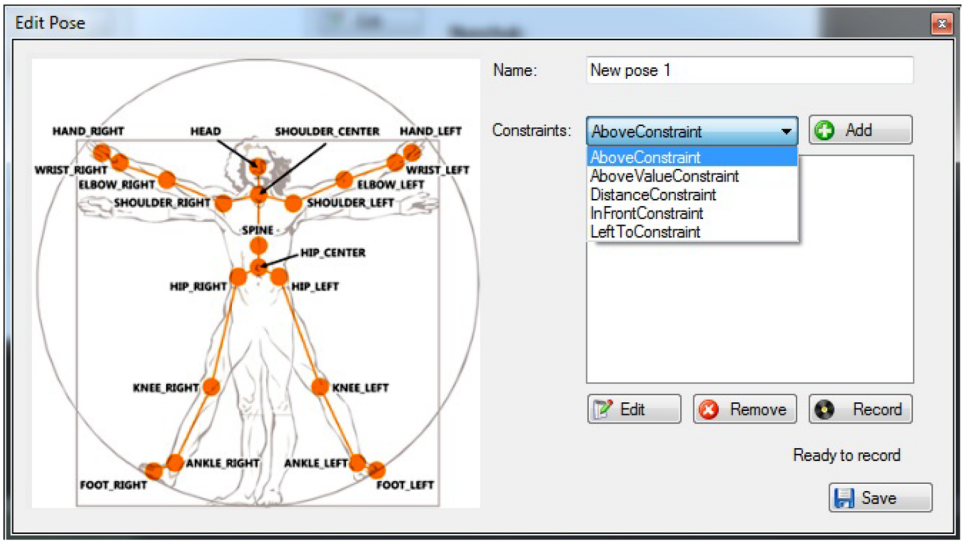

Fig. 5 Example of GUI window for defining poses

- Creation of poses though manual and automatic definition and parameterization of the spatial constraints (see Fig. 5);
- Definition of voice commands;
- Mapping of Voice Commands and poses to keyboard events;
- Mapping of Nunchuck control keys to keyboard and mouse events.

5 Tests and validation

For the accuracy and usability tests presented in this section, 25 participants with ages between 18 and 27 years, 76 % male and 24 % female with no known physical or mental limitations were recruited. Out of these 25 test subjects, 40 % were casual gamers, while the remaining 60 % were hard-core gamers. Players were considered casual gamers if they reported playing videogames less than 1 h per day or only whenever big titles were released. On the other hand, players that reported playing an average of 3 or more hours per day were considered hard-core gamers. In regards to NUI proficiency, ∼50 % of all participants reported being "knowledgeable of NUI techniques" on a 4-level Likert scale forced-choice questionnaire. An interesting aspect was that casual gamers reported a higher proficiency (70 %) with NUI, than hard-core gamers (40 %).

5.1 Pose detection and inference accuracy

All 25 test subjects were asked to perform 20 designated poses, which were recorded and automatically inferred by GeMiNI. Poses were previously enacted and verbally described to the participants, so as not to condition their interpretation of them. Participants were then asked to re-enact each one of these poses ten times, to measure the detection accuracy. The poses' inference accuracy was also considered in these tests. The inference of a pose is the process of

Table 1 Average pose accuracy detection and inference

Pose	Detection rate (%)	Inferring rate (%)
Step forwards/backwards	97	95
Lean left/right	81	90
Left/right punch	99	100
Lift left/right leg (kick)	97	100
Jump	96	96
Raise left/right arm	100	100
Grabbing motion	98	96
Crouch	76	92
Flex	93	100
Point a bow	94	94
Hands behind shoulders	86	87
Outstretched arms	95	96
Lean forwards/backwards	86	87

automatically determining the constraints that characterize a given pose. We considered a pose to be well learned if, after the inference process, it is detected with an accuracy of 80 %, or more, on subsequent repetitions. This detection threshold was empirically defined as the minimum usability baseline, since we found participants to report frustration when interacting with a configuration that had poses bellow this accuracy threshold. Each pose was repeated 20 times by each participant. Overall results for the test population are depicted in Table 1.

5.2 Speech recognition

For this test, 50 words were sampled *iid* from the game's item inventory list. This was done in order to accurately portray the game's verbal command complexity. Since a voice command (e.g. spell, town name, weapon name, etc.) is usually between one and three words, the previously sampled words were used to generate a set of 50 random sentences between one and three words. This sentence set was then segmented into three complexity categories according to the number of syllabi contained in each sentence. Thus, these categories represent the simplest to most complex voice commands possible in our test scenario.

Each test subject was asked to repeat 15 random sentences from each category in order to test the speech recognition's accuracy. Each sentence was repeated five times. A sentence was considered detected if it was correctly identified 60 % or more of the time (i.e. 3 out of 5 times). The tests were performed at a distance of about 2 m—the optimal distance suggested by the Kinect SDK [21]—from the microphone in a quiet room. The accuracy for this task is depicted in Table 2.

Table 2 Speech recognition accuracy results

Syllable count	Single words (%)	Sentences (%)
1–3 Syllables	78	89
3–5 Syllables	83	93
6–8 Syllables	NA	94
8–10 Syllables	NA	93

Table 3 Task completion times for the overall population

Task	Mean (s)	Standard deviation
New pose (manual)	46.4	14.7
New pose (auto)	17.4	5.2
New voice command	8.6	3.5
Add simple action	13.7	4.8
Add complex action	32.6	13.3
Set Wiimote button	4.1	2.3

5.3 Usability testing

To evaluate GeMiNI's GUI usability, each of the participants was asked to perform a series of tasks (see Table 3) that represented each of the previously mentioned steps involved in defining a new IS. The mean values and standard deviations for each action's completion times were calculated, as well as the total number of errors performed by the subjects. The completion times are present in Table 3, while the number of errors committed per task are depicted in Fig. 6.

6 Case study—Bethesda's Elder Scrolls V: Skyrim

Having validated the developed framework, our interest lied in asserting whether using multi-modal NUI techniques had a significant impact on players' user experience and preference ratings. Thus, a pilot study with a new population of sixteen participants with no previous knowledge of the video game's commands was conducted. Participants' were rated as hard-core or casual gamers according to their gaming proficiency within our case study's specific game genre. Exactly 50 % of the participants were hard-core gamers, with the remaining 50 % being casual ones. Each participant played the game with the native and the natural interaction scheme. The playing order was randomized to avoid order effects. Afterwards, players were asked to answer a brief forced-choice questionnaire regarding their user experience on each gaming condition. Additionally, players were also asked to rate the gaming conditions in terms of their overall preferences.

Despite having developed interaction schemes for various popular case study games in the literature (e.g. Super Mario, Legend of Zelda and Half-Life), due to space limitations, it

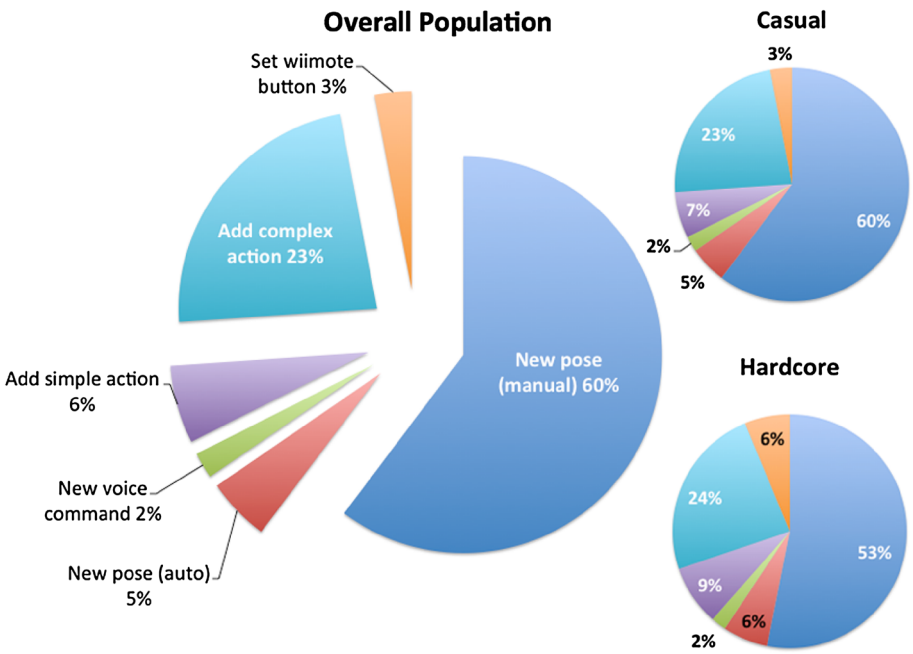

Fig. 6 Error distributions for the tested interaction tasks. The two smaller charts on the right examine the error distributions in detail across the two player types

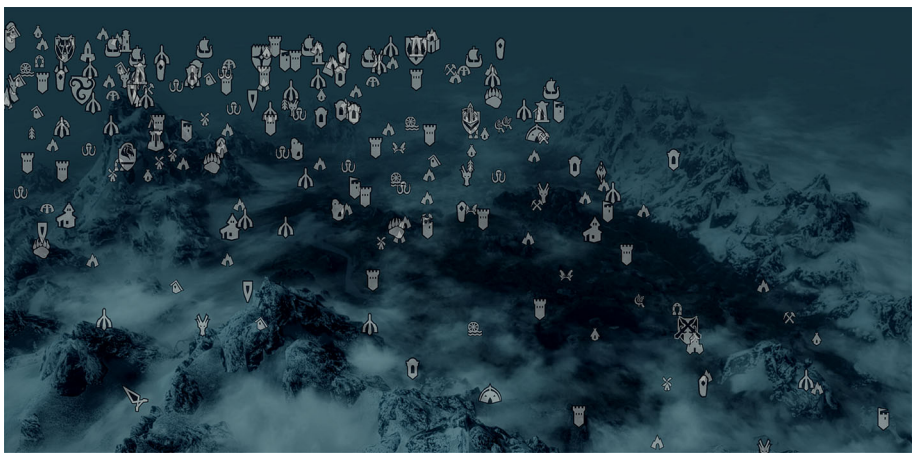

Fig. 7 Zoomed view (east region) of Skyrim's world map. *Each icon* represents an explorable location of varying size. The average time taken to explore one location ranges from 20 min to 2 h. The game world offers (discounting downloadable and user-created content), 343 explorable locations

is not possible to discuss each of these case studies individually. As such, we focused on a videogame that presented a rich variety of interaction alternatives. Ideally, the candidate videogame should provide as many traditional aspects of videogame mechanics as possible (seamless and complex combat, free virtual environment exploration, social interaction and user interface manipulation). Ultimately, we chose the open sandbox videogame: The Elder Scrolls V: Skyrim, as it presented all of these aspects to varying degrees.

Skyrim is an open-world action role-playing game where the player must explore an approximately 41 km^2 virtual world (see Fig. 7), filled with dangerous locations that are inhabited by monsters or hostile characters and large cities populated by artificial agents that drive a local economy and may belong to a specific faction. Given its genre, the game allows virtually unlimited freedom of movement and interaction through dozens of gameplay mechanics, thus proving an alluring test bed for our framework.

6.1 Interaction scheme

The game's IS is mainly divided in four components: movement, social interaction, interface manipulation and combat (as exemplified in Fig. 8). Table 4 shows the tested gameplay mechanics and the relation between their native and custom-defined multi-modal natural IS.

6.2 User experience ratings

As previously mentioned, players were asked to answer a forced-choice questionnaire regarding their user experience (measured in terms of "Fun") and preferences on each gam-

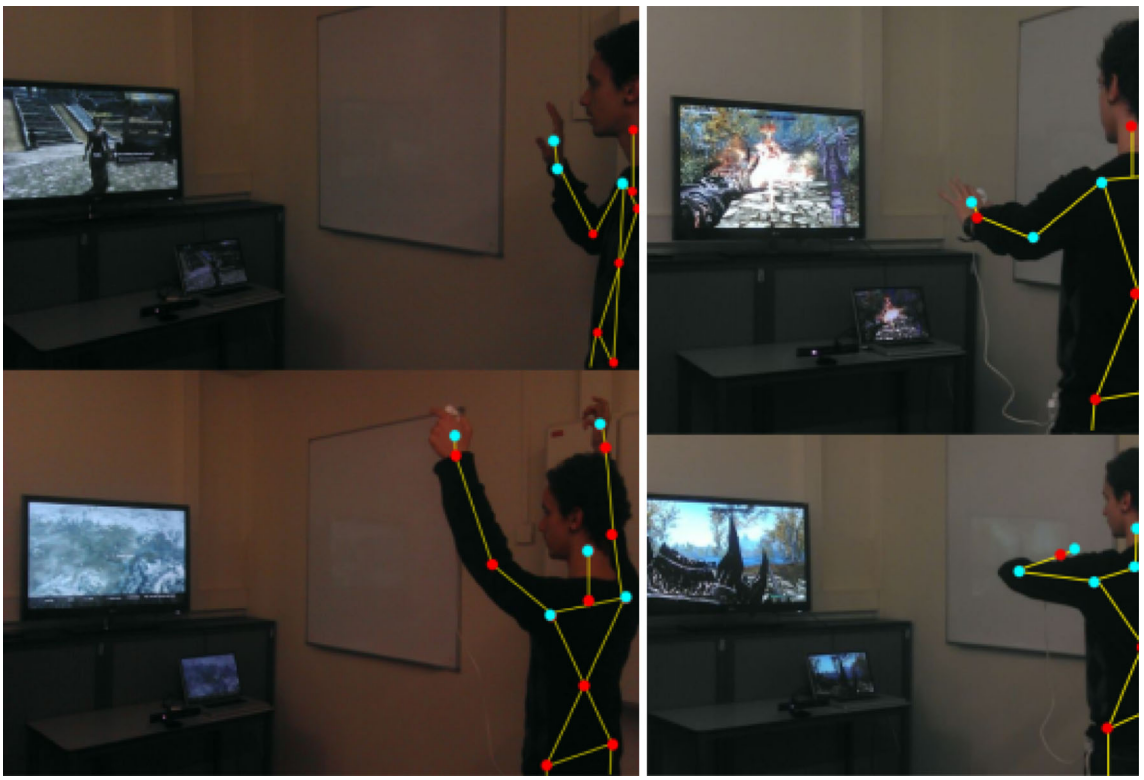

Fig. 8 Four of the defined poses. *Top-left*: the "Hello!" (initiate a conversation) pose. *Bottom-left*: the "Open map" pose. *Top-right*: the "Cast spell" pose. *Bottom-right*: the "Raise shield" pose. On each image, *dots* denote skeleton joints, while the *yellow lines* illustrate the relations between each joint pair. *Red dots* represent joints not used in the pose's constraint set, while *blue dots* represent joints that do belong to the pose's constraint set. Due to interpretability limitations, the constraint relations are not explicitly shown

Table 4 Tested gameplay mechanics

Component	Action	Native-IS	NUI-IS
Movement	Move forward/backward	'W/S'	Right foot forward/backward more than 20 cm
	Strafe left/right	'A/D'	Lean left/right more than 20°
	Orientation (look around)	Mouse cursor	Move Wii Nunchuk in desired direction
	Invoke map	'M'	Outstretched arms
Social interaction/ interface manipulation	Initiate a conversation	'E"'	Wave/say 'hello'
	Quit a conversation	'Tab'	Say 'goodbye' or 'see you soon'
	Buy/sell an item	'Enter'	Say 'buy/sell'
Combat	Equip weapon/spell	'1–8'	Say weapon/spell name
	Attack/cast spell	Mouse click	Push equipped hand forward
	Charge spell	Mouse hold	Raise corresponding arm
	Raise shield	Right mouse click	Arm in front of chest with horizontal orientation
	Charge at enemy	'Alt + W'	Right foot forwards more than 30 cm

ing condition. The questionnaire contemplated each of the three interaction components. All participants played the same game segments to elicit similar experiences. Gaming conditions, however, were randomised to avoid order effects.

Two-tailed paired t tests showed that players found the gaming conditions to have a significantly different impact on their user experience for all interaction categories. However, different patterns emerge on this impact. For the

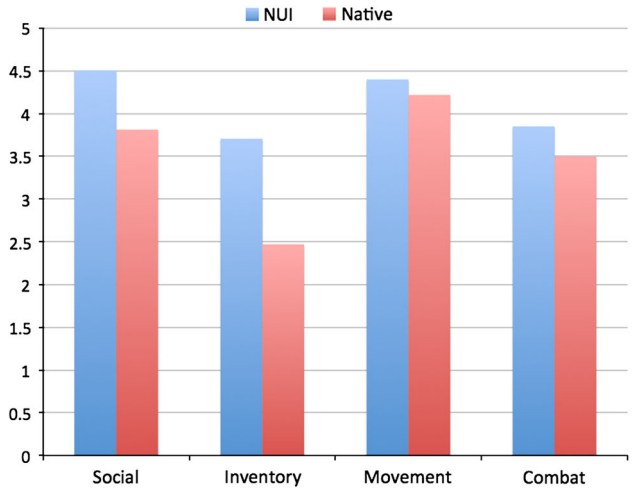

Fig. 9 Average fun ratings for each of the interaction scheme's components by gaming condition

social interaction component, there was a significant difference in the scores for natural interaction (M = 4.50, SD = 0.47) and native (M = 3.81, SD = 0.49); $t(15) = 6.15$, $p = 1.843 \times 10^{-5}$. The magnitude of this effect remain on the same order on the interface manipulation component, with natural interaction (M = 3.70, SD = 0.46) and native (M = 2.47, SD = 0.43) conditions generating the following results; $t(15) = 11.42$, $p = 8.468 \times 10^{-9}$. Significant differences were also observed in the movement and combat components, albeit to a much lesser degree. For the movement component, natural interaction (M = 4.4, SD = 0.53) differed from the native (M = 4.22, SD = 0.37) condition with t(15) = 2.6, p = 0.020; and for the combat component, natural interaction (M = 3.85, SD = 0.54) differed from the native (M = 3.5, SD = 0.60) condition with $t(15) = 2.80$, $p = 0.014$. The average reported Fun values can be examined in Fig. 9 and, along with the aforementioned tests, suggest that while the impact of introducing multi-modal natural interfaces on the social and interface manipulation conditions is more noticeable (lower p values), it still generates higher absolute Fun values for the combat and movement components. This can be trivially attributed to the game's already high Fun values for both these interaction components and is apparent in players' preferences and commentaries, presented in the two following sections.

6.3 Player preference ratings

Regarding their preferences, the majority of players reported preferring the natural to the native interaction scheme on all interaction components. However, there are contrasting patterns on the preference distribution that require further analysis, as players seemed to favour the NUI scheme more highly in the movement (56 to 19 %) and combat (81 to 13 %) interaction components. On the other hand, this preference was much lower in the social (50 to 25 %) and interface manipulation (37 to 19 %) components. These results (see Table 5 for details) show that players found the natural scheme to provide a more enjoyable and intuitive user-experience. However, they also hint that while the NUI scheme may be preferred on all components, its true impact (and thus applicability) lies in the more complex and physically demanding interaction tasks—which seems to agree with the popular notion that players derive enjoyment from their interaction with the game [28].

Table 5 Interaction scheme user preferences

Game feature	Prefer native (%)	Prefer natural (%)	No preference (%)
Social interaction	25	50	25
Interface manipulation	19	37	44
Movement/exploration	19	56	25
Combat	13	81	6
Overall	19	56	25

Overall, users preferred the natural one

6.4 Player opinions

Besides the aforementioned evaluation method, participants were also asked to assess the natural scheme qualitatively. They highlighted that the system's response time (16–25 ms) was "adequate and responsive" (P7). They also gave special emphasis to the possibility of customizing the interaction scheme, by referring that "being able to chose how I want to interact with the world really makes interacting with the game more natural and adds believability to the whole experience" (P3) and "it really allows me to get more into the game because I don't have to remember how each move is mapped" (P12). Participants also pointed out that due to the smaller amount of mix-ups in the natural interface, the learning curve became " faster and more enjoyable" (P9). Finally, participants noted that using the NUI scheme added to the game's emotional aspects of Immersion: "I felt really linked to the character" (P4), "I even tried to avoid getting hit altogether in situations where I wouldn't hesitate to take the damage" (P9) and " I took it much more personally when the guards attacked me and tried to play much more carefully" (P15), which denotes a high degree of empathic and symbiotic connection to the game character. Some participants even reported being able to achieve full immersion: "(I) felt really scared for my actual physical well-being for a moment when that zombie tried to grab me from his coffin" (P11) and " I actually forgot I was playing a game when I tripped that wire trap and almost jumped towards to ground to avoid the falling axe" (P8).

This suggests that using a NUI scheme improves the game's impact considerably, thus creating a more addictive experience. However, further studies are required to correctly assess the truthfulness of this statement and quantify how much faster the learning curve actually is.

7 Conclusions and future work

In comparison with existing works, our approach presents a shorter calibration/controller creation process, while also provides additional features, such as new pose definitions and their automatic recording, support for other devices, speech recognition and complex input mappings without the need for third-party software. However we identified two technical limitations on our implementation. Firstly, that the Kinect™ must be distant from the speakers so as to not interpret voices or sounds coming from the game as voice inputs. Secondly, some actions (e.g. shaking someone's hand, opening a door and casting a spell) require some form of context to be correctly identified, which not feasible without some form of access to the game's engine.

Retrospectively, the system has proved itself capable of delivering an accurate, versatile and satisfactory method for the implementation of multi-modal natural interfaces, as has been proved by our trials. It also succeeded in providing a pleasurable experience in one of the most complex action videogames currently available. In short, some key advantages of our system are:

- It provides a simple yet complete and customizable multimodal natural interface. Our study empirically proves that our system solves two potential problems indicated in [24] and described in Sect. 2.5. First, it grants almost zero latency in every interface with almost no false detections (see Table 1). Secondly, since the player has the ability to create his/her own interaction scheme, this partially solves the potential player's lack of expertise in using natural interfaces. However, difficulties in usage of our system could add up to the player's lack of expertise, but our usability tests (Table 3) shows that users do not require much time to complete all the interface definition tasks.
- It supports multiplayer as in [26]. No only players can use the same pose definitions, but our system also allows for players to use a completely different interaction scheme simultaneously.
- Our empirical study shows that our system is adequate for rather complex games like Skyrim.

Future work should focus on performing a wider set of case studies on both a larger population and multiple game genres, so as to quantify how much physical involvement can benefit the overall experience. Likewise, these additional case studies would allow a better understanding on the interaction challenges that each one of them presents and the actual perceived added value of multimodal NUI gameplay mechanics and the proposed framework's generality.

In parallel, we would also to experiment on the use of additional sensor types, such as for example, head trackers, datagloves, and biometric sensors, which could be used as new interaction means or to gather more precise player involvement data. We also plan on extending our framework to support complete body gestures, which can be achieved by integrating time and speed constraints alongside our spatial constraints. We believe that our automatic pose inference mechanism can be adapted to support such constraints as well, relieving users from a manual constraints definition. This could be integrated on a streamlined version of the configuration GUI, whose future implementation would potentially make the framework available to an even broader population.

Acknowledgments This research was partially funded by the Ph.D. grants with references: SFRH/BD/71598/2010, SFRH/BD/77688/2011 and SFRH/BD/73607/2010.

References

1. Bianchi-Berthouze N et al.(2007) Does body movement engage you more in digital game play? And why?. In: Proceedings of the international conference of affective computing and intelligent interaction, pp 102–113
2. Immersence website (2012) Text and videos available at the Immersence website. http://www.immersence.com. Accessed 10 April 2012
3. Lazzaro N (2004) Why we play games: four keys to more emotion without story. In: Technical report, XEO Design Inc
4. Suma EA et al.(2013) Adapting user interfaces for gestural interaction with the flexible action and articulated skeleton toolkit. J Comput Graph 37(3):247–248
5. Teófilo LF, Nogueira PA, Silva PB (2013) Springer advances in intelligent systems and computing. In: WorldCIST'13 International Conferences, vol 206, pp 873–884. Algarve, Portugal
6. Bernsen NO, Dybkjær L (2001) Exploring natural interaction in the car. In: Proceedings of the CLASS workshop on natural interactivity and intelligent interactive information representation, pp 75–79
7. Santos ES (2011) Interaction in augmented reality environments using kinect. In: Proceedings of the XIII symposium on virtual reality, pp 112–121
8. Linder N (2011) LuminAR: a compact and kinetic projected augmented reality interface. M.Sc. thesis. MIT, Cambridge
9. Vera L, Gimeno J, Coma I, Fernández M (2011) Augmented mirror: interactive augmented reality system based on kinect. In: Campos P, Graham N, Jorge J, Nunes N, Palanque P, Winckler M (eds) Human-Computer Interaction - INTERACT 2011, Lecture Notes in Computer Science, vol 6949. Springer, Heidelberg, pp 483–486. doi:10.1007/978-3-642-23768-3_63
10. Kratz L , Smith M, Lee FJ (2007) Wiizards: 3D gesture recognition for game play input. In: Proceedings of the 2007 conference on future play, Future Play '07. ACM, New York, pp 209–212. doi:10.1145/1328202.1328241

11. Lockman J, Fisher RS, Olson DM (2011) Detection of seizure-like movements using a wrist accelerometer. Epilepsy Behav 20(4):638–641
12. Zhang X et al.(2009) Hand gesture recognition and virtual game control based on a 3D accelerometer and EMG sensors. In: International conference on intelligent user interfaces
13. Yamada T, Hayamizu Y, Yamamoto Y, Yomogida Y, Izadi-Najafabadi A, Futaba DN, Hata K (2011) A stretchable carbon nanotube strain sensor for human-motion detection. Nat Nanotechnol 6:296–301
14. Sakoe H, Chiba S (1978) Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans Acoust Speech Signal Process 26(1):43–49
15. Kinect SDK Dynamic Time Warping (DTW) Gesture Recognition Website (2011) http://kinectdtw.codeplex.com. Accessed 10 April 2012
16. Schwarz LA et al.(2012) Recognizing multiple human activities and tracking full-body pose in unconstrained environments. Pattern Recogn 45:11–23
17. Arantes M, Gonzaga A (2011) Human gait recognition using extraction and fusion of global motion features. Multimed Tools Appl 55:655–675
18. Saon G, Soltau H (2012) Boosting systems for large vocabulary continuous speech recognition. Speech Commun 54:212–218
19. Jelinek F (1998) Statistical methods for speech recognition. MIT Press, New York (ISBN-13:978-0-262-1066-3)
20. Shih PY, Lin PC, Wang JF, Lin YN (2011) Robust several-speaker speech recognition with highly dependable online speaker adaptation and identification. J Netw Comput Appl 34:1459–1467
21. Microsoft Corporation (2012) Kinect sensor manual and warranty. http://download.microsoft.com/download/f/6/6/f6636beb-a352-48ee-86a3-abd9c0d4492a/kinectmanual.pdf. Accessed 9 April 2012
22. Serrano M, Nigay L, Lawson J-YL, Ramsay A, Murray-Smith R, Denef S (2008) The openinterface framework: a tool for multimodal interaction. CHI '08 extended abstracts on human factors in computing systems (CHI EA '08). ACM, New York, pp 3501–3506
23. Schröder M (2010) The SEMAINE API: towards a standards-based framework for building emotion-oriented systems. Adv Hum Comput Interact 2010:article ID 319406
24. McMahan Ryan P, Alexander Joel D, Alon Shaimaa Lazem (2010) Evaluating natural interaction techniques in video games. Proceedings of the 2010 IEEE symposium on 3D user interfaces (3DUI '10). IEEE Computer Society, Washington, DC, pp 11–14
25. Ilves M, Gizatdinova Y, Surakka V, Vankka E (2014) Head movement and facial expressions as game input. EntertainComput 5(3):147–156
26. Edward TSE, Greenberg S, Shen C, Forlines C (2007) Multimodal multiplayer tabletop gaming. Comput Entertain (CIE) Interact TV 5(2):article no. 12
27. Chatzidaki E, Liapis A, Tsironis A (2014) Users' emotional experience using different modalities: a comparative study. Int J Web Eng Technol 9(2):148–163
28. Ermi L, Mäyrä F (2005) Fundamental components of the gameplay experience: analysing immersion. In: Changing views: worlds in play, selected papers of the 2005 digital games research association's second international conference, Vancouver, Canada. DiGRA, pp 15–27